UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica

Enterprise Information Portal: integrazione di servizi Web mediante portlet sviluppato con tecnologia OpenSource

Relatore

Ing. Maurizio Vincini

Tesi di Laurea di

Luca Bonzagni

Correlatore

Ing. Francesco Guerra

Anno Accademico 2001-2002

Ringraziamenti

Desidero ringraziare la prof. Sonia Bergamaschi ed il Gruppo Pro Spa per avermi dato la possibilità di sviluppare questa tesi, l'ing. Maurizio Vincini per la disponibilità ed il supporto fornito durante la stesura.

Ringrazio la mia famiglia ed in particolar modo i miei genitori che hanno reso possibile la realizzazione di questo mio sogno.

Un ringraziamento speciale a Giovanna, per essermi stata così vicina e avermi sopportato in questo periodo così intenso.

Parole chiave:

Enterprise Information Portal
Jetspeed
J2EE
PSML
Integrazione di Servizi

Indice:

Indice delle Figure:

Capitolo 5

Capitolo 6

Capitolo 7

Capitolo 8

Introduzione

La gestione dell'informazione e la creazione della conoscenza, a partire dalle numerose sorgenti di dati raggiungibili tramite la rete, oggi più che mai può rappresentare un fattore determinante nella competitività aziendale. In questo contesto i portali hanno assunto un ruolo fondamentale e si sono evoluti fino ad arrivare agli attuali Enterprise Information Portal (EIP), applicativi che integrano un'architettura e un insieme di tecnologie che rappresentano lo stato dell'arte nell'ambito dei sistemi distribuiti.

Gruppo Pro Spa rappresenta una delle aziende leader nel settore dell'Information Comunication Tecnology (ICT) italiano e come fornitrice globale di soluzioni informatiche per le imprese individua nella gestione strutturata dei dati e delle informazioni presenti nella rete uno dei bisogni fondamentali delle aziende, ed in particolare dei propri clienti. La tendenza alla realizzazione di extended enterprise ossia di aziende capaci di estendersi, grazie a sistemi informativi, oltre ai propri confini fisici aziendali, necessita di infrastrutture informatiche, quali gli EIP, in grado di fornire un unico punto di accesso integrato a tutti quei servizi ed applicativi ormai incorporati nella maggior parte delle strutture aziendali.

La necessità di realizzare questi applicativi implica, da parte di aziende produttrici di soluzioni informatiche, dover affrontare una scelta tra vari prodotti attualmente offerti nel mercato dell'ICT, e fra le varie tecnologie disponibili. Realizzare servizi integrati in un Enterprise Information Portal utilizzando prodotti commerciali, completi di avanzati tool di sviluppo e servizi già implementati, può significare una notevole diminuzione sui tempi di sviluppo, a fronte di un aumento dei costi dovuto al pagamento delle licenze d'uso. Per aziende che dispongono di avanzati laboratori di sviluppo, come GruppoPro, la tecnologia Open Source rappresenta sia un'opprtunità per l'assenza di licenze d'uso, sia uno sforzo maggiore per la realizzazione, manutenzione ed integrazione di nuovi applicativi. I prodotti OpenSource infatti non sono forniti di tool di sviluppo e l'integrazione dell'Eip con eventuali sistemi informativi precedentemente sviluppati rappresenta una difficoltà. La problematica esposta, individuata durante il periodo di stage trascorso presso Gruppo Pro Spa, ha portato alla necessità di approfondire le varie tecnologie impiegate nella gestione delle informazioni e dei servizi nell'ambito degli EIP e quindi alla realizzazione di questa tesi, nella quale si sono:

1. Analizzate le architetture, le funzionalità ed i servizi offerti dai prodotti nell'ambito degli EIP commercializzati da aziende come IBM, SAP e BEA leader mondiali nel settore informatico.
2. Analizzate le tecnologie ed i servizi offerti in modalità OpenSource da un prodotto di punta come Jetspeed fornito dal centro di ricerca e sviluppo Jakarta.
3. Implementati alcuni servizi con l'uso della tecnologia OpenSource nell'ambito Universitario, per testare eventuali limiti e difficoltà nel realizzare applicazioni con questo prodotto.

La tesi sviluppata, composta di otto capitoli, è suddivisibile in tre macro sezioni che rispecchiano gli obiettivi descritti precedentemente:

Prima Sezione:

- Nel primo capitolo della tesi viene svolta un'analisi storica che evidenzia quali esigenze e quali tecnologie hanno caratterizzato la nascita dei Portali Web e la loro evoluzione fino ad arrivare ai moderni Enterprise Information Portal (EIP). Lo scopo del capitolo è di introdurre il concetto di EIP e definire il suo ruolo fra i vari servizi presenti nel contesto del World Wide Web.
- Nel secondo capitolo vengono analizzati i servizi e gli applicativi standard integrati nella tecnologia degli EIP come la customization, l'integrazione con sistemi transazionali, database gerarchici, accessi ai web Services, applicativi per il Knowledge Management e la Business Intelligence. Nel capitolo viene inoltre proposta una architettura standard che soddisfa la maggior parte degli Enterprise Information Portal attualmente in commercio.
- Il terzo capitolo offre un'analisi dettagliata della piattaforma Java 2 Enterprise Edition (J2EE) che rappresenta oramai uno standard de facto per la maggior parte della applicazioni Web-based.
- La prima parte del quarto capitolo analizza singolarmente i prodotti EIP attualmente commercializzati da IBM, SAP e BEA, aziende leader mondiali nel settore informatico. Nella seconda parte viene invece presentato un parallelismo atto a confrontare le architetture ed i servizi offerti dai pacchetti analizzati.

Seconda Sezione:

- Con il quinto capitolo si introduce in modo dettagliato la tecnologia integrata e l'architettura sulla quale si basa Jetspeed: il prodotto più completo e di punta attualmente disponibile in rete per implementare un EIP attraverso un prodotto Open Source. Scopo di questo capitolo è analizzare le Api ed in particolare il funzionamento di un portlet per sviluppare un servizio ed integrarlo nel portale.
- Il sesto capitolo continua l'analisi di Jetspeed soffermandosi principalmente sugli strumenti dedicati alla pubblicazione dei contenuti su Web e presentando l'approccio di sviluppo dettato dal *Model-View controller* (MVC) . Gli strumenti analizzati per la pubblicazione sono principalmente l'utilizzo di Velocity e delle Java server pages (JSP).
- Il capitolo sette descrive come implementare e configurare la piattaforma J2EE con *Tomcat* come Servlet Engine e Server Web e *Jetspeed* come EIP, utilizzando Ant per il deployment delle web application.

Terza Sezione:

- Nell'ultima sezione della tesi, rappresentata dal capitolo otto, si descrive il portlet implementato e le problematiche individuate per integrare, nel contesto di un EIP, i servizi dedicati ai docenti disponibili sul sito di facoltà. Per la realizzazione del portlet si sono analizzate le procedure standard di autenticazione ed in particolar modo le procedure di autenticazione utilizzate per l'accesso ai singoli servizi universitari quali:
  - Amministrazione pagina docente
  - Accesso alla rete Intranet
  - Gestione E-mail
  - Inserimento news di facoltà
  - Inserimento news studenti
  - Visualizzazione news-group
- L'uso di questo applicativo nel contesto del portale permette di fornire all'utente un'unica pagina di accesso ai servizi sopra citati (altrimenti dispersi all'interno del sito della facoltà,) e permettere una autenticazione trasparente all'utente senza dover ogni volta ridigitare le rispettive password e nome utente. Lo scopo dello sviluppo del portelt è quello di integrare in un Enterprise Information Portal sviluppato con Jetspeed, i servizi Web sopra citati ed analizzare le difficoltà e gli eventuali limiti nell'implementare applicazioni web con l'uso dell'attuale tecnologia OpenSource.

A conclusione della tesi è stata inserita una sezione nella quale si analizzano i limiti e le opportunità individuate nello sviluppare un EIP con tecnologia OpenSource.

Capitolo 1

Nascita ed evoluzione dei portali

Computers connessi in una vasta rete globale creano uno spazio virtuale chiamato cyberspazio. La spina dorsale del cyberspazio è internet, ovvero, un insieme di reti decentralizzate che creano un'infrastruttura atta a trasmettere e condividere informazioni sia in ambito locale che globale.

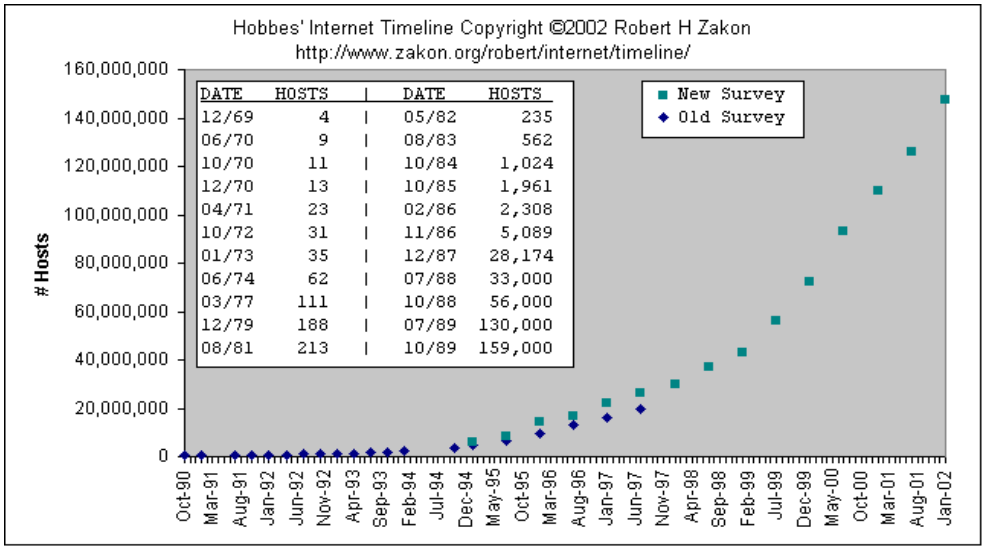
La connessione delle reti insieme al world wide web rappresentano le fondamenta sulle quali si è sviluppata la "Information Economy": un'economia nella quale l'informazione ha assunto un valore equiparabile a quella di qualsiasi altro materiale. Per poter ottenere valore aggiunto e quindi una maggiore competitività rispetto agli eventuali concorrenti, nell' Information Economy non basta più raggiungere l'informazione ma occorre saperla gestire e strutturare in base alle esigenze e preferenze dei singoli individui così da creare un accesso personalizzato al dato. Affinché il dato possa realmente divenire protagonista nella competitività aziendale, occorre manipolarlo e plasmarlo a seconda delle singole necessità professionali, fino a trasformarlo in vera e propria conoscenza.

Un qualsiasi sito internet che fornisca un punto di accesso organizzato alle informazioni presenti nella rete, costituisce metaforicamente una porta di accesso al Web e viene comunemente chiamato Portale Web (*Web Portal*). Fra i portali più conosciuti si possono citare Yahoo!, Excite, MSN, AOL, i quali forniscono servizi di ricerca delle informazioni all'interno della rete ed altri tipologie di servizi che collegano l'utente ai dati digitalizzati di internet. Gli Enterprise Information Portal (EIP) sono applicativi informatici che evolvono la tecnologia dei portali Web allo scopo di fornire all'utilizzatore un accesso personalizzato ad insiemi di dati interni o esterni ad un'impresa, fornendo ad esso una sorta di desktop configurabile nel contenuto e nella rappresentazione delle informazioni. Al fine di realizzare tale desktop per accedere ai vari servizi, applicativi di business e risorse aziendali offerte sia internamente che esternamente all'impresa, occorre riferirsi ad un'architettura ed un insieme di tecnologie che rappresentano lo stato dell'arte nell'ambito dei sistemi distribuiti.

L' EIP nasce quindi come un bisogno, indotto dalle nuove tecnologie, di standardizzare l'accesso alle informazioni, al fine di aumentare la quantità e la qualità di servizi fruibili dal web, ed evolve una tecnologia nota con il nome di Web Portal. Un'analisi storica per seguire l'evoluzione dei Portali Web dalla loro origine fino ai moderni e potenti portali aziendali è necessaria per poter individuare la collocazione di tale tecnologia nel vasto mondo delle reti.

1.1 World Wide Web e Web browser

Con la nascita del world wide web avvenuta nel 1989 ad opera di Tim Bernerst Lee e del primo browser web "mosaic" nel 1994 ad opera di Jim Clark e Mark Adressen (fondatori della Netscape Comunication), l'utilizzo di internet divenne molto intuitivo e slegato dagli aspetti più tecnici delle connessioni di rete. La tecnologia di questi anni introduce infatti la possibilità di relazionare fra loro documenti situati anche su database diversi, magari dislocati su host sparsi geograficamente, con il semplice click del mouse su determinate parole chiave (*ancore*). La possibilità di navigare in rete con l'uso di questa nuova tecnologia chiamata *Link*, segna l'inizio degli ipertesti distribuiti, ossia di quei documenti non necessariamente lineari, integranti altri testi. La semplicità di utilizzo e le alte potenzialità di questo strumento, unite alla successiva distribuzione gratuita di Mosaic (1995) furono gli ingredienti che portarono al decollo dei siti internet e delle pagine web ("Web Page"). La distribuzione su larga scala di web browser, capaci di supportare documenti multimediali, consente ai semplici ipertesti di integrare anche dati di tipo audio e video, e quindi evolversi in ipertesti multimediali.



Hosts = a computer system with registered ip address

Figura 1: - Decollo di Internet –

Lo sviluppo di questa fortunata tecnologia, nei primi anni novanta, comportò una notevole espansione della rete dovuta alla creazione e successiva pubblicazione di numerosi siti e pagine web. Questa crescita esponenziale della rete ebbe come conseguenza il trasferimento di una grosse mole di informazioni nel cyberspazio, presentando così agli utenti un nuovo problema. L'informazione era tecnicamente disponibile, a portata di tutti, ma difficilmente reperibile in quanto l'unico modo per accedervi era conoscere il nome logico o l'identificativo IP dell'host contenete le informazioni ricercate. Nasce il bisogno di semplificare l'accesso e la ricerca delle informazioni contenute nei siti web, per renderne possibile l'utilizzo da parte degli utenti.

1.2 Motori di ricerca come Entry Point

Una prima soluzione al problema della reperibilità del dato fu proposta, nel 1994, da due studenti di Ingegneria Elettronica della Standford University della California. Jerry Yang e David Filo svilupparono un'applicazione capace di catalogare siti web per argomento e successivamente ricercarli e fornirli agli utenti in base alle loro specifiche richieste. Tale applicazione venne installata nel server della Standford University e divenne il primo servizio fruibile in internet attraverso il sito noto come "Jerry Yang's Guide to WWW". Constatato il successo dell'applicazione i due studenti fondarono tempestivamente YAHOO!: la prima società resa famosa attraverso il suo sito web. Il servizio fornito da YAHOO! non aveva le caratteristiche di un vero e proprio motore di ricerca ma ne fù il presupposto. La struttura era fondamentalmente quella di una directory ossia un elenco di siti web (e quindi non solo di pagine) suddivisi per argomento, la cui classificazione non avveniva in modalità automatica, ma bensì attraverso una precedente segnalazione manuale. Tale organizzazione ad albero permetteva agli utenti di ottenere agevolmente una suddivisione di siti per tipologia, isolando solo quelli relativi allo specifico argomento di interesse.

Il grande successo di Yahoo! fece sì che in pochi anni tante altre società nacquero per imitazione presentando, attraverso il loro sito, un modo facile e strutturato per accedere alle informazioni presenti nel web: divennero questi i famosi ed indispensabili starting point ai quali la maggior parte degli utenti si riferiva per iniziare la navigazione in Internet. Numeri elevati di utenti, concentrati su poche pagine web, attrassero l'attenzione dei pubblicitari ("*Advertiser*") il cui scopo era di catturare e trasportare gli utenti da questi "*Entry Point*" verso i loro siti aziendali, posizionando inserzioni e link all'interno delle pagine web. Per fare questo gli Advertiser pagavano i proprietari degli entry point (come Yahoo, Msn, Excite, Altavista…) in base al numero di utenti che giornalmente visitavano il sito e quindi in base alla visibilità di cui il loro logo godeva giornalmente. Una tale competizione legata al traffico di utenti nei siti, spinse i proprietari degli entry point a differenziarsi gli uni dagli altri per individuare un target di utenti più definito e poter vendere gli spazi pubblicitari a prezzi più elevati.

Oltre alla specializzazione riguardante precisi argomenti nacquero nuovi motori di ricerca "*search engine"*, capaci di sviluppare tecniche di gestione e recupero dati sempre più raffinate. Il loro differente funzionamento, rispetto alle directory, consisteva e consiste ancora oggi, nel censire i siti Web in base alla rilevanza delle parole contenute in ogni pagina del sito, evidenziando quelle riportate più spesso, presupponendole rappresentative dell'argomento principale della pagina stessa. Per fare questo, oltre ad inserire i siti segnalati dagli utenti, scandagliano continuamente l'intero Web attraverso degli specifici software (i cosiddetti spider o crawler), acquisendo tutte le pagine non ancora presenti nei loro archivi. Tra i primi ad applicare questa tecnica e tuttora tra i più utilizzati, si possono segnalare Lycos, Excite, HotBot e Google.

Alcuni Search Engine che si specializzarono nelle ricerche per argomento sono:

- **Medscape.com** per ricercare informazioni mediche ;
- **AOL, Yahoo, Virgilio, Excite** per ricercare categorie comuni;
- **CNET** per argomenti legati allo sviluppo tecnologico;

- **Safety.com** per ricercare informazioni riguardo la sicurezza;

Quindi il motore di ricerca rappresentava un punto di accesso alle informazioni contenute in altri siti, i quali non erano altro che pubblicazioni statiche o semplici locandine pubblicitarie. Avere il sito internet da parte delle aziende nei primi anni novanta era un modo come un altro per pubblicizzarsi e rendersi visibili anche via Web. Oltre a semplici informazioni di tipo storico, economico e geografico erano infatti poche le società che presentavano su internet la possibilità di consultare i listini di vendita, di visionare i loro prodotti ed ancora meno erano le aziende che ne permettevano l'acquisto via internet. Le prime pagine del web offrivano quindi scarsi servizi .

1.3 Esordio del Web Portal

L'elevato indotto derivante dalla vendita di spazi pubblicitari all'interno delle pagine web più visitate, unito alla quotazione in borsa delle società proprietarie di alcuni dei più famosi entry point, concentrarono grossi capitali in questo nuovo mercato. Una tale euforia economica instaura un meccanismo vorticoso che porta tali società ad escogitare soluzioni per aumentare ulteriormente le visite degli utenti sul loro sito. Si cominciarono così ad investire i capitali guadagnati per sviluppare servizi aggiuntivi da affiancare al motore di ricerca; lo scopo era di attirare l'utente periodicamente sul sito fornendogli non più solo un servizio di starting point, ma una serie di utili applicazioni che necessitassero una continua presenza dell'utente sul sito. E' grazie alla disponibilità dei sopra citati capitali che società come Yahoo si permisero l'acquisizione di fornitori gratuiti di e-mail ("free e-mail provider") come rocketmail.com o ancora meglio l'acquisizione di fornitori di siti web fisici come geocities. Tali servizi inducevano gli utenti alla creazione e manutenzione di siti web personali, all'utilizzo dell'e-mail e quindi ad un uso più intensivo degl'entry point permettendo così l'aumento del costo delle inserzioni pubblicitarie. Il sito web contenente il motore di ricerca noto fino ad ora come entry poin o starting point si evolve fino a divenire un fornitore di servizi gratuiti disponibili ai propri utenti e diventare quello che oggi conosciamo come portale Web (o "W*eb po*rtal"). I servizi che vengono integrati nel portale in questa fase sono di tanti tipi e di varia natura ; si dividono infatti in servizi di

<u>Informazione:</u>

- Notiziari flash;
- Approfondimenti sulle principali notizie;
- Notizie di Sport;
- Meteo;
- Lavoro;
- Mappe;

Intrattenimento & tempo libero:

- Condivisione foto/hard-disk;
- Chat & newsgroup;
- Aste on line;
- Oroscopo;
- Giochi on-line;
- Concorsi e quiz a premi;

Utility Personali:

- E-mail;
- Sms;
- Money;
- Agenda;

Anche se, come previsto, il numero di utenti che giornalmente navigano sul portale aumenta grazie all'utilizzo dei servizi sopraccitati, non altrettanto proporzionalmente aumenta il costo di una inserzione pubblicitaria all'interno del portale. Il rapporto di proporzionalità instauratosi fra quantità di traffico e prezzo di un'inserzione pubblicitaria aveva già raggiunto precedentemente i limiti massimi, portando alla saturazione quel vortice economico che fino a questo momento aveva contraddistinto lo sviluppo e assicurato la crescita economica e tecnologica dei portali. Gli indotti pubblicitari risultano inoltre insufficienti a coprire le spese di sviluppo e gestione dei nuovi servizi e questo induce i gestori dei portali a cercare nuove tipologie di finanziamenti e di ricavi. Lo sviluppo tecnologico informatico raggiunto unito alla disponibilità di connessioni a maggior ampiezza di banda, permettono ai portali di creare e fornire, previo pagamento, servizi qualitativamente avanzati.

Inizia in questo modo la seconda fase del Web Portal, caratterizzata dal fatto che i ricavi del portale non dipendono più solo dagli indotti pubblicitari legati alla quantità di traffico del sito, ma dipendono ora anche dalla qualità ed utilità dei servizi offerti dal portale. Il Web Portal diventa a tutti gli effetti un fornitore di servizi e come ogni altra attività commerciale deve competere con i suoi concorrenti, cercando di fornire prodotti strategici capaci di dare valore aggiunto alla competitività dei propri clienti. Assistiamo per esempio alla creazione di veri e propri centri commerciali virtuali paralleli ai motori di ricerca, nei quali i molti utenti del portale possono comprare o solo consultare i prodotti esposti dai venditori; i quali, a loro volta, pagano il portale affinché questo gestisca oltre che l'interfaccia grafica del negozio virtuale (front-end) anche l'aspetto distributivo della vendita. Proprio la logistica e l'outsourcing di processo sono infatti fra i primi servizi che vengono forniti dal portale ai propri clienti ed ai propri visitatori. La tecnologia dell' E-commerce ed il raffinamento di tecnologie informatiche per la gestione del commercio elettronico permettono in questi anni agli utenti di usufruire di alcuni servizi integrati ed aggregati in pagine web chiamate Web Portal.

1.4 Dal web portal all' Enterprise Information Portal

Il recente sviluppo unito al massiccio utilizzo di sistemi di Enterprise Resource Planning (ERP) e di datawarehousing, combinato alla creazione di sistemi di immagazzinamento dati in sistemi legacy Mainframe e client-server, ha causato un incremento esponenziale dei dati digitali aziendali; fra questi si individuano dati di tipo transazionale, video files, audio files, manuali dei prodotti, informazioni riguardo i clienti, e-mail, ecc… i quali rappresentano la base della conoscenza che un'azienda ha maturato nel corso del suo sviluppo. Se si considera inoltre che secondo un'analisi pubblicata su "Building Corporate Portal using XML", solo il 10% di tutti questi dati viene strutturato nei database relazionali, in modo tale da rappresentare una sorgente di informazione e conoscenza, risulta che circa il 90% dei dati potenzialmente utili rimane bloccato all'interno dei sistemi di immagazzinamento e non viene quasi mai utilizzato. L'incapacità di gestire questi dati non strutturati implica che una potenziale base di conoscenza aziendale risulta presente nei sistemi ma non è strutturata in modo da essere accessibile e quindi non partecipa nel fornire conoscenza e quindi valore aggiunto all'impresa.

La necessità di creare un unico punto di accesso a gruppi di dati eterogenei, di formati diversi e distribuiti fra più sistemi di immagazzinamento e fra vari dipartimenti aziendali distribuiti geograficamente sul territorio, ha portato ad applicare quello che oggi è la tecnologia internet anche nelle strutture di connessione aziendali. Considerato che ormai tutte le imprese hanno abbandonato le vecchie connessioni dirette sostituendole con le attuali reti intranet, sfruttando la tecnologia hardware di internet, risulta immediato pensare che la necessità di queste aziende è ora di creare un portale di accesso alle informazioni ai vari servizi offerti dall'azienda (raggiungibili tramite le nuove reti), sfruttando ed evolvendo quelle tecnologie utilizzate per la creazione dei portali web. Considerato che il primo bisogno dell'utente aziendale è quello di avere un accesso significativo nei contenuti e personalizzato alle informazioni, a seconda del proprio ruolo professionale, il portale aziendale chiamato Enterprise Information Portal (EIP) si sviluppa inizialmente come una estensione di quelli che sono i sistemi di datawarehousing e si evolve successivamente divenendo un centro personalizzato per la gestione e l'utilizzo dei servizi aziendali. Gli attuali e moderni portali dedicati alle imprese, sono tipicamente un miscela delle moderne tecnologie informatiche ed includono:

- database gerarchici e multidimensionali,
- architetture client/server,
- Interfacce grafiche,
- Server per la gestione delle informazioni
- Tool di ricerche avanzate (search-tool)
- Motori automatici per la classificazione delle informazioni
- Motori di Estrazioni dati da altri programmi,
- Direttori
- Tool di per l'estensione dei linguaggi di markup

Una espansione ed integrazione di tecnologie eterogenee in brevi tempi è stata possibile in quanto si è potuto sfruttare tecnologie hardware come connessioni di reti e tecnologie software, come l'utilizzo di protocolli e web browser, ormai pienamente

testate e standardizzate nel mondo internet. L'unificazione di questa base tecnologica unita alla sua stratificazioni gnoseologica ed all'elevata standardizzazione raggiunta, ha permesso alle aziende informatiche la creazione di nuovi servizi e la vendita di questi ad un numero maggiore e più eterogeneo di clienti. L'industria informatica ha potuto sfruttare grosse economie di scala sia in ambito hardware che software .

La capacità di gestire il dato indipendentemente dalla sua forma, tipologia e contenuto permette oggi anche l'intercomunicabilità fra sistemi e processi che si sono sempre rivelati monoliti proprietari chiusi, incapaci di comunicare con sistemi e processi esterni se non dopo un'opera manuale di processazione del dato atta a renderlo compatibile alle esigenze del programma o del processo destinatario. Tale informatizzazione permette oggi all'azienda di migliorare ed ottimizzare la maggior parte dei processi legati alla gestione del proprio business, rendendo possibile una stretta comunicazione ed una condivisione di conoscenza fra:

- Fornitori
- Terzisti
- Rete di vendita
- Clienti

La suddetta condivisione rende effettivamente possibile la realizzazione di un Sistema "*Extended enterprise*" che estende l'impresa oltre i confini fisici aziendali inglobando ed unificando tutti i quei processi di sviluppo che la caratterizzano, riuscendo a condividere dati ed unificare processi anche con sistemi esterni, minimizzando in questo modo una serie di ridondanze applicative che fino ad oggi hanno appesantito non solo le strutture tecnologiche informatiche ma anche il modo di fare Business.

Capitolo 2

Caratteristiche degli EIP e tipologie di servizi offerti

Al giorno d'oggi ogni persona all'interno di un'impresa è un lavoratore che necessita e sviluppa conoscenza. Per istituire delle unità lavorative occorre fornire ai componenti informazioni provenienti da sistemi eterogenei e disparati. Il portale rende l'informazione accessibile, pertinente e cosa più importante aiuta gli utilizzatori a gestire i dati dell'impresa in modo da sfruttare al meglio la conoscenza aziendale al fine di ottenere un concreto valore aggiunto. L'accessibilità alle applicazioni di servizio e di integrazione delle varie sorgenti di informazione sono i requisiti chiave per fornire un efficiente accesso alla varietà di informazioni richieste dai vari utenti che lavorano in una azienda estesa (*Extended Enterprise*). Obiettivo del capitolo è trovare un approccio comune alla risoluzione di queste necessità ed individuare una serie di servizi base che vengono inclusi negli attuali Enterprise Information Portal.

2.1 Portal Services

I servizi offerti da un Enterprise Information Portal sono classificabili secondo due categorie distinte:

- Integrazione dei contenuti provenienti da sorgenti eterogenee;
- Personalizzazione del portale gestita dagli utenti sia per la definizione dei contenuti che del Layout.

Una tipica interfaccia grafica di un portale si presenta quindi come in Figura, nella quale sono state evidenziate diverse aree che delimitano i servizi offerti e le particolari sezioni che caratterizzano comunemente i portali:

- Customization,
- Personalization,
- Portlets.

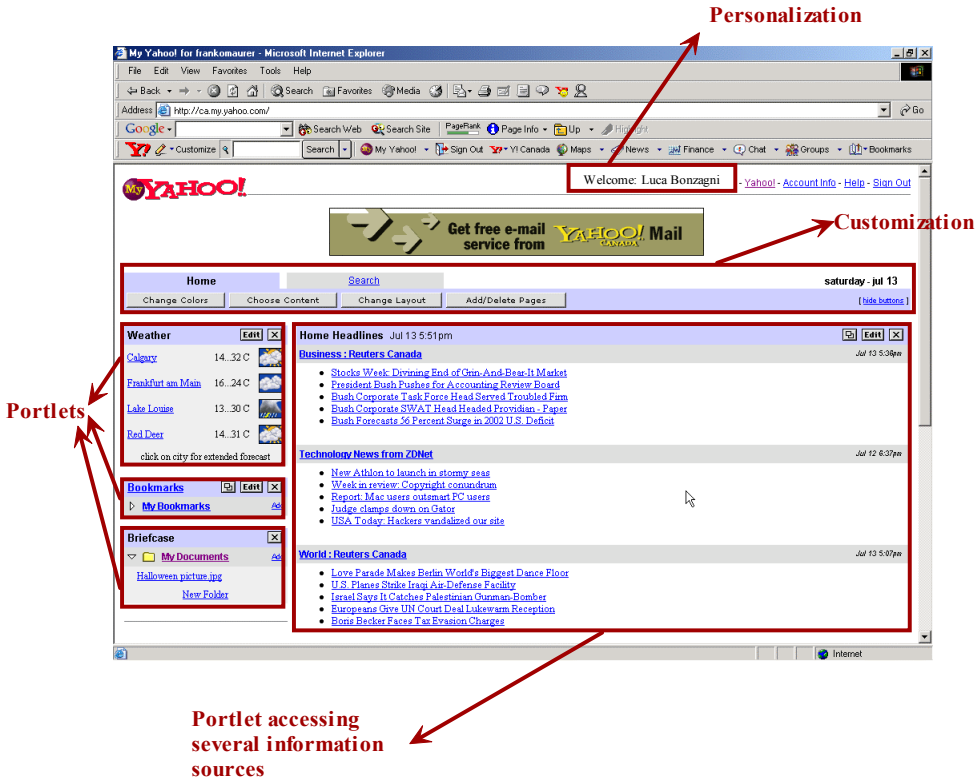


Figura 2: - Esempio di Portale –

Nei prossimi paragrafi saranno analizzati in dettaglio i servizi offerti:

2.1.1 Personalization

La sezione di Personalization rappresenta la fase conclusiva del processo di autenticazione dell'utente all'EIP. Quando l'utente accede all'Enterprise Information Portal deve infatti eseguire una procedura di Log-In attraverso la quale il portale riconosce l'utente e gli mette a disposizione i servizi e le informazioni a seconda dei diritti di accesso ad esso associati dall'amministratore del sistema..

2.1.2 Customization

Nella zona di customizzazione, viene data la possibilità all'utente di decidere il proprio profilo. In particolare l'utente può decidere di:

- Aggiungere o togliere servizi, scegliendo fra quelli resi disponibili dall'EIP durante la fase di Personalization. (vedi paragrafi successivi).
- Cambiare i colori di visualizzazione
- Modificare il Layout della pagina e dei relativi servizi integrati

2.1.3 Content aggregation

Caratteristica fondamentale di un Enterprise Information Portal è la capacità di creare un unico punto di accesso ai servizi ed alle informazioni provenienti da qualsiasi struttura organizzativa aziendale e non. Nel caso particolare dei portali rivolti alle imprese, occorre analizzare ipotetici modelli di diversi utilizzatori, al fine di individuare e classificare le sorgenti di informazioni che normalmente vengono utilizzate. Da questa analisi svolta da SAP e riportata su un articolo pubblicato on line al sito http://www.sap.com risulta che le sorgenti standard di informazioni utilizzate dai lavoratori possono essere suddivise in quattro categorie principali:

- applicazioni e database gerarchici,
- business intelligence,
- documenti non strutturati, contenuti,
- servizi fruibili dal web.

Al fine di unificare queste sorgenti di informazioni e per meglio individuare i dati chiave, fra importanti quantitativi di dati che costantemente vengono accumulati in tali sistemi, viene usata una tecnologia informatica in grado di correlare e capire le connessioni logiche fra i dati ricavati dalle varie sorgenti. Tale stratificazione software deve inoltre percepire le esigenze dell'utilizzatore, la sua autorità e quindi i relativi permessi di accesso ai dati. Il Content Aggregation deve quindi presentare l'informazione e le connessioni logiche con le altre, ogniqualvolta l'utente ne necessiti .

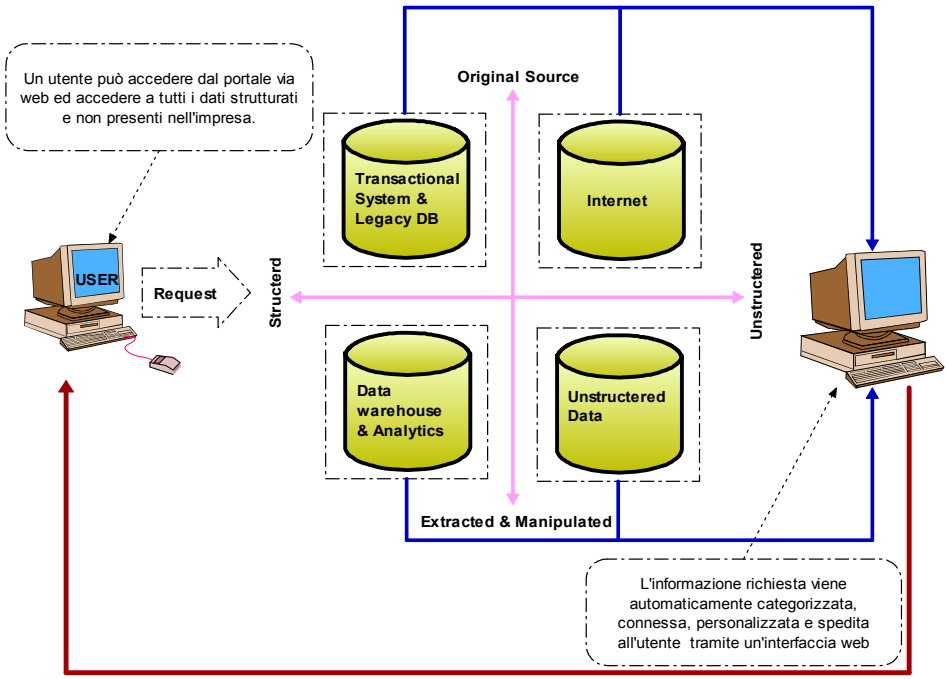


Figura 3: - Ricavare l'informazione da dati strutturati e non –

Creare un ambiente di navigazione integrato fra le diverse tipologie di sorgenti, di applicazioni e di dati, mantenendo una coerenza nel contesto di ricerca, può essere raggiunto solo creando un mondo unificato di dati, metadati e componenti applicativi. Questa collaborazione distribuita di conoscenza è ottenuta grazie un processo di astrazioni di varie procedure fondamentali che regolano il modo di fare business e dalle relazioni fra e attraverso applicazioni di vario genere.

2.1.3.1 Accesso ai sistemi transazionali e database gerarchici (integrazione con il datawarehouse e sistemi ERP)

Le applicazioni delle imprese contengono sia informazioni critiche che logiche applicative utilizzate per la risoluzione dei bisogni transazionali delle aziende. Database gerarchici contengono la maggior parte dei dati che giornalmente vengono manipolati dai processi aziendali attraverso programmi, schermate e reports, creando spesso replicazione del dato. Ancora oggi sono proprio queste basi dati, ottenute attraverso un processo di acquisizione affiancato allo sviluppo aziendale, a rappresentare le fondamenta informative che le imprese utilizzano maggiormente. In molti casi la mancanza di consistenza di questi dati memorizzati e la diversità dei sistemi che li gestiscono, abituano l'utente a servirsi e fidarsi solo di pochi applicativi, limitando la possibilità di accedere ad altre tipologie di informazioni (gestite da quegli applicativi da lui non conosciuti). La tecnologia EIP integra un sistema in grado di fornire all'utilizzatore la conoscenza di vari eventi e permette la rapida creazione di blocchi di informazioni estrapolandoli dalle varie applicazioni aziendali; tale sistema crea un comodo livello di astrazione dei dati per gli sviluppatori ed un elevato grado di flessibilità per coloro che devono gestire le informazioni. Per fornire una unificazione contestuale dei dati, ottenuti da questa astrazione, occorre un ulteriore sistema capace di ricercare i dati a seconda del contesto e rappresentarli in modo unificato all'utilizzatore il quale avrà un documento dinamico ottenuto dall'integrazione di più dati e processi aziendali. Per ottenere un livello di astrazione dei dati distribuiti nelle procedure aziendali, l'EIP deve sapersi interfacciare con il **datawarehouse** e i sistemi di **Enterprise Resource Planning (ERP)** aziendali**.**

Integrazione con il Datawarehouse e sistemi ERP:

Per poter descrivere quali relazioni intercorrono fra un Enterprise Information Portal (EIP) ed un data warehouse aziendale è necessario definire preventivamente che il data warehouse in ambito intranets è un ambiente tecnologico progettato per assemblare e gestire correttamente dati provenienti da sorgenti diverse al fine di ottenere una visione dettagliata di un intero sistema economico. In modo preciso il [data] warehouse è una raccolta di dati integrata, permanente, variabile nel tempo orientata a precisi argomenti, a supporto di decisioni manageriali e di qualsiasi altro livello della struttura aziendale. Per poter interrogare tale raccolta di dati memorizzati nei database in modo veloce e preciso, il datawarehouse dispone di semplici tools che l'utente può utilizzare in tempo reale per l'accesso alle informazioni.

Se pensiamo al concetto di EIP ed alla descrizione data di un data warehouse , è condivisibile l'idea espressa da Clive Finkelstein e Peter Aiken, nel libro "Building Corporate Portals with XML", secondo i quali l'Enterprise Information Portal rappresenta l'evoluzione del concetto di datawarehouse, in particolare affermano che: "*An Enterprise Portal extend theBusines of a data warehouse to intranets and the internet*". Considerato che il portale ha la capacità di estendere l'impresa oltre ai semplici confini fisici e comunicare con altri processi, prima scollegati fra di loro, i dati che entrano attraverso l'EIP nei warehouse aziendali rappresentano ora una estensione di quelle che erano considerate le sole informazioni aziendali. Occorre quindi distinguere dati già strutturati da quelli che sono invece non strutturati:

STRUCTURED DATA: Allo scopo di accedere ad informazioni che sono strutturate all'interno di applicazioni utilizzate durante i processi della catena del business aziendale (come gli ERP), il portale aziendale integra tecnologie di business intelligence e di ETL (Extract, Transform, and Load), capaci di estrapolare da questo ambiente OLAP (On Line Analytical processing) dati rilevanti e fondamentali per la conoscenza aziendale.

UNSTRUCTERED DATA: Anche se l'accesso ai dati strutturati, è vitale, abbiamo già descritto come questi rappresentino solo una minima parte rispetto a tutta la conoscenza che viaggia invece in strutture informali come newsletter, e-mail, appunti elettronici o altro.

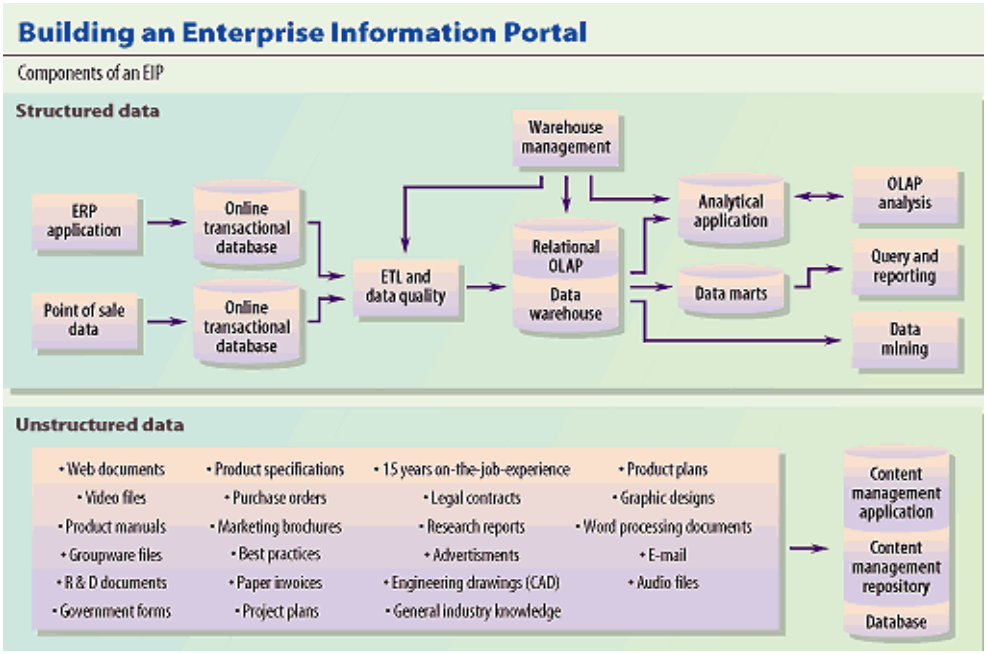


Figura 4: - Enterprise portal Concepts: [InfoWorld] Web site –

Risulta quindi fondamentale che il portale sia affiancato anche da motori capaci di scorrere all'interno di questi dati non strutturati, crearne classificazioni ed archivi che saranno poi consultati in base alle richieste ed esigenze dei singoli utenti. Al fine

di gestire un richiamo totale dei dati da qualsiasi sorgente, l'EIP deve essere costruito con l'ausilio di componenti e strutture capaci di integrarsi nelle varie sorgenti e nelle varie applicazioni come mostrato nella Figura sopra.

2.1.3.2 Business Intelligence

Le soluzioni di business Intelligence integrate nell'EIP permettono una completa ed estesa piattaforma di analisi dei dati. Le interfacce standard permettono di inserire nel warehouse aziendale dati provenienti da on line transaction processing (OLTP) come ad esempio marketplaces, e dati provenienti da extraction, transformation and load process (ETL) come dati estrapolati da applicazioni proprietarie quali gli ERP. I dati ottenuti da queste estrazioni vengono poi utilizzati nelle regole di business per estrapolare trend aziendali come prospettive di crescita, previsioni di vendite ecc…. Questa infrastruttura permette quindi di monitorare ed ottimizzare i processi, prevedere scenari di business e indicatori chiavi capaci di evidenziare situazioni che potrebbero rappresentare opportunità o pericoli allo sviluppo aziendale. La business intelligence integrata nel portale aziendale permetterà l'accesso immediato a queste informazioni e creerà maggiore collaborazione e condivisione degli obiettivi.

2.1.3.3 Knowledge Management (KM)

Le massicce quantità di infrastrutture informatiche presenti nelle aziende, producono giornalmente varie tipologie di documenti le quali necessitano di essere catalogate, integrate, aggregate, disseminate fra le varie divisioni aziendali. Gli utenti dell'EIP, grazie all'integrazioni di sistemi di Knowledge Management, potranno accedere on demand e potranno anche specificare la tipologia del documento di cui necessitano. Per poter far questo, i dati devono essere classificati in multiple tassonomie integrate con i motori di ricerca del portale, in questo modo l'utente può trovare i documenti anche non conoscendo esattamente il contenuto degli stessi. Il knowledge Management integrato nel portale permetterà accedere ai documenti, attraverso un'interfaccia, la quale rappresenterà un unico punto di accesso ai dati. La possibilità di implementare queste interfacce con l'uso dello standard web-based distributed Authoring and Versioning (WebDAV)[1] fornisce il front-end ideale per l'accesso a servizi dedicati alla gestione della conoscenza, questo grazie alla possibilità di individuare un linguaggio comune per la gestione del metadato allocato sul web. Tale tecnologia permette all'utilizzatore di accedere non solo a documenti interni all'organizzazione aziendale, ma anche di condividere informazione con sorgenti esterne, automaticamente individuate dai motori di ricerca.

1 Insieme di estensioni al protocollo HTTP per la gestione, l'aggiornamento e la modifica di documenti tramite Internet o Intranet. Consente il lavoro in workgroup sullo stesso documento, anche contemporaneamente, sullo stesso server Web, gestendo il blocco e lo sblocco dei file.
E' indicato, ad esempio, per la gestione di siti complessi da parte di gruppi di lavoro.

2.1.3.4 Accesso ai contenuti del web e gestione dei servizi

Una crescente parte delle informazioni aziendali proviene attraverso sorgenti allocate esternamente dei confini aziendali. Gli impiegati sono in questo modo già abituati ad accedere a vari siti esterni all'intranet aziendale al fine di ricercare informazioni ed utilizzare quei servizi capaci di complementare il set di quelli offerti dai sistemi aziendali. L'EIP deve quindi essere in grado di connettere l'utente al mondo internet e permettere l'uso dei servizi fruibili dal web, come infatti un sistema di knowledge management fornisce una gerarchizzazione dei dati anche non strutturati dell'impresa, i moderni portali internet come yahoo! forniscono un simile servizio di gerarchizzazioni delle informazioni presenti su tutto il web. Risulta importante quindi che un enterprise information portal possa permetterne l'utilizzo o ancora meglio integrare tali servizi nei propri frame, dando all'utente la possibilità di essere affiancato, nel proprio processo decisionale, anche da notizie provenienti esternamente dall'impresa .

2.1.4 Multidevice support:

Considerata l'importanza di utilizzo che L'EIP assumerà nelle imprese, sarà necessario che questo venga reso fruibile da un qualsiasi dispositivo connesso alla rete. Ecco perché attualmente i protocolli di comunicazione utilizzati per connettersi sono l'http per il Web ed il WAP per dispositivi mobili quali i cellulari, affiancati dai rispettivi linguaggi di markup per la gestione dei dati come il PSML ed il WML.

2.1.5 Portal Administration

Gl EIP devono integrare, fra le varie tecnologie, tool di sviluppo e tool per la gestione del portale e degli utenti capaci di snellire le procedure di amministrazione di un portale di elevate dimensioni. Anche se la piattaforma J2EE, sulla quale si appoggiano la maggior pare dei portali risulta altamente scalabile, la mancanza di tool di gestione potrebbe rendere il costo di backlog amministrativo del portale talmente elevato da ridurne le potenzialità. Basti pensare infatti alla gestione di tutti gli utenti, dei gruppi di lavoro e delle relative autorizzazioni di accesso per capire con quanta facilità si potrebbe appesantire la gestione del portale e quindi ridurne la scalabilità. Ci si troverebbe ad usare una tecnologia altamente scalabile ma non si riuscirebbe a sfruttare tale caratteristica a causa della difficoltà di gestire un portale di grosse dimensioni.

2.2 Portal Architecture

Come la maggior parte delle applicazioni distribuite, l'architettura di base nella quale opera un EIP, è quella definita dalla *Java 2 Enterprise Edition* (J2EE), ampiamente descritta nel capitolo successivo. L'obiettivo di questo paragrafo è scendere nel dettaglio della piattaforma J2EE e focalizzare l'analisi sull'architettura generalmente assunta da un qualsiasi Enterprise Information Portal.

Generalmente un EIP è una Web Application che necessita di un *Servlet Engine* per processare le pagine dinamiche e di un *Server Web* per poter reindirizzare le richieste http. L'architettura di base che si è potuto estrapolare dai portali analizzati, e dallo schema realizzato da C. Wedge e pubblicato su "Portal Server Tecnology" IEEE Internet Computing May/June 2002 risulta rappresentata nella Figura sotto.

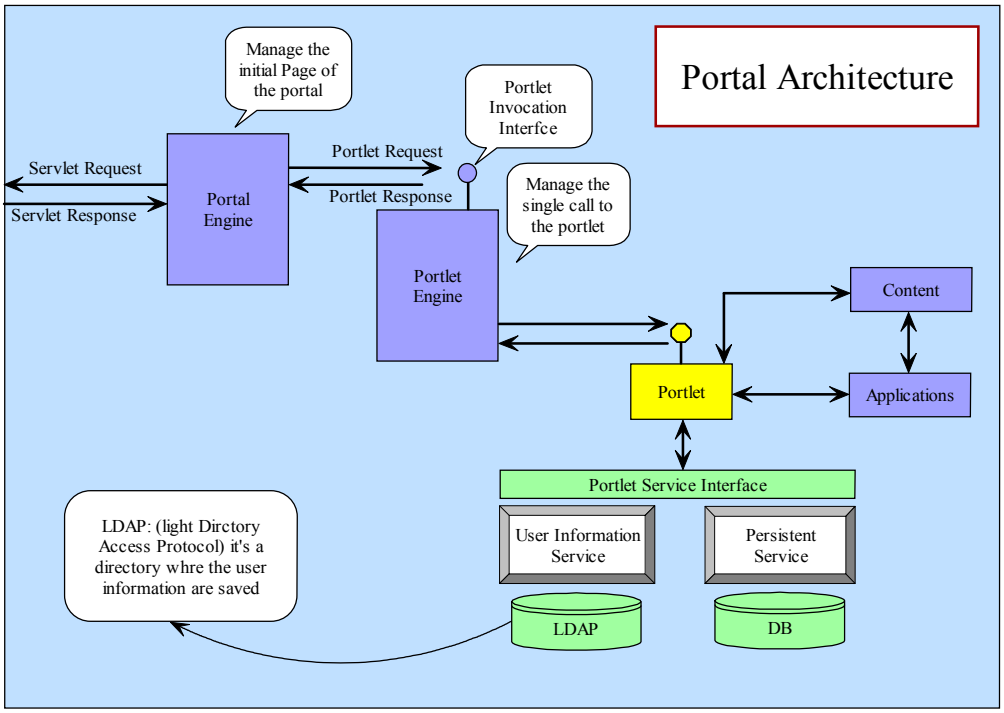


Figura 5: -General Portal Architecture

2.2.1 Portlet Application

Uno dei componenti fondamentali che costituiscono il portale sono delle applicazioni chiamate portlet. Da un punto di vista dell'utente, un portlet è un' area delimitata da una piccola finestra, contenente dati specializzati posizionato all'interno di un portale. Per esempio un portlet può contenere itinerari di viaggio, notizie di business, informazioni meteorologiche o andamenti borsistici. Gli utenti possono personalizzare il contenuto, il layout e la posizione del portlet secondo le loro preferenze e secondo la disponibilità lasciatagli dall'amministratore del portale.

Da un punto di vista informatico, i portlets sono componenti, simili alle servlet, progettati per essere aggregati nel contesto di una pagina web. Solitamente più portali vengono richiesti contemporaneamente da una singola richiesta eseguita da una pagina iniziale. Ogni portlet produce un fragmento di codice markup che viene combinato assieme al markup degli altri portlets. Le Portlet specification, consultabili in rete al sito ufficiale della Sun: Java Specification Request Comunity [JSRs] elencano le caratteristiche dei cicli di vita, delle semantiche e delle interfacce da utilizzare per programmare con questi componenti. I portlets girano all'interno del portlet container ossia di un servlet engine come Tomcat, nel quale trovano l'ambiente dove istanziarsi, essere usati ed infine venir distrutti. In tale infrastruttura i portlets possono accedere ad informazioni private dell'utente, prendere parte ad eventi, azioni, comunicare con altri portlets ed accedere a contenuti remoti o dati persistenti. I portlets risultano essere dinamicamente meglio amministrabili rispetto alle servlet. Per esempio, a differenza delle servlet, una applicazione portlet consiste di più portlets che possono essere installati o rimossi mentre il portal server è attivo. Anche le proprietà e i diritti di accesso di un portlets possono essere cambiati dall'amministratore mentre il portal server è attivo. Al portlet è permesso di apparire in diverse interfacce chiamati *modi* e la loro invocazione è resa possibile attraverso l'uso delle icone poste nella parte alta a destra della barra del titolo.

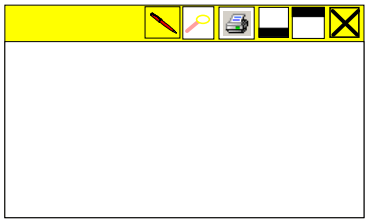


Figura 6: - Esempio di Portlet –

Lo scopo dei portlets risulta essere quello di:

- Avere piccole e multiple applicazioni web.
- Permettere agli utenti la gestione degli "skin" cioè dei colori di sfondo, titoli e colori delle barre delle icone cioè di manipolare a piacimento la presentazione del dato.
- Mantenere performanti le prestazioni di svariati portlets con l'uso di sottosistemi di cahing.
- Tenere traccia di tutte le applicazioni web e presentarle su richiesta all'utente.
- Facilitare la personalizzazione dei servizi da caricare nella Home page.
- Permettere lo sviluppo di questi applicativi pur non conoscendo in modo preciso i dettagli implementativi del motore Jetspeed nel quale agiscono.
- Gestire gli skin singolarmente.
- Essere gestiti da un PortletController il quale ha una implementazione specifica che può essere modificata dallo sviluppatore per fornire una migliore personalizzazione.

2.2.2 Flusso di informazioni nell'EIP

Una volta che il Server ha ricevuto una richiesta http da parte di un client ed ha verificato come questa debba essere processata, viene inoltrata al Portal Engine, che rappresenta una estensione del Server, una servlet request. Il *Portal Engin*e prepara la struttura della pagina del portale ed invia al *Portlet engine* una richiesta che contiene, a sua volta, la richiesta di n portlets da caricare nella pagina iniziale. Compito del Portlet engine è di richiamare i singoli portlet attraverso l'uso del loro URL. Il *portlet* legge i dati da una sorgente persistente, li processa, ed invia il risultato al portlet engine, mantenendo una copia dei contenuti creati in cache. A questo punto è compito del portlet engine creare uno script da inviare al portal engine che lo inserirà fra gli script della pagina iniziale; tale script permetterà l'inclusione dei contenuti del portlet nel contesto della pagina.

Capitolo 3

La Piattaforma J2EE

Considerate le potenzialità del World Wide Web ed Internet risulta normale che queste rappresentino le fondamenta sulle quali le imprese tendono a lavorare sempre più intensamente, cercando di integrare le loro informazioni in *applicazioni su misura distribuita (distribuited custom applications).* E' compito degli sviluppatori di queste applicazioni capire quali sono le specifiche esigenze dei possibili utenti delle informazioni necessarie e fornire dunque funzionalità adeguate. E' altresì fondamentale nello scenario altamente competitivo dell'information economy avere un buon response time, cioè progettare applicazioni in grado di potersi adattare alle esigenze delle imprese nel modo più semplice possibile a fronte di eventuali interventi di aggiornamento.

L'obiettivo della piattaforma *Java 2 Enterprise Edition* [J2EE], distribuita dalla Sun Microsystem [SUN], è di definire funzionalità standard che aiutino a sviluppare queste applicazioni distribuite, utilizzando un vasto range di nuove tecnologie e un modello di progettazione component-based.

3.1 Vantaggi della piattaforma J2EE

La piattaforma J2EE offre una serie di benefici agli sviluppatori di applicazioni distribuite.

3.1.1 Architettura e sviluppo semplificati

La J2EE supporta un modello di sviluppo component-based semplificato. Essendo basata sul linguaggio di programmazione Java e sulla Java 2 Standard Edition (J2SE platform), questo modello offre grandi vantaggi di portabilità (sistema Write Once, Run Anyway) e garantisce il supporto su tutti i server conformi a questo standard.

Il modello di sviluppo component-based J2EE arricchisce la produttività delle applicazioni in diversi modi:

1) assicura flessibilità alle funzionalità desiderate, consentendo diverse possibilità di configurazione dell'architettura della applicazione, a seconda di vari fattori, come ad esempio il tipo di client, la sicurezza richiesta per gli accessi alle sorgenti di dati, e altri che verranno approfonditi in seguito. Il

modello component-based inoltre semplifica la manutenzione dell'applicazione, in quanto i singoli componenti possono essere aggiornati o sostituiti indipendentemente.

2) assicura ai componenti la disponibilità di una serie di servizi nell'ambiente di run-time, e la possibilità di essere dinamicamente connessi ad altri componenti, semplicemente fornendo opportune interfacce. Ad esempio attraverso i deployment descriptors[2] è possibile comunicare specifici parametri all'ambiente di run-time personalizzando l'applicazione senza dover modificare il codice dei componenti.
3) supporta la suddivisione dei compiti, in quanto ciascun set di componenti può essere associato ad un certo ambito di sviluppo, consentendo a ciascuna figura professionale di concentrarsi specificatamente sulle proprie competenze e abilità. Questa suddivisione dei compiti, oltre a favorire la specializzazione e quindi migliorare la qualità dei singoli componenti, consente un criterio di sviluppo dell'applicazione in parallelo, aumentandone la velocità di produzione e manutenzione.

3.1.2 Scalabilità

La J2EE fornisce supporti per adattare in modo semplice e funzionale una applicazione ad eventuali aumenti o riduzioni delle dimensioni del progetto senza comprometterne l'efficienza delle prestazioni. Ad esempio il supporto per il *connection pooling*[3] assicura rapido accesso ai dati anche ad un numero crescente di clients.

Inoltre l'architettura distribuita che la J2EE supporta consente di ottenere un vantaggioso sistema di bilanciamento del carico di lavoro, permettendo la configurazione dei vari livelli dell'architettura per la gestione su server diversi.

3.1.3 Integrazione su sorgenti di informazioni già esistenti

La J2EE platform, assieme alla J2SE (Java 2 Standard Edition) platform, include una serie di API (Application Program Interface) standard per accedere alle varie sorgenti di informazione esistenti nell'impresa.

Si riporta l'elenco di queste API.

- JDBC (Java DataBase Connectivity) è l'API per la connessione ai database relazionali. Essa fornisce un interfaccia a livello di applicazione usata nel codice dei componenti per accedere al database in modo indipendente dal particolare DBMS utilizzato; sarà poi necessario

[2] Il deployment descriptor è un file di testo in formato XML che specifica il comportamento di un componente attraverso tag XML.
[3] Il pool di connessioni è un meccanismo molto utile per diminuire il gravoso tempo di accesso ai database.

utilizzare un driver specifico che si ponga come interfaccia tra l'API JDBC e lo specifico DBMS utilizzato. Grazie a questo meccanismo a due livelli è possibile mantenere il codice dei componenti dell'applicazione indipendente dal DBMS che può essere sostituito con un altro semplicemente cambiando il driver, e senza necessità di riscrittura del codice.

- JTA (Java Transaction API) è l'API per la gestione e il coordinamento delle transazioni attraverso sorgenti di informazioni transazionali eterogenee.
- JNDI (Java Naming and Directory Interface) è l'API per accedere ad informazioni attraverso un sistema di nomi e percorsi, utilizzando così lo stesso meccanismo utilizzato per riferire i files nel File System.
- JMS (Java Message Service) è l'API per inviare e ricevere messaggi attraverso sistemi di enterprise-messaging come l'IBM MQ Series e TIBCO Rendez-vous che richiedono l'attivazione del servizio attraverso un JMS provider.
- *JavaMail* è l'API utilizzata dai componenti per inviare e ricevere email.
- *Java IDL* è l'API per richiamare oggetti CORBA remoti, attraverso il protocollo IIOP. Questi oggetti sono tipicamente esterni alla J2EE e possono essere scritti in qualunque linguaggio.

Oltre a questi servizi che già sono presenti nell'attuale versione della J2EE platform, ve ne sono altri in via di sviluppo che saranno aggiunti nelle successive versioni attraverso l'architettura dei *Connectors*.

3.1.4 Scelta di servizi, tools e componenti

Il marchio J2EE è punto centrale per creare un mercato di servizi, tools e componenti tutti conformi allo stesso standard.

Le organizzazioni che sviluppano applicazioni J2EE possono contare su una varietà sempre crescente di fornitori di piattaforme J2EE con diverse possibilità di scelta sia a livello hardware, sia a livello di sistemi operativi e configurazioni di server, a seconda degli obiettivi e strategie adottate.

I vari componenti J2EE, con particolare riferimento agli EJB e alle JSP di cui si tratterà dettagliatamente in seguito, sono stati ideati per essere facilmente manipolati da tool grafici che consentono di automatizzare tanti dei tradizionali compiti richiesti, come ad esempio il debugging. Gli sviluppatori J2EE hanno così la possibilità di scegliere il tool conforme allo standard J2EE che meglio si addice alle loro esigenze.

Grazie al modello component-based si ha poi la possibilità di sviluppare componenti conformi allo standard che possono essere così riutilizzati da una qualsiasi applicazione J2EE.

3.1.5 Modello di sicurezza semplificato e unificato

Gli sviluppatori possono specificare i requisiti di sicurezza di un componente. Attraverso un sistema dichiarativo, sia gli EJB-component sia i Web-component possono specificare attraverso i deployment descriptor i criteri per associare i vari livelli di accesso a diversi ruoli e quindi alle diverse categorie di utenti. Per gli EJB esiste addirittura la possibilità di definire ruoli diversi per ciascun metodo.

Le specifiche J2EE incoraggiano l'utilizzo di questo sistema dichiarativo per consentire al codice dei componenti di rimanere totalmente indipendente dai vincoli legati alla sicurezza. Ciò non sempre è possibile, poiché in taluni casi è necessario creare livelli e vincoli di sicurezza che non possono essere espressi con un semplice mapping tra ruoli e utenti. Perciò viene comunque mantenuta la possibilità di inserire security-checks anche all'interno del codice attraverso opportune API.

3.2 Architettura multilivello della J2EE

Si analizza ora nel dettaglio il modello di applicazione distribuita multilivello della J2EE platform.

Si è già accennato alla caratteristica component-based del modello J2EE con i vantaggi che ne derivano. La logica di ogni applicazione J2EE può essere infatti suddivisa in più componenti in accordo con la funzione da essi svolta, ciascuno dei quali può essere installato in una macchina differente a seconda del livello logico di appartenenza all'interno dell'architettura J2EE.

La architettura multilivello J2EE individua tre livelli fondamentali: il *client-tier*, il *middle-tier* e l'*EIS-tier.*

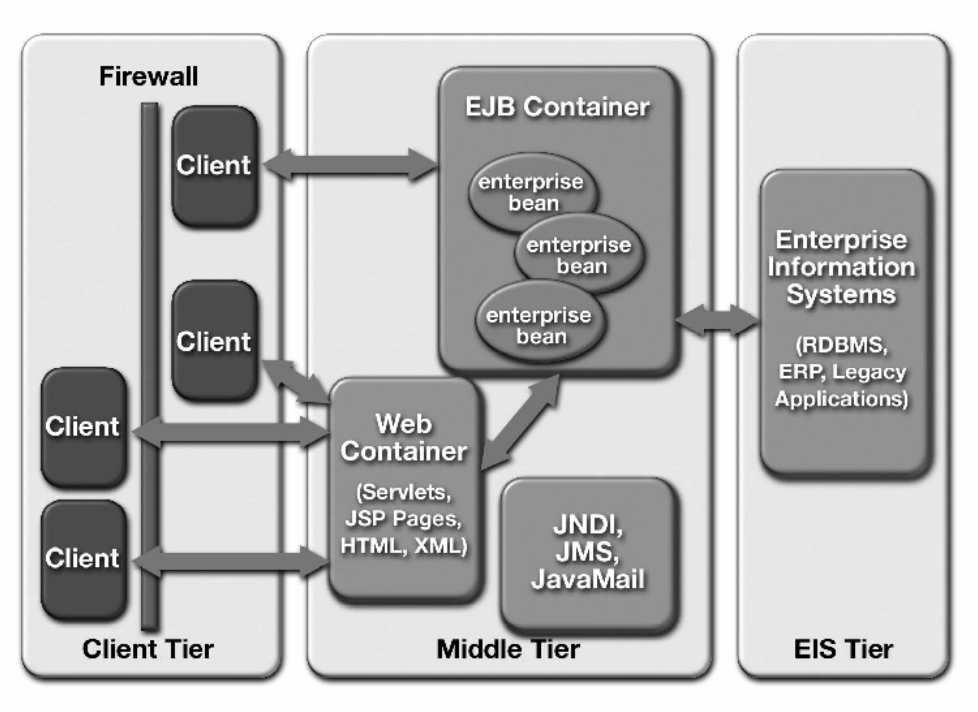


Figura 7: - Architettura multilivello J2EE –

La figura mostra i vari componenti e sevizi che costituiscono un tipico ambiente J2EE. La presenza di un livello intermedio tra il client e le sorgenti di informazione fornisce alla J2EE platform i tipici vantaggi derivanti da un'architettura multilivello.

Le tipiche architetture client-server a due livelli, nelle quali il server è in pratica costituito dalle sorgenti di informazione, hanno dato prova di essere fortemente limitanti a causa della necessità di installare e mantenere un front-end completo di interfaccia grafica e di logica applicativa direttamente su ciascun client. Ciò naturalmente si traduce in un forte carico di lavoro sulla macchina client, creando problemi derivati dalla limitatezza delle risorse di tali macchine, tipicamente dei semplici PC, e nel non meno grave problema della manutenzione dell'applicazione: ogni più piccolo cambiamento nel codice dell'applicazione deve essere ripetuto su tutti i client.

Con un'architettura a tre livelli questi problemi vengono superati grazie all'introduzione di un middle tier che si pone tra client e la sorgente di informazioni. In questo scenario il server può ospitare sia il middle tier che l'EIS tier, oppure questi ultimi possono essere ospitati su server differenti, dando origine al middle tier server e all'EIS server.

Il middle tier server si occupa di gestire le richieste dei vari client, di reperire le opportune informazioni dall'EIS server, rielaborarle secondo la logica applicativa, e fornirle al client, che si trova così notevolmente alleggerito.

La J2EE fornisce uno standard ben preciso per implementare il middle tier server, e fornisce indicazioni e raccomandazioni per implementare gli altri due livelli.

Si rimanda ai prossimi paragrafi la descrizione dettagliata dei tre livelli della J2EE, mentre in questa sede si vuole introdurre un altro elemento chiave dell'architettura: i *containers*.

I containers sono degli ambienti run-time standardizzati in grado di fornire specifici servizi ai vari componenti. Ciascun componente può contare sul supporto del container in cui è inserito per ottenere gli stessi servizi standard in tutte le piattaforme J2EE qualunque sia lo specifico provider. I vantaggi che ne derivano sono i seguenti:

- gli sviluppatori possono disinteressarsi di come i container gestiscono tali servizi, potendo così concentrarsi sulla funzionalità del codice.
- attraverso i deployment descriptor (DD), che fungono da interfaccia tra i componenti e il container si possono specificare i parametri e i comportamenti dei componenti stessi al momento del deployment dell'applicazione senza dover interferire sul codice.

3.3 EIS (Enterprise Information System)

E' il livello più basso dell'architettura, cioè la parte di dati memorizzati in modo permanente con varie tipologie di risorse, tipicamente RDBMS. Questi ultimi possono supportare accessi controllati attraverso le transazioni. Essi possono partecipare in transazioni assieme ad altre risorse transazionali in un sistema di database distribuito attraverso il protocollo *two-phase commit*, gestito da un transaction manager supportato dal J2EE server.

Questa integrazione di sorgenti di informazione funziona bene quando queste sono omogenee. Purtroppo non sempre è così quando si sviluppa una applicazione distribuita che si deve appoggiare a varie sorgenti.

E' attualmente in via di sviluppo da parte dei progettisti della Sun un nuovo tipo di API, le *API J2EE Connector*, allo scopo di definire uno standard per connettere la J2EE platform con sorgenti di informazioni eterogenee.

3.4 Middle Tier

I maggiori benefici del modello di applicazioni J2EE si riscontrano nel middle tier. Esso è ulteriormente scomposto in due sottolivelli: un *Web-tier* e un *EJB-tier* che possono trovarsi anche su differenti hosts, oppure sullo stesso host ma in differenti Java Virtual Machines, o infine anche sulla stessa JVM.

Le funzioni di business logic della applicazione vengono implementate attraverso componenti *Enterprise Java Beans (EJB)* all'interno dell'EJB-tier.

Al Web-tier spetta invece il compito di "presentare" al client (in questo caso un Web-client) i risultati dell'elaborazione della business logic, attraverso la generazione di pagine web dinamiche.

3.4.1 EJB tier

La tecnologia degli *Enterprise Java Beans (EJB)* [SPECEJB] offre un modello component-based distribuito che consente agli sviluppatori di concentrare la loro abilità sui business problem dell'applicazione, lasciando al *EJB container* il compito di gestire tutta una serie di servizi di sistema a basso livello. Questa separazione di ruoli, punto di forza dell'architettura J2EE che si ritrova anche nel Web tier, permette un più rapido sviluppo di applicazioni altamente scalabili, flessibili e sicure.

Gli EJB sono componenti che costituiscono un collegamento fondamentale tra i componenti del Web tier ai quali viene affidato esclusivamente il compito di gestire la presentation logic, e le risorse di informazione mantenute nel EIS tier.

La business logic e i business-objects

E' difficile dare una definizione rigorosa di business logic di una applicazione. In senso lato può essere definita come un insieme di direttive per gestire ogni specifica funzione che l'applicazione deve svolgere.

Volendo adottare un approccio object-oriented, una funzione può essere scomposta in un insieme di componenti, o elementi chiamati *business-objects*. Come gli altri elementi, anche questi sono dotati di una specifica struttura dati e di un determinato comportamento. Ad esempio un impiegato può essere modellato con un oggetto-impiegato che ha una struttura dati, rappresentata da un insieme di attributi quali il nome, l'indirizzo, il codice fiscale e così via; inoltre ha dei metodi ad es. per assegnarlo ad un nuovo dipartimento, aumentargli lo stipendio, ecc.

In modo più rigoroso si può allora definire la business-logic come l'insieme delle regole che servono ad identificare struttura e comportamento dei business-objects, e i criteri di interazione con gli altri oggetti dell'applicazione.

Requisiti comuni dei business-object

Si riportano qui di seguito alcuni requisiti comuni ai business object.

a) *Mantenere lo stato*. I business-objects solitamente necessitano di mantenere, in modo conversazionale o permanente, lo stato rappresentato dalle variabili di istanza tra le invocazioni dei vari metodi.
b) *Operare su dati condivisi*. La condivisione di certe risorse deve essere gestita attraverso il controllo della concorrenza e un appropriato livello di isolation dei dati condivisi, in modo da assicurarne la consistenza.
c) *Partecipare alle transazioni*. L'atomicità di una transazione deve essere garantita anche se questa si distribuisce su diverse risorse remote.
d) *Servire un gran numero di client*. Un business-object deve essere disponibile in istanze multiple a diversi client contemporaneamente. Il meccanismo di gestione di queste istanze deve essere in grado di fornire a ciascun client un

business-object disponibile a servire la sua richiesta. Senza un tale meccanismo il sistema potrebbe eventualmente esaurire le risorse e non essere più in grado di servire ulteriori richieste.

e) *Fornire accesso remoto ai dati*. I client devono poter accedere alle risorse dei business-objects anche da remoto attraverso la rete.
f) *Controllare gli accessi*. I servizi offerti dai business-objects spesso richiedono l'autenticazione del client che ne fa richiesta, per consentire l'accesso a risorse protette ai soli client autorizzati.
g) *Garantire la riusabilità*. E' questo un requisito comune a tutti i tipi di oggetti in un approccio object-oriented.

Gli Enterprise Java Beans e l'EJB Container

Nell'architettura J2EE i business-objects vengono modellati con componenti Enterprise Java Beans. Come già accennato essi possono contare sul supporto fornito dall'EJB container che ne gestisce il ciclo di vita e fornisce loro una varietà di servizi. Quando un client invoca un'operazione su un EJB questa viene prima intercettata dal container che può in tal modo gestire i vari servizi che devono eventualmente propagarsi tra diverse sottochiamate ad altri componenti dell'EJB tier.

Grazie all'intercessione del container è anche possibile definire determinati comportamenti del componente al momento del deployment a seconda delle esigenze senza dover effettuare alcun cambiamento nel codice e ricompilazione, ma semplicemente configurandoli attraverso il deployment descriptor.

Il Bean Provider, cioè la figura professionale che si occupa di sviluppare questi componenti, deve fornire anche una "client view" del componente stesso attraverso la definizione di due interfacce: la *home interface* e la *remote interface*. Questo assicura che il client abbia una visione semplificata del componente e che possa disinteressarsi dei dettagli implementativi. Sarà compito del container implementare queste interfacce, crearne le istanze e gestirne il ciclo di vita.

La *home interface* fornisce i metodi per creare e rimuovere gli EJB. Essa deve estendere l'interfaccia `javax.ejb.EJBHome` . Attraverso la home interface il client può inoltre ottenere informazioni sui meta-dati dell'EJB . Per particolari tipi di EJB, di cui si parlerà in seguito, la home interface consente anche di ottenere un handle, cioè un riferimento, alla interfaccia stessa che può essere serializzato e salvato in modo permanente, e infine consente di recuperare istanze particolari del bean salvate in precedenza.

La *remote interface* definisce la vera e propria "client view" dell'EJB, cioè il set di business methods disponibili ai clients. Essa deve estendere l'interfaccia *javax.EJB.EJBObject* .

Per completare lo sviluppo di un EJB occorre fornire anche l'implementazione attuale dei business methods del bean. Ciò viene fatto fornendo la *Enterprise Bean class*che deve contenere il codice di tu tti i metodi dichiarati nella remote interface e inoltre deve contenere un metodo *ejbCreate* per ciascun metodo create della home interface. Quando il client invoca un metodo della remote interface, il container lo intercetta e richiama il corrispondente metodo della classe del bean.

L'Enterprise Bean class deve estendere la interfaccia *javax.ejb.EntityBean* oppure *javax.ejb.SessionBean* , a seconda del tipo di bean che implementano.

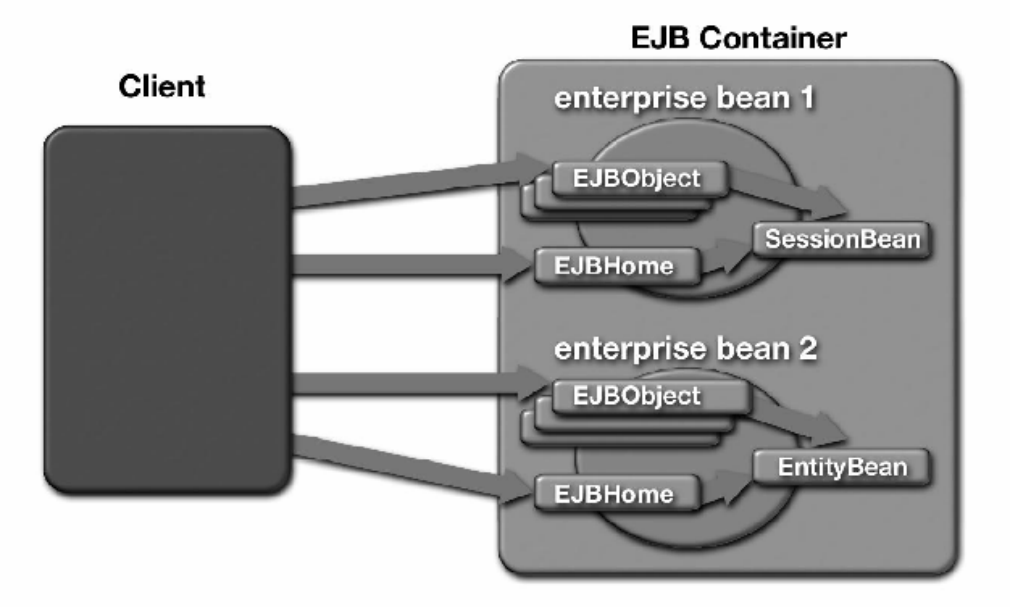


Figura 8: - Implementazione della client view di un EJB –

Session Beans

Come suggerisce il nome, un *session bean* è simile ad una sessione di interazione con un particolare client. Infatti un session bean rappresenta un unico client all'interno del J2EE server, e non è condiviso da nessun altro client. Esso non è persistente, cioè quando termina la sessione del client, ogni suo riferimento è perso e le risorse associate ad esso sono rilasciate.

Esistono due categorie di session bean: lo *stateful session bean* e lo *stateless session bean*.

Lo *stateful session bean* contiene lo stato conversazionale del client, cioè lo stato viene mantenuto per tutta la durata della sessione. Per stato conversazionale si intende non solo i valori delle variabili di istanza, ma anche gli oggetti raggiungibili attraverso tali variabili.

Lo *stateless session bean* non mantiene lo stato conversazionale del client. Quando un client invoca un metodo di un stateless session bean, le sue variabili di istanza possono contenere uno stato, ma solo per la durata dell'invocazione. Tranne che durante l'invocazione di un metodo tutte le istanze del bean sono equivalenti, lasciando al container il compito di assegnare un'istanza ad ogni client. La home interface di questo tipo di bean può contenere solo un metodo create senza argomenti.

Entity Beans

Un entity bean differisce da un session bean per i seguenti aspetti:

- è persistente, nel senso che viene mantenuto lo stato su un database anche al di fuori del ciclo di vita della applicazione e dei processi del J2EE middle server. Ci sono due possibili tipi di persistenza che possono essere dichiarati nel deployment descriptor: *bean-menaged-persistence (BMP)* e *container-menaged-persistence (CMP)*. La prima impone allo sviluppatore di prevedere nel codice tutte le chiamate al database con opportuni statement SQL necessari al container per gestire il ciclo vita del bean; ad esempio il metodo ejbCreate richiamato dal container invierà lo statement SQL di insert opportunamente fornito dallo sviluppatore. La seconda consente invece allo sviluppatore di non curarsi delle chiamate al database, delle quali si occupa automaticamente il container senza necessità di includere alcun statement SQL nel codice.
- consente accessi concorrenti da parte di più client. A tal scopo è importante che si utilizzino entity beans sempre all'interno di transazioni, che possono essere gestite automaticamente dal sistema.
- ha un identificatore unico (primary key).

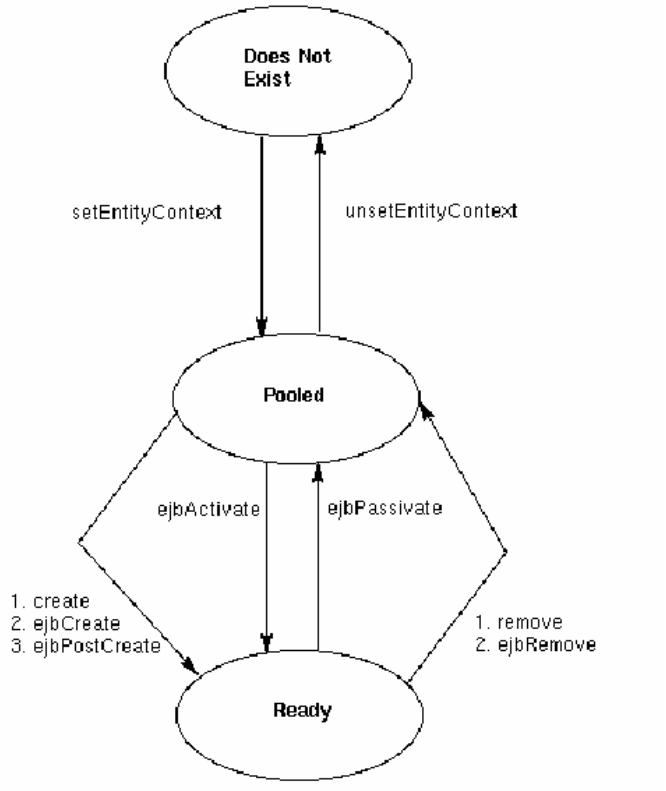


Figura 9: - Ciclo di vita di un entity bean –

La figura sopra riporta il ciclo di vita di un entità bean. Nel momento della partenza del server, tutti gli entity bean dei quali è stato fatto il deployment vengono caricati in memoria in un pool di entity bean, che si trovano dunque in uno stato "pooled". In questo stato tutte le variabili assumono il loro valore di default ed il

container assegna l'entity context col metodo setEntityContext. A questo punto ci sono due modi possibili in cui il bean può passare dallo stato "pooled" a quello "pronto". Il primo si ha quando il bean viene creato dal client con l'invocazione del metodo create: in questo caso in realtà non viene creato, ma viene prelevato un bean disponibile dal pool ed assegnato al client; successivamente il container richiama il metodo ejbCreate corrispondente e il metodo ejbPostCreate per terminare la fase di inizializzazione. Il secondo modo si ha quando il container invoca il metodo ejbActivate per recuperare un bean che era stato deallocato dalla memoria primaria. Esistono anche due percorsi che portano dallo stato "pronto" allo stato "pooled" e sono per invocazione del metodo ejbPassivate da parte del container o del metodo remove da parte del client.

Alla fine del ciclo di vita di un entity bean il container richiama il metodo unsetEntityContext che rimuove il bean dal pool.

Come si è detto ogni entity bean ha un suo identificatore unico, che ne costituisce la primary key. Questa può essere implementata da una qualsiasi classe Java serializzabile. Tipicamente questa classe contiene semplicemente delle variabili di istanza corrispondenti alle variabili di istanza della primary key della tabella del database associata. Il client può utilizzare il metodo FindByPrimaryKey, che deve essere implementato nella classe del bean, passando come argomento una particolare istanza della classe primary key per ottenere l'entity bean specifico corrispondente. Il client può avere a disposizione anche altri metodi cosiddetti finder per recuperare una particolare entity bean passando altri valori univoci.

Il legame fra un entity bean e i dati che rappresenta sul database è talmente stretto che una modifica nel primo si riflette in un aggiornamento dei dati. Se il database da utilizzare per la memorizzazione dei dati è di tipo relazionale, però, si può creare un problema nel mappaggio dei dati con le variabili del bean. Infatti la logica a oggetti del linguaggio Java consente di realizzare strutture dati di complessità elevata rendendo difficile l'operazione di conversione nel formato accettato da un database relazionale. Il container ha a disposizione due metodi, ejbLoad e ejbStore, per sincronizzare i dati memorizzati nel database con quelli incapsulati nel bean. Tipicamente ogni invocazione di un metodo associato ad una transazione prevede prima l'invocazione di ejbLoad , poi l'esecuzione del metodo ed infine l'invocazione di ejbStore.

3.4.2 Web Tier

La maggior parte dei nuovi servizi offerti sul Web, sono stati realizzati grazie all'incremento delle potenzialità delle applicazioni Web-Based, che consentono di generare contesti dinamici modificabili a seconda delle proprie necessità.

Le Servlets e le Java Server Pages (JSP) sono tecnologie della J2EE che supportano la generazione di contesti dinamici.

Le applicazioni J2EE Web-Based che utilizzano queste tecnologie possono essere costruite in vari modi:

- quelle più semplici possono utilizzare di base pagine JSP e servlets oppure pagine JSP con componenti modulari,

- applicazioni più complesse che devono gestire transazioni usano pagine JSP e componenti modulari in congiunzione con Enterprise Java Beans.
- Quando si parla di Web application si intende una collezione di documenti HTML/XML, mentre per Web components si intendono servlets, pagine JSP e altre risorse archiviate in un formato conosciuto come Web ARchive (WAR). Tutto questo e' localizzato e viene eseguito in un ***Web container*** di un Web server centrale che fornisce vari servizi con contesti dinamici ed interattivi ad una varietà di clients browser-based. Il Web container si occupa di fornire componenti web attraverso un naming context e gestisce i loro cicli di vita. Alcuni Web server possono fornire servizi aggiuntivi come la sicurezza, la gestione degli accessi concorrenti, le transazioni e il passaggio dei dati nello storage secondario, inoltre possono lavorare con un EJB server per fornire altre utilità senza dover per forza essere localizzati nella stessa macchina di quest' ultimo.
- La possibilità di sviluppare contesti generati automaticamente, permette di variare i dati contenuti, senza dover modificare il codice dell'applicazione, e quindi di aggiungere nuovi prodotti e servizi in modo immediato. Proprio queste tecnologie hanno trasformato il Web, evolvendolo dai primi contesti statici, agli attuali siti dinamici, modificabili a seconda delle necessità dei singoli utenti.
- Le funzioni offerte dal Web Tier sono tipicamente quelle sotto elencate:
- ***Web-enables business logic***---Il Web tier gestisce le interazioni fra Web Clients e applicazioni di business logic
- ***Generate dynamic content***---I componenti del Web tier generano contenuti dinamicamente e gestiscono vari formati di dati come HTML, immagini, suoni e video.
- ***Presents data and collects input***---I componenti del Web Tier traducono le istruzioni http di GET e di POST in form specifiche comprensibili alle apllicazioni di business logic e vengono utilizzate per presentare i risultati come conteuti del Web.
- ***Control screen flow***---Altro compito del Web Tier è la gestione della logica che determina quali videate presentare ai singoli clients infatti il flusso di videate tende sempre più ad essere specifico per ogni utente.
- ***Maintains state***---Il Web Tier gestisce meccanismi semplici e flessibili per accumulare dati per le transazioni e per le interazioni anche oltre i cicli di vita delle sessioni degli utenti.
- ***Support multiple and future client types***---Con l'ausilio degli Extensible MIME è possibile descrivere i contenuti del Web , in questo modo il client è in grado di selezionare le tipologie dei dati che stà gestendo e che stà scaricando.

Tradizionali tecnologie del Web tier

La prima risposta all'esigenza di fornire contenuti dinamici è stata la tecnologia delle Common Gateway Interface (CGI) un' interfaccia che permette ai Web servers

di chiamare degli scripts per ottenere dei dati da (o mandare dei dati a) databases, documenti ed altri programmi e presentare quei dati agli utenti tramite web. Questa tecnologia ha però un gran numero di limitazioni :

1) Il codice di uno script CGI che ha accesso a delle risorse, come file system o database, deve essere specifico per quella piattaforma, cioè molte applicazioni CGI non potranno essere eseguite su un'altra piattaforma server, limitandone quindi l'utilizzo nel caso che la Web application sia di tipo distribuito.
2) Poiche' un nuovo processo deve essere creato ogni qual volta che uno script CGI e' invocato, gli scripts risultano essere risorse lente e poco scalabili (problema risolvibile con l'incremento lato hardware, ma anche piuttosto dispendioso).
3) Difficoltà nella manutenzione perchè gli scripts combinano la logica e il contesto nello stesso codice, richiedendo l' utilizzo di due tipologie di esperti per poterli mantenere.

Queste limitazioni hanno frenato lo sviluppo delle CGI.

Tecnologie del Web tier nella piattaforma J2EE

La piattaforma J2EE supporta due tecnologie (Servlets e JSP) che sono una valida alternativa e che risolvono i problemi del CGI.

Java Servlets

Le java Servlets sono classi java che estendono un Web server compatibile con la piattaforma J2EE e producono contenuti dinamici come risposta alle richieste dei client. Le servlets possono essere viste come delle applets che girano su server totalmente indipendenti dal tipo di Web server su cui debbano girare. Un'applicazione browser-based che richiama servlets non ha bisogno di supportare il linguaggio di programmazione Java, perchè il codice inviato da una servlet al client può essere HTML, XML o qualsiasi content type supportato dai browser.

Le servlets hanno performance migliori degli scripts CGI : possono essere caricate nella memoria una volta e poi richiamate tutte le volte che sia necessario, inoltre possono girare su un singolo thread mentre gli scripts CGI devono essere caricati su un processo differente per ogni richiesta. Altro beneficio é che le servlets possono mantenere o riunire connessioni ai databases.

Le servlets eliminano molta della complessità legata al fatto di prendere dei parametri da una richiesta http poiché con quest'ultime i componenti hanno accesso diretto ai parametri perché sono presentati come oggetti, mentre nelle applicazioni CGI i parametri forniti da un form sono convertiti in proprietà dell' ambiente e sono disponibili all'interno del programma. Uno dei più grandi benefici é che le servlets forniscono delle API per mantenere i dati in sessione attraverso un'applicazione web e per interagire con le richieste dell' utente. I dati in sessione possono essere usati per scavalcare le limitazioni delle Web applications a causa della non capacità di

mantenere gli stati dell' http. In riferimento alla gestione dei dati su richieste multiple, occorre sottolineare il fatto che esistono quattro ambiti (scope) di esistenza, condivisione e visibilità delle sorgenti di informazioni di una determinata applicazione Web. Ad ognuno di questi scope è associato un particolare oggetto detto appunto Scope Object, e le informazioni ad esso associate vengono mantenute come attributi di tali oggetti.

I quattro scope object con la relativa visibilità sono i seguenti:

1) *web context* : rappresenta l'ambito con maggiore visibilità; le informazioni memorizzate come attributi sono visibili a tutti i web component dell'applicazione
2) *session* : rappresenta un ambito associato ad un utente; e informazioni memorizzate come suoi attributi sono visibili a tutti i web component che servono le request di uno stesso client. E' questo il classico esempio del carrello della spesa delle applicazioni e-commerce
3) *request* : rappresenta l'ambito associato ad una stessa request
4) *page* : rappresenta l'ambito associato ad una stessa JSP

Essendo scritte in Java, le servlet possono essere utilizzate su qualsiasi piattaforma che abbia una Java Virtual Machine ed un Web server che le supporti e quindi possono essere usate su differenti piattaforme senza dover essere ricompilate.

Le servlet hanno un ciclo di vita che viene gestito dal web-container. Quando una request è mappata ad una servlet, il container esegue i seguenti passi:

1) carica la classe Java della servlet;
2) ne crea una istanza;
3) inizializza l'istanza invocando il metodo `init`; questo metodo viene richiamato una sola volta all'atto del caricamento della servlet stessa e può essere ridefinito per personalizzare la configurazione iniziale della istanza. Viene inoltre assicurata la sincronizzazione automatica dell'esecuzione di questo metodo in modo da consentirne l'accesso ad un thread per volta;
4) invoca il metodo service passando come parametri un oggetto `HttpRequest` e un oggetto `HttpResponse`.

Se il container necessita di rimuovere una servlet, ne richiama il metodo `destroy`.

Volendo entrare un po' più nei dettagli si riporta qui di seguito un breve esempio di una servlet.

```
Import java.util.*;
Import java.text.*;

public class SimpleServlet extends HttpServlet {


/* ridefinizione del metodo doGet */

public void doGet (HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {

PrintWriter out;

// istanziamento degli oggetti per data e ora
String dateformat = "EEEE d MMMM yyyy";
String timeformat = "H:mm";
DateFormat df = new SimpleDateFormat(dateformat);
DateFormat tf = new SimpleDateFormat(timeformat);
Date datetime = new Date();

// prima occorre settare il content type
response.setContentType("text/html");

// poi costruire la response
out = response.getWriter();
out.println("<HTML><HEAD><TITLE>");
out.println("DATA E ORA");
out.println("</TITLE></HEAD><BODY>");
out.println("<H1>DATA E EORA ATTUALI</H1>");
out.println(df.format(datetime));
out.println("<BR>");
out.println(tf.format(datetime));
out.println("</BODY></HTML>");
out.close();
}
}
```

Figura 10: Codice di esempio – Servlet che genera una pagina dinamica con dat e ora –

Questo è un banalissimo esempio di una servlet che genera una pagina web che contiene la data e l'ora attuali nel momento in cui la pagina viene spedita al client.

Si analizzano ora brevemente le parti più significative del codice.

Una classe Java per essere una servlet deve estendere la classe astratta `HttpServlet` e deve ridefinire almeno un metodo di questa classe tra i seguenti: `doGet`, `doPost`, `doDelete`, `doPut` a seconda del tipo di request http da gestire. In particolare il metodo `doGet` gestisce le request http di tipo GET, cioè quelle che includono eventuali parametri attaccandoli in coda all'URL consentendo quindi un limitato invio di dati, mentre il metodo `doPost` gestisce le request http di tipo POST che includono eventuali parametri nella request stessa, consentendo invio di un quantitativo di dati molto più ampio. A tutti questi metodi devono essere passati come argomenti la `HttpRequest` ricevuta dal client e la `HttpResponse` da settare ed inviare al client.

Nell'esempio il metodo utilizzato è il `doGet` e la request non contiene parametri. Inoltre si è fatto uso di istanze di classi apposite per ottenere la data attuale opportunamente formattata.

Per costruire la response occorre prima settare i campi dell'intestazione (*header fields*) : tra questi in particolare il campo content-type è quello che definisce il tipo di

contenuto del body della response. Nell'esempio il metodo `setContentType` dell'interfaccia `HttpResponse` viene utilizzato per settare il content-type al valore "text/html" che indica che il body della response conterrà del testo in formato HTML, cioè la sorgente di una pagina web che può essere interpretata da un browser.

Per costruire il body della response occorre innanzitutto ottenere un output stream da usare per inviare dati al client. Dal momento che, come si è detto, il contenuto della response dovrà essere interpretato dal browser del client come testo HTML, nell'esempio si è utilizzato il metodo `getWriter` per ottenere un oggetto `PrintWriter` che consente di inviare uno stream di caratteri. Se si fosse utilizzato il metodo `getOutputStream` si sarebbe ottenuto uno oggetto `OutputStream` associato alla response sul quale sarebbe stato possibile inviare uno generico stream di bytes in forma binaria.

Come si nota le servlet utilizzano degli statement Java di stampa per l'invio di HTML al client. La pagina viene costruita dal codice dell'applicazione che scrive nello stream di output il codice HTML carattere per carattere.

Il risultato dell'elaborazione di questa servlet è la seguente pagina HTML:

```
<HTML>
 <HEAD>
        <TITLE>DATA E ORA</TITLE>
</HEAD>
 <BODY>
        <H1>DATA E ORA ATTUALI</H1>
        lunedi' 4 febbraio 2002<BR>
            15:58
 </BODY>
</HTML>
```

Figura 11: Codice di esempio – HTML prodotto dalla servlet precedente –

Java Server Pages (JSP)

La tecnologia delle Java Server Pages oltre a tutti i benefici delle servlets offre la possibilità di separare il contesto dalla logica mentre le servlets sono espresse in linguaggio Java, le pagine JSP sono documenti di testo che includono una combinazione di HTML, XML, speciali tags JSP, codice Java e altri linguaggi (Javascript, CSS).

Sebbene entrambi possano essere utilizzate per gli stessi obiettivi, le servlets vengono impiegate come meccanismo per accettare richieste da un browser, recuperare dati dai databases, eseguire la parte di logica dell' applicazione (specialmente se queste accedono al database direttamente) e dare una formattazione a quei dati per la loro presentazione sul browser (di solito HTML). Per inviare i dati utilizzano dei metodi di scrittura con dentro il codice da spedire al client, ma questo tipo di approccio puo' causare due problemi:

1) essendo il codice HTML embedded nei metodi Java il Web Designer non può vedere un'anteprima della pagina per correggere eventuali errori;
2) quando i dati da mostrare cambiano, non e' proprio semplice ricercare la parte di codice da modificare (inoltre essendo la parte di logica ed il

contesto intrecciati anche se si modifica qualcosa di quest' ultimo la servlet dovrà essere ricompilata e ricaricata dentro al Web server).

L'esempio di servlet illustrato precedentemente è molto semplice, per cui non risultano molto evidenti gli inconvenienti di cui sopra. Si immagini però una servlet che generi una pagina HTML molto più elaborata con tantissimi tags innestati tra loro e si avrà così chiara la problematica in questione.

Per ovviare a questi problemi la J2EE raccomanda l'utilizzo di JSP per la generazione di pagine web dinamiche.

JSP architecture

Le pagine JSP sono documenti di testo che forniscono una descrizione di come processare una request per creare una response utilizzando un determinato protocollo (l'http è quello di default).

Sarà compito del web-container interpretare nel modo corretto il contenuto di una pagina JSP traducendolo in una servlet che potrà poi essere eseguita dal server per generare la pagina dinamica. Le specifiche servlet e JSP della Sun forniscono le direttive standard che deve seguire un web-container che supporta le servlet (detto quindi anche servlet-container).

Le pagine JSP semplificano quindi il lavoro di quei Web Designer che non conoscono il linguaggio Java. Quando un Web designer cambia una pagina JSP questa e' automaticamente ricompilata e ricaricata dentro al Web Server (inoltre tutte le pagine JSP possono essere compilate a priori fornendo una maggiore efficienza all'applicazione). Le pagine sviluppate negli standard sintattici delle JSP, non possono essere formattati in modo completo dal XML ecco quindi che è stata intrdotta una sintassi alternativa chiamata XML Schema Definition Language (XSDL) che gestisce anche potenziali errori verificabili solo in esecuzione.

Una pagina JSP che realizza la servlet dell'esempio precedente è la seguente:

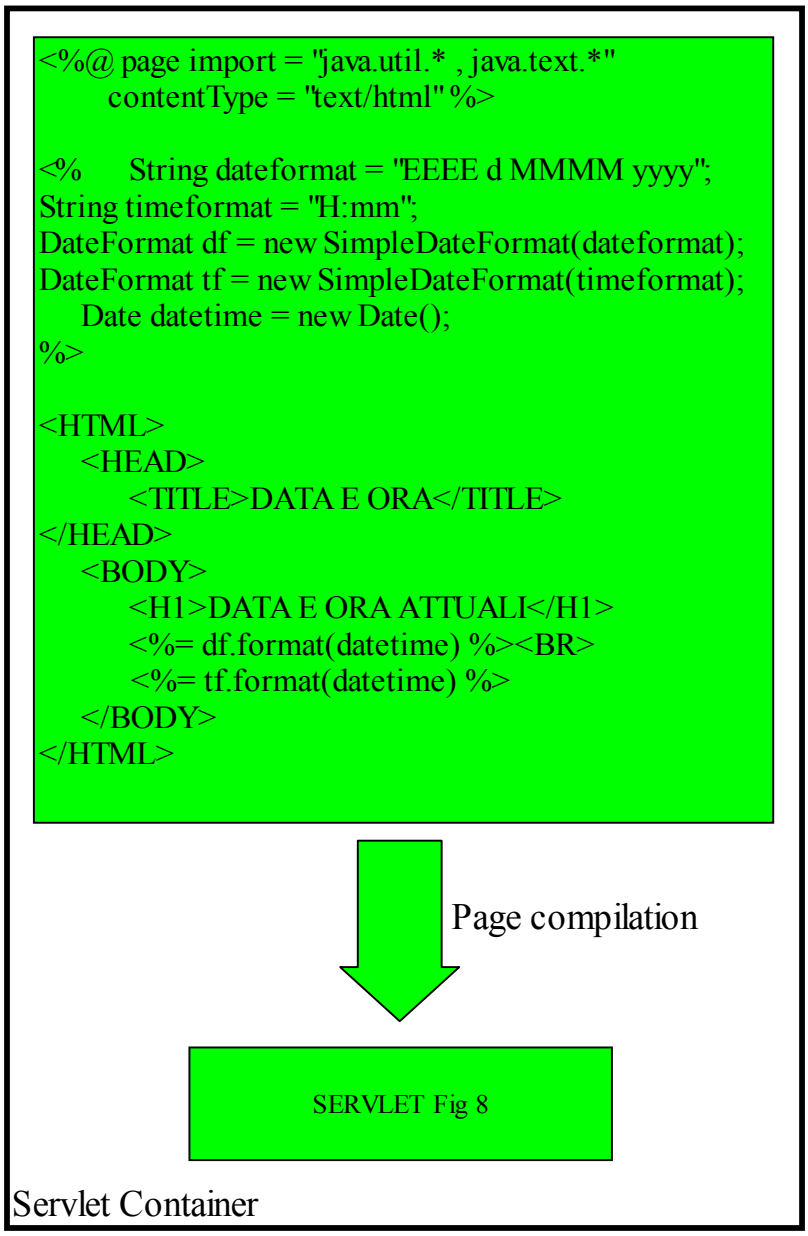


Figura 12 Codice di esempio – Pagina JSP che realizza la servlet precedente –

La tecnologia JSP permette inoltre agli sviluppatori di definire dei ***Custom tags*** ossia dei tags che vengono sostituiti da contenuti dinamici nel momento in cui viene servita la pagina. L'insieme di custom tags che forniscono funzionalità di base per le pagine JSP possono essere raccolti in Standard Tags Libraries (STL) e rappresentano un modo efficace per ridurre la quantità di codice Java dentro una pagina JSP fornendo inoltre una migliore gestione della manutenzione.

Sebbene sia un' opinione comune che i componenti web siano principalmente utilizzati per fornire una presentazione dell' applicazione, nelle applicazioni J2EE questi possono avere due ruoli: quella di presentare soltanto i dati (presentation components) e quella di filtrare le richieste e reindirizzarle alle pagine corrispondenti (front components).

Nel primo caso di funzionamento (come presentation components) le richieste che arrivano dal web browser vengono direttamente indirizzate alla pagina JSP la quale è responsabile della generazione della risposta HTML/XML che determinano l'interfaccia del client. Anche in questo caso esiste ancora separazione fra presentation e contenuti, in quanto tutta la gestione dei contenuti viene affidata a dei Java Beans.

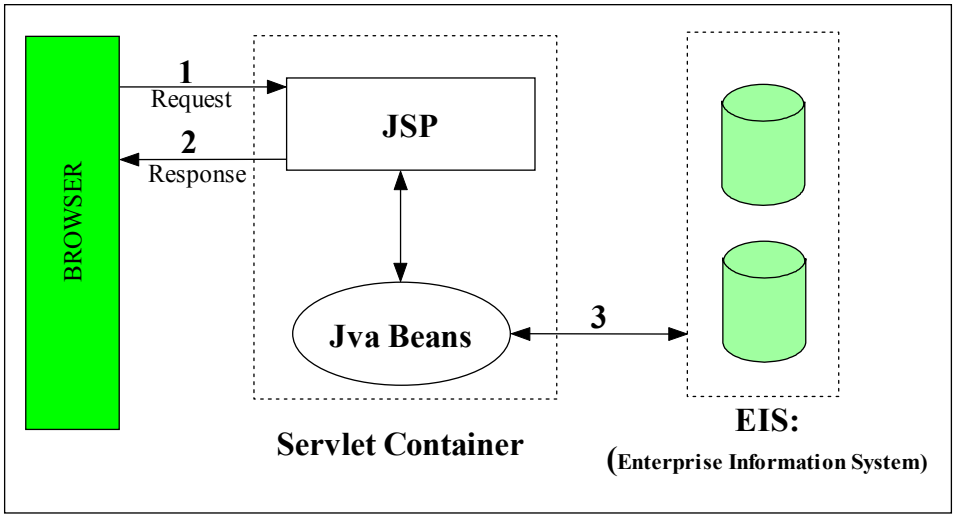


Figura 13: - JSP come presentation Component –

L'architettura rappresentativa del secondo caso rappresenta il Server-side del popolare modello (Model/View/Controller). In questo caso il processo è diviso fra presentation e front component. I presentation components sono le pagine JSP le quali generano la risposta HTML/XML da tornare al client tramite un reindirizzamento sul web browser, mentre il front component (anche conosciuto come controller) non ha nessun compito di presentazione, ma si limita a processare tutte le richieste http. Risulta inoltre di sua competenza creare qualsiasi bean o oggetto che sarà poi utilizzato dal presentation component. Il front component può a sua volta essere implementato sia con servlet che come pagine JSP.

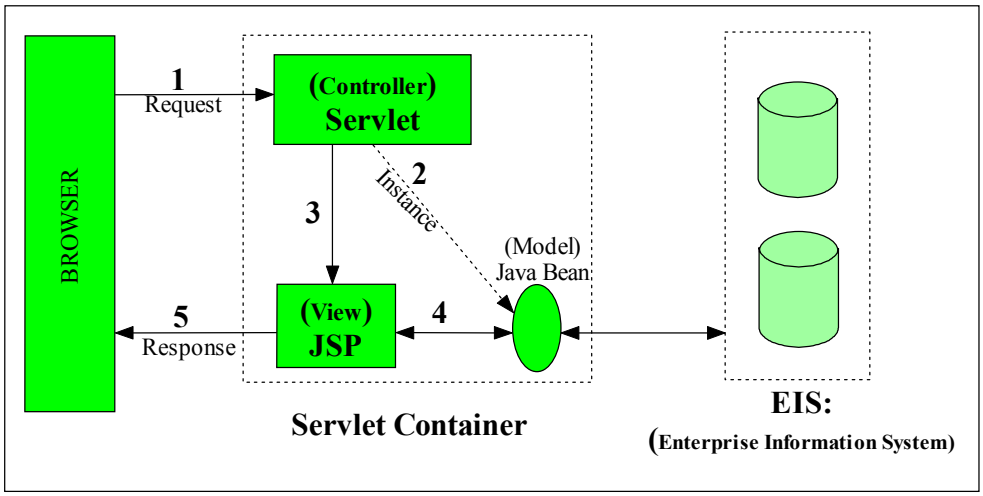


Figura 14: - JSP come front component –

Il vantaggio di questo architettura è rappresentato dal fatto che non ci siano processing logic mescolati con presentation component; il presentation component è responsabile semplicemente di utilizzare quegli oggetti o bean creati o istanziati precedentemente dal controller. Questa chiara separazione fra presentazione e contenuti permette di individuare nitidamente i ruoli e le responsabilità degli sviluppatori, in particolari dei programmatori da quelli dei disegnatori di pagine web.

Un altro fondamentale beneficio di questa architettura è dato dal fatto che il front component possa rappresentare un unico punto di accesso a tutte le applicazioni e a tutti i dati contenuti nell'Enterprise Information System.

JSP – Sintassi di base

Per quanto riguarda le espressioni JSP, queste possono essere riconosciute come tali e opportunamente interpretate dal web-container solo se rientrano nelle seguenti quattro differenti categorie:

Directive: sono direttive di carattere generale, indipendenti dal contenuto specifico della pagina, relative alla fase di traduzione/complilazione; Sintassi: `<%@ directive ... %>`.

Le directive sono tre:

page : definisce diverse proprietà relative alla pagina stessa che vengono comunicate al web-container;

taglib : dichiara le tag libraries che verranno usate nella pagina (vedere oltre);

include : comunica al container che nella fase di traduzione deve includere nella pagina il contenuto di un file il cui nome viene specificato nella direttiva stessa;

Action: vengono eseguite nella fase di processing e danno origine a codice Java specifico per la loro esecuzione; Sintassi: quella dei tags XML, cioè `<tag attr1="value1" attr2="value2" ... > body </tag>`

Scripting element: sono frammenti di codice Java, o in un altro linguaggio specificato e si dividono a loro volta in tre sottocategorie:

Scriptlet: sono veri e propri statement scritti nello scripting language che danno origine a porzioni di codice Java all'atto della traduzione in servlet; Sintassi: `<% codice %>`

Declaration: sono dichiarazioni che vengono inserite nella classe Servlet come elementi della classe stessa, senza essere racchiuse in alcun metodo. Possono essere sia variabili di classe che metodi. Se si tratta di variabili, la loro durata e' quella del Servlet quindi sopravvivono e conservano il loro valore nel corso di tutte le esecuzioni dello stesso Servlet. Sintassi: `<%! Declaration [declaration] ... %>`

Expression: contengono un'espressione che segue le regole delle espressioni dello scripting language; l'espressione viene valutata e scritta nella pagina di risposta nella posizione corrispondente a quella dell'espressione JSP. Sintassi: `<%= expression %>`

Comment: sono commenti che riguardano la pagina JSP e che vengono eliminati dal web-container nella fase di traduzione; sintassi: `<%-- comment --%>` Una JSP che realizza la pagina dell'esempio precedente è la seguente:

```
<%@ page import = "java.util.* , java.text.*"
         contentType = "text/html" %>

<%      String dateformat = "EEEE d MMMM yyyy";
String timeformat = "H:mm";
DateFormat df = new SimpleDateFormat(dateformat);
DateFormat tf = new SimpleDateFormat(timeformat);
  Date datetime = new Date();
%>

<HTML>
  <HEAD>
        <TITLE>DATA E ORA</TITLE>
</HEAD>
  <BODY>
        <H1>DATA E ORA ATTUALI</H1>
        <%= df.format(datetime) %><BR>
            <%= tf.format(datetime) %>
  </BODY>
</HTML>
```

Figura 15: Codice di esempio – JSP che realizza la pagina dinamica con data e ora –

Essa verrà tradotta dal web-container nella servlet precedente la cui esecuzione fornisce esattamente la pagina HTML dell'esempio visto prima.

Si è fatto uso di una Directive page nella quale si sono specificate due proprietà associate alla JSP, la prima mediante l'attributo import al quale è stato assegnato come valore la lista dei package Java da importare, la seconda mediante l'attributo contentType al quale si è assegnato il valore "text/html", cioè il tipo di contenuto da inviare al client come pagina HTML. Quest'ultimo attributo se non specificato viene assunto per default come "text/html" con charset = Latin-1 ad indicare il set di caratteri utilizzato per codificare la pagina.

Come si può notare, per mezzo di una JSP un Web designer può avere subito una immagine chiara di come risulterà la pagina web. Inoltre, essendo le JSP dei

documenti di testo, egli può anche utilizzare dei tool per creare e visionare il loro contenuto.

La J2EE raccomanda anche di fare meno uso possibile di elementi scriptlet.

Una JSP con molti scriptlet al suo interno, infatti, può causare alcuni problemi:

1) diventa più difficile da leggere ed è necessaria la conoscenza del linguaggio di scripting per capirne la funzionalità;
2) nel caso in cui alcune funzionalità debbano essere ripetute in più pagine, occorre riportare gli scriptlet ricopiandoli con un copia e incolla, con evidente ridondanza di codice e difficoltà di manutenzione;

Al contrario una JSP con pochi scriptlet, o addirittura senza, non solo evita i problemi sopraelencati, ma consente una ancor più netta seprazione del codice dalla presentazione, favorendo in tal modo la manutenzione e la separazione dei ruoli, e quindi il parallelismo, tra web designer che può concentrarsi sull'aspetto della presentazione, e il web developer che si occupa dei contenuti trascurando l'interfacciamento grafico.

I due principali meccanismi che consentono di raggiungere questo obiettivo sono i JavaBeans e i Custom Tags.

JavaBeans

I JavaBeans sono delle classi Java facilmente riusabili e composte insieme in una applicazione. Ogni classe Java che segue certe regole di composizione, può essere un JavaBean component.

Tali regole governano le proprietà della classe del JavaBean e i metodi pubblici che ne danno accesso.

Ogni JavaBean, infatti ha una serie di proprietà che possono essere di sola lettura, di sola scrittura o di lettura e scrittura.

Ogni proprietà leggibile deve avere un metodo pubblico getter del tipo

```
PropertyClass getProprty() { ... }
```

e ogni proprietà settabile deve avere un metdo pubblico setter del tipo

```
setProperty ( PropertyClass pc) { ... }
```

Inoltre un JavaBean deve avere un costruttore senza parametri.

Le JSP supportano l'utilizzo di JavaBeans diretto attraverso Actions che possono essere inserite nella pagina:

1. ***jsp:UseBean*** : serve per utilizzare un bean già esistente o crearne una nuova istanza;
   i suoi attributi sono:
   a. *id* : definisce il nome di una variabile che identifica il bean nello scope specificato;
   b. *scope* : definisce l'ambito di esistenza e di visibilità della variabile il cui nome è definito dall'attributo id. Si tratta dunque di un attributo che specifica a quale scope object andrà associato il bean. I valori ammessi sono i seguenti:

c. *page* : la variabile è utilizzabile solo all'interno della pagina in cui compare il tag, o di una pagina inclusa staticamente;
d. *request* : la variabile è utilizzabile nell'ambito di una singola request;
e. *session* : la variabile è utilizzabile nell'ambito di un'intera sessione; la pagina in cui il bean è creato deve contenere una direttiva page con l'attributo session settato a true;
f. *application* : la variabile è utilizzabile nell'ambito dell'intera applicazione da tutte le sue pagine JSP.
g. *class*: definisce il n ome della classe Java a cui il bean appartiene;
h. *type* : identifica il tipo della variabile il cui nome è definito dall'attributo id, che può così anche non essere necessariamente la classe specificata dall'attributo class, ma anche una superclasse o un'interfaccia implementata dalla classe;

2. ***jsp:getProperty***: inserisce nella pagina il valore di una proprietà del bean;
3. ***jsp:setProperty***: assegna il valore di una o più proprietà del bean

Il body contenuto all'interno dell'Action `jsp:useBean` viene eseguito solo all'atto dell'istanziamento del bean.

L'utilizzo dei JavaBeans è quindi vantaggioso poiché consente di incapsulare del codice Java al suo interno e di inserirlo nella JSP attraverso i tag sopraelencati, invece di immetterlo direttamente nella pagina come scriptlet.

Dalla pagina è possibile manipolare il bean attraverso il settaggio e il ripescaggio delle sue property, ma anche richiamando direttamente da uno scripting element un suo metodo attraverso la variabile identificata col nome specificato nell'attributo id.

Ritornando all'esempio di prima, si può creare il seguente bean:

```
import java.util.*;
import java.text.*;

public class DateTime {
  String dateformat = “EEEE d MMMM yyyy”;
String timeformat = “H:mm”;
DateFormat df = new SimpleDateFormat(dateformat);
DateFormat tf = new SimpleDateFormat(timeformat);

public String getDate() {

Date datetime = new Date();
return df.format(datetime);
  }


public String getTime() {
Date datetime = new Date();
return tf.format(datetime);
  }
}
```

Figura 16: Codice di esempio – JavaBean per incapsulare il codice per visualizzare data e ora –

Poi si può richiamare il bean dalla pagina utilizzando le Actions descritte, perciò la JSP avrà il formato riportato qui di seguito, nel quale si è omessa la Directive non essendo più necessario specificare l’attributo import e avendo assunto il contentType di default :

```
<jsp:useBean id=”datetime” class=”DateTime” scope=”page” />

<HTML>
  <HEAD>
        <TITLE>DATA E ORA</TITLE>
</HEAD>
  <BODY>
        <H1>DATA E ORA ATTUALI</H1>
        <jsp:getProperty name=”datetime” property=”date”/>
<BR>
            <jsp:getProperty name=”datetime” property=”time”/>
  </BODY>
</HTML>
```

Figura 17: Codice di esempio – JSP che utilizza il precedente JavaBean per generare data e ora –

In quest’ultima versione della pagina JSP si può apprezzare la differenza con la versione script: facendo uso del JavaBean per incapsulare il codice Java la pagina appare molto più pulita e compatta e il contenuto risulta di facile e intuitiva comprensione. Inoltre tutta la funzionalità racchiusa nel JavaBean può essere utilizzata anche da web designers che, senza necessità di conoscenza del linguaggio Java, possono occuparsi esclusivamente del layout delle pagine in cui dovrà essere inserita la data e l’ora. Qualora si volesse effettuare una qualunque modifica, ad esempio cambiare il formato di visualizzazione della data, sarebbe sufficiente cambiare la stringa `dateformat` e ricompilare il JavaBean per ottenere automaticamente l’aggiornamento di tutte le pagine JSP che lo utilizzano, oppure si

potrebbe prevedere un metodo setFormat che consenta di personalizzare il formato di pagina in pagina semplicemente settando una property del bean.

Custom Tags

Come si è visto i JavaBeans forniscono un potente strumento per incapsulare funzionalità di presentation logic da inserire in una JSP. Essi però sono manipolabili solo attraverso le loro properties, mentre in taluni casi potrebbe essere utile poter far uso di strumenti in grado di offrire maggiore flessibilità.

Dalla versione 1.1 delle specifiche JSP è stata introdotta un'importante innovazione: la definizione delle tag extension, ossia un meccanismo attraverso il quale gli sviluppatori possono estendere le JSP con tags creati ad hoc, i cosiddetti custom tags, che, una volta dichiarati e inclusi nella JSP, diventeranno delle action a tutti gli effetti. Possono così essere create delle vere e proprie librerie di nuovi tags (tag libraies) con le caratteristiche di riusabilità e portabilità, che forniscono agli web designers potenti strumenti con sintassi XML a loro molto più familiari che non i linguaggi di scripting.

Per implementare un custom tag si deve definire una classe, chiamata genericamente tag handler, che implementi una delle tre interfacce Tag , BodyTag o IterationTag, definite nel package javax.servlet.jsp.tagext.

L'interfaccia Tag viene usata per implementare un tipo di tag empty, cioè non dotato di body, oppure per implementare una action che richiede semplicemente l'esecuzione di operazioni quando viene incontrato il tag iniziale e altre operazioni quando viene incontrato quello finale. Questa interfaccia dunque contiene i metodi di base che sono tipici di tutti i tag handlers, ossia metodi setter per inizializzare il tag con un context e gli eventuali attributi dell'azione, e i metodi che devono essere ridefiniti per implementare la funzionalità vere e proprie da attribuire al custom tag; in particolare questi ultimi sono due: doStartTag e doEndTag.

Il metodo doStarTag deve contenere le operazioni che il container dovrà eseguire nel momento in cui nell'elaborazione della JSP viene incontrato il tag iniziale. Esso restituisce una valore intero che indica al container se e come compiere un processing del body, qualora il tag lo preveda. In particolare esistono due differenti valori di ritorno che indicano due differenti alternative:

1. ignorare il body (utilizzato per i tag empty);
2. valutare il body e inserirne la valutazione direttamente nello stream di output della response (utilizzato ad esempio per inclusioni condizionali).

Il metodo doEndTag deve contenere le operazioni che il container dovrà eseguire nel momento in cui nell'elaborazione della JSP viene incontrato il tag finale. Il suo valore di ritorno è un intero che indica al container se il resto della pagina deve essere valutato o meno.

L'interfaccia IterationTag è un'estensione della interfaccia Tag che fornisce un ulteriore metodo, doAfterBody, invocato dal container per la rivalutazione del body del tag, qualora si voglia implementare un tag iterativo. Esso infatti restituisce un intero che indica sel il body deve essere rivalutato ancora o se deve essere richiamato il metodo doEndTag.

L'interfaccia BodyTag è un'estensione di IterationTag che contiene ulteriori due metodi che devono essere ridefiniti qualora si richieda una gestione più

complessa del body del tag. Tali metodi sono: `setBodyContent`, `doInitBody`. Il primo è chiamato dal container prima della valutazione del body, solo nel caso questo abbia luogo e assegna al tag un oggetto `BodyContent` che funge da buffer temporaneo per l'output del body. Il secondo viene chiamato dopo il primo soltanto alla prima valutazione del body (e non alle eventuali successive iterazioni).

Per l'interfaccia `BodyTag` il metodo doStartTag può ritornare, come avveniva per l'interfaccia Tag, un valore che indica al container di ignorare il body, mentre l'altra possibilità prevista è quella di valutare il body e inserirlo nel `BodyContent`, invece di mandarlo direttamente sullo strema di output.

Esistono anche delle classi predefinite che implementano le interfacce di cui sopra, e che possono essere utilizzate direttamente come tag handler, al limite ridefinendo solo alcuni metodi strettamente necessari. Esse sono `TagSupport` e `BodyTagSupport`.

All'interno di una JSP è possibile importare più librerie di tag specificandole attraverso la directive taglib. Ciò viene fatto riportando un attributo della directive il cui valore è un URI univocamente associato ad una tag library, e un ulteriore attributo il cui valore è il prefisso che verrà utilizzato per identificare i custom tags della tag library quando verranno inseriti nella JSP.

Nel deployment descriptor web.xml della applicazione web in cui la tag library viene utilizzata deve essere riportato un mapping tra l'URI della tag library e un percorso in cui si trova un particolare file XML detto *TLD (Tag Library Descriptor).*

```
<taglib>
  <taglib-uri>
        http://sparc20.ing.unimo.it:8063/md/taglib
</taglib-uri>

  <taglib-location>
        /WEB-INF/taglib/tagdescriptor.tld
</taglib-location>

</taglib>
```

Figura 18: Codice di esempio - Esempio della sezione del deployment descriptor che specifica una taglibrary –

Il TLD è un file di configurazione in formato XML, che fornisce al container le informazioni necessarie per poter utilizzare correttamente la libreria. Esso riporta per, ciascuna action della tag library, il nome del tag, il nome della classe del tag handler, informazioni su tutte le eventuali variabili di scripting create dall'action e informazioni sugli eventuali attributi dell'action.

Questo file va inserito in un percorso della applicazione web che deve corrispondere a quello riportato nel tag `<taglib-location>`. Inoltre occorre inserire nel percorso in cui tipicamente vengono riportate le libreie di classi utilizzate dalla applicazione web, tutte le classi utilizzate nella tag library. Solitamente queste classi si trovano impacchettate in un file JAR da inserire nella directory WEB-INF/lib.

Il DTD del file TLD è riportato nelle specifiche JSP dalla versione 1.1 in poi.

3.4.3 Client tier

Costituisce il front-end più o meno complesso che risiede e viene eseguito sul client che accede alla applicazione. Esso inoltra le richieste al server per conto dell'utente e ne presenta a quest'ultimo i risultati.

La J2EE supporta vari tipi di client che possono connettersi sia all'interno del firewall aziendale che da fuori attraverso il World Wide Web.

Un client può comunicare con, e usare i servizi forniti da, uno o più tiers dell'applicazione enterprise.

A seconda del tier cui si collega, si possono distinguere tre tipologie principali di client: il *Web-client* l'*EJB-client* e l'*EIS-client*

Capitolo 4

Confronto fra le piattaforme e le tecnologie dei principali prodotti presenti nel mercato degli EIPs

Il Web (World Wide Web) è una struttura ancora relativamente nuova, ma la sua diffusione sia fra utenti privati che presso le aziende ha avuto una rapida crescita. Mentre gli utenti privati si servono del web per una serie di scopi diversi, le aziende lo utilizzano principalmente per fornire prodotti, servizi ed informazioni a tutti coloro che entrano a far parte della loro catena di business.

Compito delle aziende operanti nel settore dei servizi di informazione e più in particolare, di quelle software, è stato fornire soluzioni che creassero l'infrastruttura ed i servizi per affiancare e sostenere le imprese nella loro apertura al mondo web. Questo è lo scenario che ha spinto aziende private e gruppi di ricerca a studiare ed implementare soluzioni informatiche capaci di integrare servizi, informazioni e dati interni ed esterni all'impresa. Obiettivo di questo capitolo è visionare i prodotti offerti attualmente da alcune delle principali aziende informatiche, così da fornire una istantanea delle tecnologie sviluppate in ambito privato da confrontare con le soluzioni offerte in ambito OpenSource.

4.1 IBM: WebSphere

La famiglia WebSphere, offerta da [IBM], è un insieme di prodotti software per lo sviluppo e la gestione di siti Web ad alte prestazioni e l'integrazione di questi siti con sistemi aziendali non Web nuovi o esistenti. I prodotti offerti da IBM si rivolgono principalmente alle seguenti aziende:

- Aziende che desiderano utilizzare le più recenti tecnologie per creare una forte presenza sul Web o per aggiornare quella attuale
- Aziende che desiderano sviluppare applicazioni e sistemi aziendali distribuiti ed estesi
- Aziende che desiderano integrare la loro presenza sul Web con sistemi e applicazioni di tipo non Web

Allo scopo di fornire un unico punto di accesso a tutte le informazioni e servizi fruibili dalla rete, IBM offre WebSphere Portal Server: un EIP che permette agli utenti di interagire con tutte le applicazioni, i contenuti, le persone ed i processi

aziendali. WebSphere Portal Server offre agli utenti la possibilità di organizzare e gestire una visualizzazione personale del portale e fornire servizi addizionali come gestione dei contenuti, ricerche di documenti automatizzate, gestione di tassonomie, supporto per dispositivi mobili e tanti altri servizi che tuttora vengono sviluppati ed integrati nel portale.

4.1.1 WebSphere Portal Server Architecture

WebSphere Portal Server è composto da numerosi prodotti che vengono integrati in diverse soluzioni, a seconda delle esigenze dei clienti. Le principali tecnologie sulle quali si appoggia l'EIP di IBM sono le seguenti:

- WebSphere Portal Server Framework- Un'infrastruttura di portale dotata di funzionalità per l'integrazione del portale con applicazioni e fonti di dati, per le attività amministrative e gestionali quali l'abbonamento al portale e i servizi di connettività e di presentazione. WebSphere Portal Server rappresenta il punto di connessione per aggiungere al portale applicazioni e servizi rivolti ai clienti.
- WebSphere Application Server, Advanced Edition- (definito anche Advanced Application Server) Introduce le capacità del server per le applicazioni create, per le specifiche Enterprise JavaBeans della Sun Microsystems e fornisce supporto per l'integrazione di applicazioni Web ad altri sistemi aziendali non orientati al Web.
- Portlets- WebSpherePortalServer viene venduto con svariati Tools per la semplificazione dello sviluppo dei portlet. Vengono inoltre integrati nel pacchetto base portlet pronti all'uso facilmente integrabili sull'infrastruttura.
- Tivoli SecureWay LDAP Directory- tale applicativo fornisce una modalità standard di autenticazione al porale.

In più oltre a questi applicativi base integrati nel WebSphere Portal Server, è possibile aggiungere, con maggiorazione al prezzo base, altri servizi come:

- **Enterprise Information Portal v7**- utilizza avanzate tecniche di web crawling e text mining sviluppate dai ricercatori IBM. Queste tcniche permettono all'EIP di catalogare, aggregare ed indicizzare documenti di testo, immagini, audio, video in modo da fornire un valido supporto ai motori di ricerca, in modo tale da rendere accessibili la maggior parte dei documenti.
- **QuickPlace, Sametime, MindSpan, Email**- risultano facilmente integrabili nel portale i prodotti di Knowledge Management e collaborazione offerti da Lotus Corporation's, così da includere funzionalità di gruppi di lavoro quali la messaggistica immediata e la team room.
- **WebSphere Personalization**- un servizio integrato per la gestione di regole base e di filtraggi .

- **Ed altri ancora**…

WebSphere Portal Server viene attualmente installato su piattaforme: **UNIX** (IBM AIX e Sun Microsystems Solaris) e su **Microsoft** Windows NT. WebSphere Application Server è inoltre disponibile per le piattaforme OS/390 e AS/400.

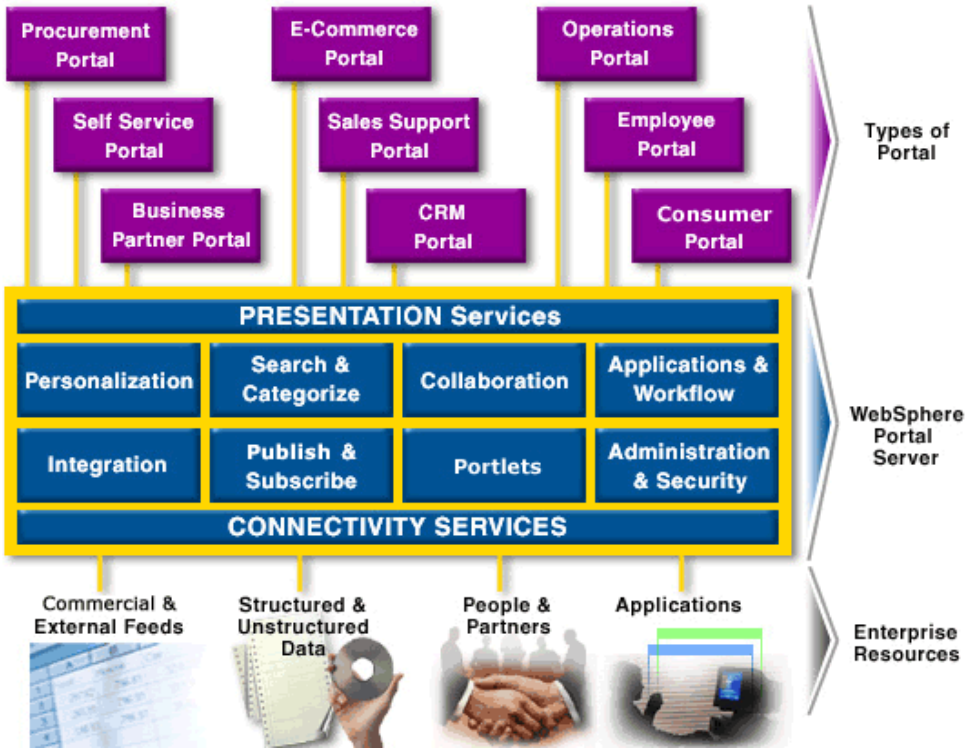


Figura 19: -Architettura WebSphere Portal Server-

Fonte: http://www-3.ibm.com/software/webservers/portal/architect.html

WebSphere Portal Server risulta essere il punto di contatto fra l'insieme di tutte le risorse ed i processi sia interni che esterni all'azienda e le loro visualizzazioni gestite attraverso il portale. Lo strato di WebSphere Portal Server si preoccupa quindi di gestire la personalizzazione, l'integrazione, la pubblicazione, la collaborazione fra i vari dati di un'impresa, collocandosi sullo strato web ("web-tier") dell'architettura fornita da webSphere Application Server.

4.1.1.1 WebSphere Application Server

La piattaforma di WebSphere Portal Server si appoggia su WebSphere Application Server: un componente che offre un ambiente per l'elaborazione

distribuita aperta nella quale gli utenti e le procedure, disposti anche su una serie di piattaforme diverse, possono interagire grazie alle funzioni fornite dal pacchetto.

Proprio come nella architettura J2EE, il software da eseguire su sistemi distribuiti viene organizzato in due settori principali: il lato client ed il lato server. In tale struttura il client invia la richiesta di utilizzo di un servizio reso disponibile dal lato server, il quale si preoccupa di fornire e gestire i servizi offerti . I programmi client gestiscono in genere le interazioni tra gli utenti e spesso richiedono dati o avviano le modifiche ai dati a favore di un utente. Un progetto comune di sistemi per client e server utilizza tre elementi: un client che interagisce con un utente, un server di applicazioni che contiene la logica aziendale dell'applicazione e un programma di gestione risorse che ordina i dati. Questa procedura è illustrata nella Figura sotto riportata. In questo modello, il client non deve necessariamente conoscere l'attuale programma di gestione risorse. Se il database in uso viene modificato, potrebbe essere necessario modificare anche il server, ma non il client. Poiché in genere esiste un numero minore di copie del server rispetto al client, e i server si trovano spesso in posizioni più semplici da aggiornare (per esempio, su macchine centrali invece che su computer in esecuzione sulle scrivanie degli utenti), anche la procedura di aggiornamento risulta semplificata. Inoltre, questa procedura garantisce una maggiore sicurezza. Solo i server, e non i client, devono accedere ai dati controllati dal programma di gestione risorse.

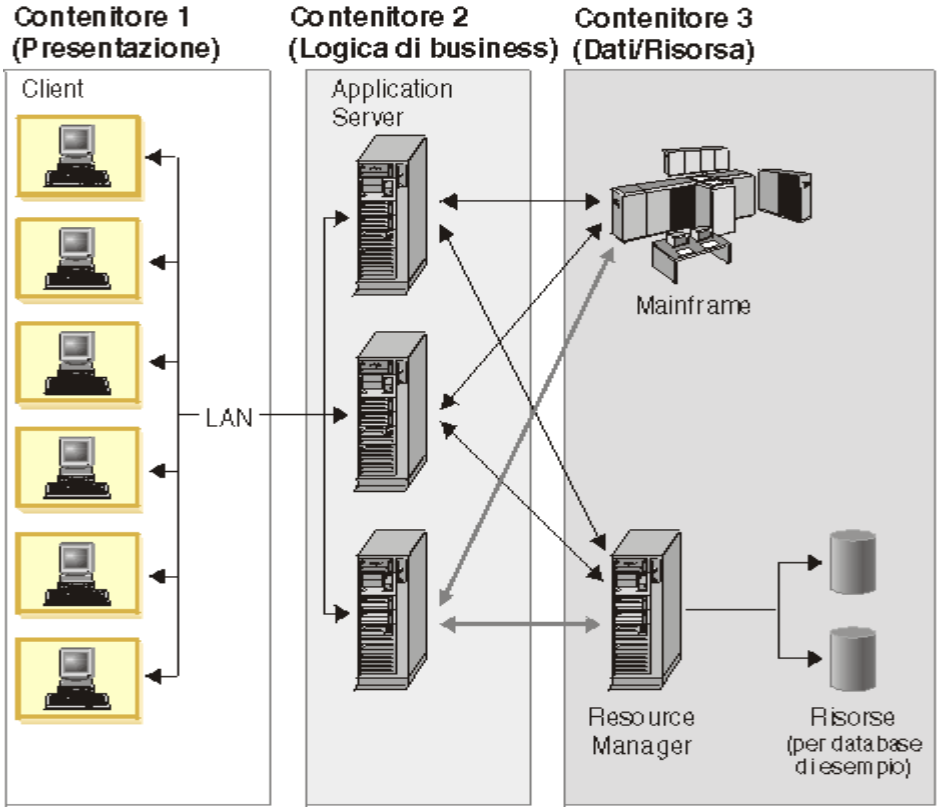


Figura 20: -Architettura multilivello di WebSphere portal server –

WebSphere Application Server costituisce il livello centrale di questa struttura e consente ai client-applet, client Visual Basic, client C++ e via di seguito di interagire con le risorse dati (database relazionali, MQSeries e simili) e al tempo stesso con le applicazioni esistenti.

I componenti integrati in Advaced Application Server e quindi combinabili per creare sistemi potenti su tre livelli basati su Java vengono illustrati in Figura.

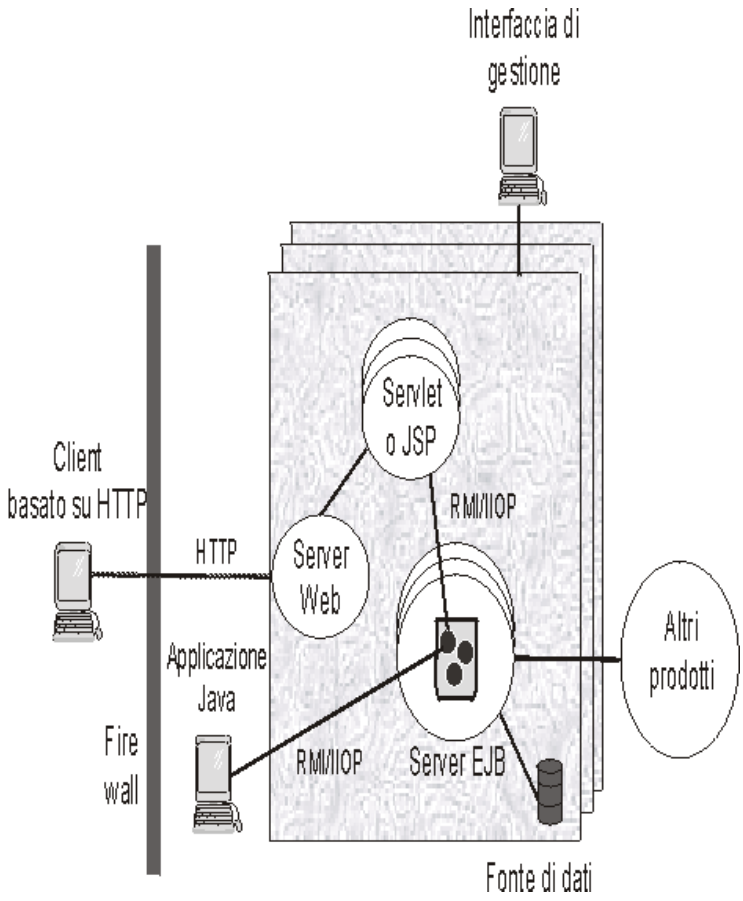


Figura 21: - I componenti dell'ambiente di Advanced Application Server –

I principali componenti integrati nell'ambiente WebSphere sono:

- *Applicazioni basate su browser*—Consentono di inviare e ricevere informazioni dai siti Web utilizzando il protocollo http (Hypertext Transfer Protocol). Esistono tre tipi principali di applicazioni basate su browser: applet Java, servlet Java e JSP (JavaServer Pages).
- server Web—A parte applet Java di tipo standalone, che sono limitate dalla protezione interna Java, per le applicazioni basate su browser è

necessario installare, su almeno una macchina nell'ambiente Advanced Application Server, uno qualsiasi dei tanti server Web supportati. Il Server Web fornisce il link tra le applicazioni basate su browser e gli altri componenti di Advance Application Server. Il server http IBM, associato a Advanced Application Server, costituisce una versione modificata del server di Apache.

- server EJB e bean enterprise—Il server WebSphere EJB contiene uno o più bean enterprise, che comprendono i dati e la logica aziendale utilizzati e condivisi dalle applicazioni EJB. I bean enterprise installati in un server EJB non comunicano direttamente col server; un contenitore EJB fornisce un'interfaccia tra i bean enterprise e il server EJB, che a sua volta fornisce alcuni servizi semplici come thread, supporto per transazioni e gestione memoria e recupero dati.
- applicazioni Java—Le applicazioni Java possono interagire direttamente con un server EJB utilizzando RMI (remote method invocation) Java su IIOP (Internet Inter-ORB Protocol).
- Origini dati—Esistono due tipi di bean enterprise: bean di sessione, che comprendono attività e oggetti di breve durata, specifici per client, e bean di entità, che comprendono dati persistenti o permanenti. Il server EJB memorizza e recupera questi dati permanenti in un database.
- server di amministrazione e interfaccia di amministrazione—Il server di amministrazione gestisce servlet, file JSP, bean enterprise e server EJB. Questa gestione è diretta dall'amministratore di WebSphere Application Server che utilizza WebSphere Administrative Console, l'interfaccia di amministrazione per il relativo server.

4.1.1.2 WebSphere Portlets

L'interfaccia utente, o meglio il front end delle applicazioni distribuite gestite attraverso Webspehere, avviene attraverso l'utilizzo di una infrastruttura detta framework, sulla quale vengono caricate delle piccole applicazioni java riusabili chiamati portlets. Questi componenti hanno la capacità di raccogliere contenuti interfacciandosi con servizi web, tool di collaborazione e servizi di back-end aziendali. I portlet vengono collocati ed aggregati con altri portlet all'interno delle pagine del portale e vengono visualizzati contemporaneamente alla chiamata del portale.

La tecnologia utilizzata da IBM per la gestione del portale e dei singoli portlet, estende quella fornita da Jetspeed: un EIP sviluppato e distribuito in modo open source dal gruppo di ricerca internazionale di volontari uniti nell'Apache Software Foundation's Jakarta Project (vedi capitolo 5). In particolare WebSphere Portal Server, estende alcune delle potenzialità offerte dalle portlet API di Jetspeed e le integra in alcuni Tool di sviluppo. L'estensione delle API e la loro integrazione nei Tool di sviluppo, ha lo scopo di fornire agli utenti di WebSphere Portal Server un ambiente più semplice, pratico ed intuitivo nel quale poter sviluppare nuovi portlet e

gestire l'intero portale. Oltre a queste caratteristiche, nel pacchetto venduto da IBM vengono integrate alcune tipologie di portlet già costruite, e quindi applicazioni di base già sviluppate e riutilizzabili. Al fine di visionare alcuni dei servizi offerti dal pacchetto di IBM, vengono ora analizzati in dettaglio i portlet installati di default sulla Home page di WebSphere Portal v. 4.1.

World Clock

Il World Clock Portlet permette di settare e visualizzare la data e l'orario corrente specificando le varie zone di interesse.

Reminder

Il portlet di reminder permette di visualizzare, scrivere e cancellare appuntamenti o promemoria.

QuickLinks

La funzionalità di QuickLinks Portlet è di gestire e quindi Inserire, cancellare e modificare i link che più frequentemente vengono visitati dall'utente. Tale portlet ha la capacità di interfacciarsi con la sezione "bookmark" dei browser Netscape ed Explorer .

In Figura è riportata la HomePage di IBM WeBSphere Portal nella configurazione di default nella quale sono stati incorniciati i portlet sopra elencati.

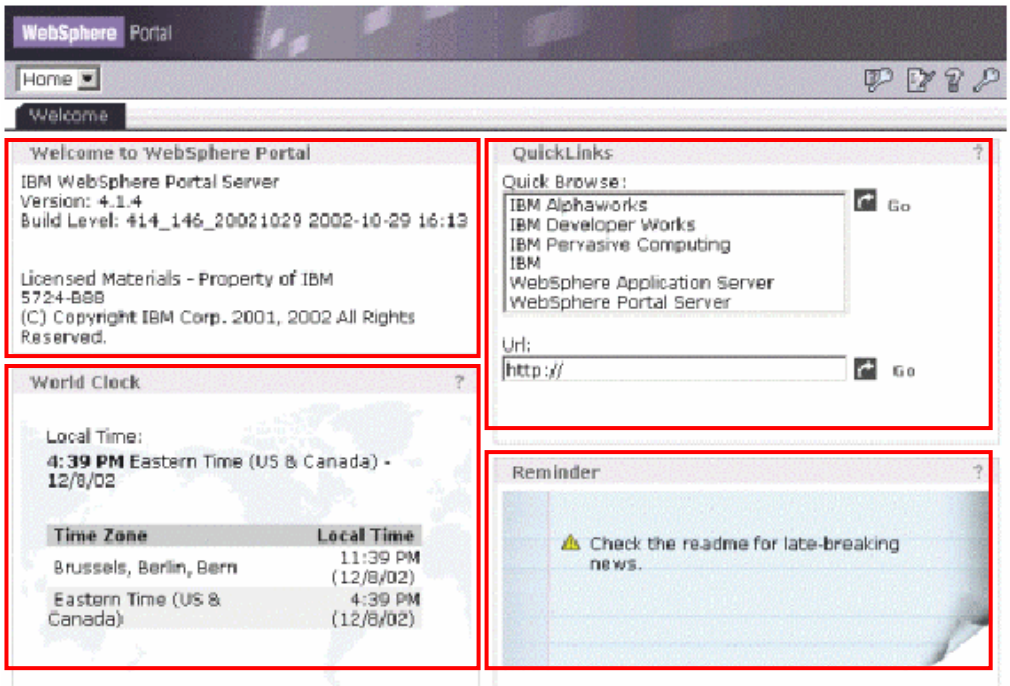


Figura 22: -HomePage di WebSphere Portal –

(Fonte: "Viewing the WebSohere Portal V4.1 Portlets" Sung-IK-Son WebSphere Enablement & consultino Team ; dicembre 2002)

Figura 23: - Visualizzazione di Reminder e QuickLinks portlets attraverso tecnologia Wap-

Altre tipologie di portlets offerte da IBM possono essere scaricate gratuitamente da internet, oppure possono essere acquistate a seconda delle necessità. WebSphere Portal Server risulta quindi un framework sul quale è possibile applicare vari servizi di base ed altri aggiuntivi come:

Internet Mail Box:

Internet Mail Box è un portlet che permette di spedire e ricevere mail, ed inserire allegati. Internet Mail Box dà la possibilità di spedire e ricevere mail attraverso l'uso della tecnologia (WAP) Wirless Application Protocol , oppure offre la possibilità di accedere dal portale direttamente ad un mail server, configurando preventivamente i parametri della connessione.

Document Viewer Portlets:

Document Viewer Porlets permette all'utente di visualizzare direttamente dal portale documenti sviluppati in Word, Excel, Power Point, PDF o in Rich Test Format. Attraverso tale servizio, l'utente può decidere se lavorare all'interno del portale, oppure portare il documento su una nuova finestra e quindi deciderne le dimensioni.

Driving Directions :
La funzione di questo portlet è di fornire all'utente, noto il punto di partenza, la strada più breve per raggiungere una particolare destinazione e quindi stampare a video le indicazioni delle strade da seguire e della relativa cartina stradale.

Questi e tanti altri sono i servizi che WebSphere può includere, ma oltre a queste soluzioni complete, è possibile utilizzare portlet creati per integrare, nelle pagine del portale, file di tipo HTML statico oppure pagine JSP dinamiche. Tali applicativi, integrati nei Tool di sviluppo, hanno lo scopo di agevolare gli sviluppatori nella realizzazione di nuovi servizi da integrare negli EIP aziendali. In particolare sono state create da IBM tipologie di licenza che permettono, ad eventuali aziende software, di utilizzare i Tool ed i portlet di WebSphere Portal Server per sviluppare soluzioni ad hoc capaci di interfacciare i sistemi proprietari dei propri clienti con il mondo web. WebSphere Portal Server si presenta quindi sia come soluzione standard pronta ad essere applicata all'azienda, ma anche come base di lavoro corredata da Tool e moduli software precostituiti per sviluppare agevolmente soluzioni orientate al web.

4.1.1.3 Information mining

Information Mining è un applicativo sviluppato da IBM ed integrabile in WebSphere Portal che fornisce agli utenti una modalità di accesso mirata alle informazioni, grazie ad un sistema automatizzato di estrazione ed analisi dei dati.

L'estrazione dei dati può avvenire da sorgenti eterogenee di dati grazie all'utilizzo della tecnologia *EIP Web Crawler* (vedi paragrafo successivo) e viene organizzata in modo da poter essere consultabile. Il processo di ricerca delle informazioni è guidato da algoritmi i quali possono gestire il recupero dei dati utilizzando sia le informazioni nei singoli documenti che modelli statistici orientati alle tipologie di argomenti. I modelli statistici hanno la capacità di individuare relazioni e correlazioni fra i documenti catalogati che potrebbero non sembrare ovvie all'utente. Per fornire questi servizi Information Mining nella versione 8.1 integra le seguenti funzioni:

- **categorizzazione**: capacità di assegnare ai documenti una categoria di appartenenza utilizzando tassonomie definite dagli utenti e strutture gerarchiche di classificazione degli argomenti.
    - **Information Structuring Tool (IST)**: una applicazione con interfaccia grafica orientata alla gestione delle tassonomie
    - **Category Assignment**: capacità di assegnare in modo automatico i documenti a determinate categorie attraverso l'uso delle tassonomie definite con IST
- **Summarization**: capacità di creare riepiloghi rappresentativi dell'intero documento, in modo tale che una breve letta possa aiutare il lettore a capire il contenuto del documento.

- **Language identification**: determina la lingua nella quale è scritto il documento.
- **Information extraction** : capacità di estrapolare le definizioni delle parole inserite nei documenti.
- **Clustering**capacità di raggruppare i documenti in base alle tipologie dei contenuti trattati.

4.1.1.4 Web crawler Service

La capacità di Web Crawler Service è di recuperare dati, documenti ed informazioni in intranet, extranet, o internet, nei database di LotusNotes©, nei file system locali o nelle collezioni di dati FTP. Gli utenti possono definire un seme o un gruppo di semi di partenza come documenti HTML dai quali partire per la ricerca e Web crawler Service, scorrerà lungo tutti i possibili link e trasferirà i dati trovati in cache o in sistemi di memorizzazioni per metadati. Tali contenuti possono poi essere processati da un EIP Informatio Mining o altri servizi di classificazione dei dati. Un problema di questo servizio è l'impossibilità di accedere, e quindi fornire all'utente tutti quei dati memorizzati in database e protetti da autenticazione. Tale limitazione impedisce l'accesso alla maggior parte di dati gestiti da ERP e quindi lascia una profonda incomunicazione fra applicativi eterogenei.

4.1.1.5 EIP Information Integration ToolKit

EIP Information Integratio ToolKit è un componente integrabile nell'offerta di WebSphere Portal orientato ai clienti ed ai partner IBM che intendono sviluppare servizi personalizzati da integrare nell'EIP. Il Tool in questione integra le tecnologie

- C++, Java™ ed integra ActiveX, JavaBeans
- Abilitazione all'accesso dei cataloghi prodotti di IBM
- Approcci strategici per lo sviluppo di applicazioni

La tecnologia integrata permette all'utente di sfruttare le potenzialità offerte da una programmazione orientata agli oggetti ("Object Oriented"), come la semplicità di gestione, l'estendibilità e la massima ricusabilità dei programmi creati.

4.1.2 Licenze di WebSphere Portal Server

Considerato le varie tipologie di utenti ai quali è destinabile il prodotto WebSphere Portal, risulta interessante analizzare come uno stesso applicativo possa essere venduto e con quali diverse tipologie di licenze. A scopo rappresentativo si è quindi scaricata l'offerta di IBM WebSphere Portal Express V4.1 dal sito ufficiale

dedicato agli EIP di IBM http://www-3.ibm.com/software/data/eip) . La versione per Windows di WebSphere Portal Express viene così venduta:

WebSphere Portal Express Intranet per Windows V4.1. Questo prodotto comprende l'infrastruttura di portale, le funzioni di personalizzazione, una selezione di portlet e WebSphere Application Server. Il prezzo del prodotto si basa sul numero di utenti con incrementi di un utente. Il prodotto è concesso in licenza per l'uso da parte di utenti di portale che lavorano per l'azienda.

WebSphere Portal Express Extranet per Windows V4.1. Questo prodotto comprende tutte le funzionalità di "WebSphere Portal Express Intranet per Windows V4.1" e il suo prezzo si basa sul numero di processori. Il prodotto è concesso in licenza per l'uso da parte di utenti di portale che non lavorano per l'azienda.

WebSphere Portal Express Intranet Plus per Windows V4.1. Questo prodotto comprende tutte le funzionalità di "WebSphere Portal Express Intranet per Windows V4.1" e in più quelle offerte da Lotus Quickplace e Lotus SameTime per gruppi di lavoro. Il prezzo del prodotto si basa sul numero di utenti con incrementi di un utente. Il prodotto è concesso in licenza per l'uso da parte di utenti di portale che lavorano per l'azienda.

WebSphere Portal Express Extranet Plus per Windows V4.1. Questo prodotto comprende tutte le funzionalità di "WebSphere Portal Express Intranet Plus, V4.1". Il suo prezzo si basa sul numero di processori. Il prodotto è concesso in licenza per l'uso da parte di utenti di portale che non lavorano per l'azienda.

4.2 BEA: WebLogic Portal

La soluzione all'integrazione dei servizi e all'accesso semplificato alle informazioni aziendali offerta da [BEA] si traduce in BEA WebLogic Portal™. Come dichiarato sul sito ufficiale italiano, [BEA] rappresenta oggi il maggiore fornitore mondiale di infrastruttura applicativa. Oltre 13.000 clienti nel mondo utilizzano le soluzioni di BEA per costruire, integrare ed estendere le applicazioni aziendali.

BEA WebLogic Portal nasce allo scopo di interconnettere processi eterogenei, informazioni strutturate, informazioni non strutturate e persone che partecipano nell'ambito di uno stessa catena di business, BEA WebLogic Portal è l'EIP sviluppato e commercializzato da BEA.

4.2.1 BEA WebLogic Architecture

Bea mette a disposizione ai propri utenti una piattaforma di e-business perfettamente integrata e completa, che comprende un portal framework con servizi base di gestione del portale, gestione di personalizzazione e interazione, amministrazione intelligente e servizi di integrazione che si appoggiano ad uno strato

centrale che funziona da Server. I principali componenti che compongono la piattaforma di BEA sono quindi:

- **BEA WebLogic Server (WLS)**: Come scritto da Jose Annunziato e Stephanie Fesler Kaminaris nel libro "Java Server Pages" WebLogic Server è il server di applicazioni attualmente più diffuso al mondo ed è l'unico server indipendente a essere certificato J2EE dalla Sun Microsystem

WebLogic Portal include in particolare quattro principali aree funzionali così suddivise:

- **Portal Foundation Service** : che offre un insieme di funzionalità di base di alto livello per semplificare sviluppo, manutenzione e sicurezza di portali complessi, massimizzando l'efficienza complessiva delle risorse tecnologiche ed informative.
- **Personalization and Interaction Management:** migliora l'esperienza utente e mette a disposizione i vantaggi di un framework per definire e misurare l'interazione utente, per raggiungere il successo attraverso incentivi mirati o meglio applicazioni e proposte dinamicamente presentate ai dipendenti, partner e clienti a seconda della loro esperienza vissuta nel portale.
- **Intelligence Administration**: questo componente ha la capacità di ridurre il backlog amministrativo ed il costo di ownership del portale, rende veloce l'accesso degli utenti alle risorse di cui hanno bisogno e migliora la capacità di adattare il portale a nuove esigenze di business.
- **Integration Service:** utilizzano un approccio basato su standard per ridurre i costi di integrazione del portale e per sfruttare i Web Services per l'integrazione delle applicazioni in tutta l'azienda e oltre. In ambito aziendale lo scopo di questi componenti è di interconnettere fra loro i sistemi di gestione delle diverse aree funzionali aziendali come gli enterprise resource planning (ERP), i Suply chain management (SCM) ed i customer relationship management (CRM) .

4.2.1.1 BEA WebLogic Server (WLS)

Come tutte le applicazioni Server anche BEA WebLogic Server fornisce l'ambiente necessario alla creazione di applicazioni distribuite. La Java 2 Enterprise edition (J2EE), è la piattaforma Java per lo sviluppo di applicazioni servers. WebLogic Server rispetta questa architettura multilivello, già descritta nei precedenti capitoli, ed include le seguenti APIs per la generazione di:

- EJB 2.0: componenti distribuiti,
- JSP 1.2 e Servlet 2.3 : per la generazione di pagine dinamiche
- JMS 1.0.2 per la gestione dei messaggi
- JDBC 2.0 per l'accesso ai database,
- JNDI 1.2 per la gestione dei naming

- J2EE connector Architecture 1.0
- JTA 1.0.1
- Java RMI 1.0
- RMI/IIOP 1.0
- JAAS 1.0
- JavaMail 1.1
- JAXP 1.1

Oltre alle caratteristiche architetturali della piattaforma multilivello, è importante sottolineare come, a partire dalla versione 6.0, BEA WebLogic Server dispone di un'utile console per la gestione delle operazioni di configurazione del server. Questa console di gestione solleva l'amministratore del server alla modifica diretta dei file di configurazione attraverso il prompt dei comandi, e gli fornisce una comoda interfaccia per la gestione di tali dettagli.

Scendendo nel dettaglio dell'installazione, si nota come la struttura della home directory di WebLogic Server contiene varie sottodirectory:

- JDKxx
- Logs
- Utils

La directory JDKxx contiene il Java Development Kit versione xx che, a partire dalla versione 6 di WLS, viene incluso nel pacchetto di vendita del server BEA. Le directory di logs e util contengono rispettivamente file log e di utilità. La directory di installazione del Server contiene le seguenti directory:

- Bin
- Config
- Ext
- Lib
- Samples
- Uninstaller

La directory bin contiene i file eseguibili. La directory config è particolarmente importante, perché si tratta della directory di configurazione per tutti i domini. Se per esempio durante la prima installazione del server, si accetta di configurare il dominio di default proposto dal server, mydomain, si troverà allora una sottodirectory col nome di mydomain nella quale si potranno inserire tutte le varie pagine del dominio. Le directory ext e lib contengono una serie di file jar di utilità per il sistema. La directory samples contiene una serie di esempi utili per fare pratica nell'utilizzo del server, e la directory uninstaller contiene il sistema di disinstallazione del software. La directory mydomain contiene a sua volta la directory applications in cui ritrova la directory defaultWebapp_myServer, che corrisponde al server di default. Questa è la directory di default per file HTML, e JSP destinati all'uso con Weblogic Server.

4.2.1.2 BEA Foundation Service

Considerato che l'EIP rappresenta per l'azienda la punta dell'iceberg dove applicazioni, contenuti e processi di business vengono unificati per fornire una visione completa della funzionalità del business, il portale deve soddisfare le esigenze di utenti, progettisti, sviluppatori, amministratori contabili, dirigenti in quanto deve fornire un unico punto di accesso a tutte le informazioni che, in un qualche modo, intervengono nella catena di business aziendale. Portal Foundation Service fornisce un insieme di funzionalità di base che investono le necessità di qualsiasi impresa, rispettando i requisiti di affidabilità e scalabilità che dovrebbero accomunare gli applicativi aziendali. I servizi offerti da questo componente sono la base necessaria all'integrazione di servizi multipli e rappresenta quindi l'infrastruttura atta a gestire il layout delle pagine, il posizionamento dei servizi sulle home page personalizzate, decidere i dimensionamenti dei frame, gli skin dei frame e tutto ciò che gestisce le personalizzazione dei portlets. Oltre a questi servizi di personalizzazione, vengono venduti con questo componente portlets in grado di integrare servizi commerciali con cataloghi, gestione degli ordini, che rappresentano comunque un ambiente abbastanza standard per qualsiasi tipologia di azienda.

Bea WebLogic Foundation Service non si limita solo a servire applicazioni di base ma è studiato per fornire agli utenti comode console per la creazione di nuovi portlets in grado di essere programmati per interfacciarsi anche con servizi precostituiti. Queste potenzialità sono integrate grazie all'uso di pubblicazioni con Java Server Pages (JSP) ed interfaccie per la gestione dei contenuti Service Provider Interface (SPI) che uniti permettono progettare pagine in real time.

4.2.1.3 BEA Personalization and Interaction Management

Per generazioni, il successo o il fallimento del business è stato legato alla qualità delle relazioni con i clienti. In modo particolare il trend degli ultimi anni ha mostrato come il successo del prodotto aziendale fosse direttamente proporzionale alle conoscenze che l'azienda produttrice aveva nei confronti dei clienti e delle loro esigenze. Anche attualmente è compito delle imprese capire l'esigenze dei propri clienti, ma a differenza di anni fa, il mercato si è esteso ed i clienti sono aumentati. Il portale risulta, secondo la visione di BEA, il punto nel quale interpretare ed assecondare le esigenze del cliente. Personalization and Interaction Management è un servizio che viene integrato nel portale di BEA che permette una personalizzazione del portale basata su regole di tipo implicite ed esplicite.

La regole di personalizzazione implicite adattano il portale al fine di misurare le esigenze del visitatore sconosciuto che naviga nel portale.

Le regole di personalizzazione esplicite adattano il portale alle esigenze degli utenti che vengono riconosciuti grazie alla fase di login.

In dettaglio, sull'articolo: "personalization and Interaction management" riportato sul sito ufficiale di BEA, viene riportato l'esempio di personalizzazione di una

fittizia azienda di nome Avitek. Nel caso riportato in figura si nota che le regole di personalizzazione esplicita sono state utilizzate per identificare il visitatore, il suo profilo ed i suoi attributi, i quali vengono utilizzati per esempio nel portlet delle news per pubblicare solo gli articoli che riguardano la sua area di business. Durante la navigazione dell'utente, il portale applica comunque le regole di personalizzazione implicita al fine di catturare eventuali altre nozioni che riguardino l'utente.

Figura 24: -portale personalizzato realizzato con BEA WebLogic –

Le regole di personalizzazione implicite ed esplicite risultano fondamentali per la soddisfazione dei clienti, dei dipendenti, dei partner e di qualunque utilizzatore del portale in quanto questo dimostra quanto l'impresa conosca le particolari esigenze di ciascun utente. Tali esigenze possono poi essere utilizzate dall'impresa per raggiungere in modo immediato i propri obiettivi. Per esempio queste regole, applicate al mondo del business to consumer, possono essere utilizzate per indirizzare le promozioni ai clienti, ma in modo altrettanto utile tali regole potrebbero essere utilizzate dall'ufficio personale per indirizzare corsi di formazione ai propri dipendenti. Come evidenziato nella figura sopra sono quindi utili anche quei frame di incentivazione e di promozione delle iniziative.

4.2.1.4 BEA Intelligence Administration

BEA WebLogic Portal™ include una capacità amministrativa intelligente di progettazione del portale (chiamata Intelligent Administrtion) mirata alla riduzione dei costi amministrativi di gestione del portale. La riduzione dei costi di gestione avviene grazie alla delega di alcune mansioni di gestione a processi automatizzati e ad una netta separazione degli incarichi gestionali.

L'automazione degli accessi, per esempio, è un servizio integrato nel portale che permette di settare i diritti di accesso alle risorse integrate nelle singole pagine o ai singoli portlets del portale. I parametri per i diritti di accesso possono essere gestiti da una semplice interfaccia grafica chiamata "E-Business control center" (EBCC) che permette di definire i diritti di accesso alle funzionalità del portale in base a determinati gruppi di utenti.

Delegare o decentralizzare l'amministrazione del portale permette invece sia di ridurre il carico di lavoro dovuto ad una gestione centralizzata che autorizzare gli utenti a gestire direttamente i loro bisogni. L'utilizzo della decentralizzazione amministrativa porta alla creazione di amministratori paralleli in grado di gestire la distribuzione del carico di lavoro delle singole sezioni del portale. La decentralizzazione comporta quindi una suddivisione degli incarichi ed uno sviluppo di servizi scomposto in moduli più piccoli e semplificati.

La suddivisione degli incarichi è visibile attraverso comode interfacce grafiche che permettono inoltre di visualizzare l'architettura del portale ed il flusso di navigazione da seguire per ottenere determinati servizi.

Lo scopo di Intelligence Administrator è quindi quello di rendere il portale altamente scalabile senza appesantirne la gestione di amministrazione anche a fronte di una notevole espansione dei servizi.

4.2.1.5 BEA Integration Service

La necessità di integrare sistemi monolitici software, come gli ERP e permetterne l'utilizzo attraverso gli EIP, ha spinto BEA a migrare i pesanti processi monolitici in componenti modulari integrabili nei servizi del portale. Allo socpo di integrare tali moduli nel portale, BEA WebLogic Portal fornisce Web Flow editor: una applicazione visuale per la definizione della navigazione nel sito e dell'interazione dei visitatori con i processi aziendali attraverso l'utilizzo dei componenti modulari. Altro servizio offerto da questo Tool è la possibilità di definire, fra vari componenti (detti pipelined components), sequenze di avanzamenti paralleli che interagiscono con i servizi aziendali e comunicazioni fra i vari portlets. I pipelined components possono essere raggruppati in pipelined session, lasciando comunque piena flessibilità all'utente di interrompere una sessione lavorativa senza interrompere il lavoro del singolo componente. Mentre il web flow permette quindi agli utenti di cambiare il flusso della navigazione nel sito, indipendentemente dalla logica dei

processi, i pipelined component permettono di modificare le logiche dei processi indipendentemente dalla navigazione intrapresa.

Oltre al beneficio sopra descritto, un approccio modulare permette di estendere i processi aziendali grazie ai servizi offerti da WebLogic Integration. Tale estensione promuove l'utilizzo di standard supportati dal WebLogic Server come J2EE Connector Architecture (J2EE CA), Web Service support for Universal Description, Discovery and Integration (UDDI), Web Services Description Lenguage (WSDL), e Simple Object Protocol (SOAP) . L'integrazione con i sistemi aziendali è semplificata grazie alla J2EE CA la quale è stata progettata per abbassare i costi operazionali utilizzando adattatori di integrazioni modulari.

4.3 SAP: MySap Portal

Considerata la dimensione e l'importanza del ruolo assunto dalle soluzioni informatiche fornite da SAP, si è pensato fondamentale analizzare anche la proposta di questa azienda nell'ambito degli Enterprise Information Portal.

Per accedere a qualsiasi tipo di sorgente informazionale disponibile nelle imprese, i portali Sap hanno integrato quattro differenti tecnologie che rappresentano i quattro pilastri fondamentali dei SAP Enterprise Portal. Nella tabella riportata in Figura è possibile individuare, per ognuna delle quattro tecnologie integrate nei portali SAP, la soluzione sviluppata da SAP.

Information Pillars	**MySap.com Enterprise Portal Solution**
Transaction System / Legacy Database	Iview & Unification
Internet	Yahoo
Content & Documentation	Knowledge Management
Data Warehousing & Analytical Processing	Business Warehouse

Figura 25: -I quattro pilastri dei portali SAP –

(Fonte: "mySap.com Enterprise Portal Cookbook" rapporto tecnico scritto da: Jude Lobo)

A livello funzionale mySap.com Enterprise Portal è scomponibile in tre parti fondamentali:

- **Portal Platform**: fornisce un ambiente per lo sviluppo e l'amministrazione dei contenuti del portale, permettendo di inserire i contenuti in diverse finestre di visualizzazione chiamate Iview che possono essere assemblate all'interno del portale. Le principali soluzioni

fornite da MySap.com Enterprise Portal Platform sono proprio dedicate alla gestione e creazione delle principali strutture del portale.

- **Iview Tecnology**: è la tecnologia che permette di creare ed amministrare le iView le quali possono essere sviluppate da SAP o personalmente dagli utenti. Le iView possono essere programmate in qualsiasi linguaggio come Java, JSP, ASP, XML, COM, etc…
- **Unification Tecnology**: permette di accedere in modo uniforme a varie applicazioni aziendali, permettendo di relazionarle fra loro o con altre applicazioni extraziendali grazie alla presenza di R/3 Unifiers.
- **Page Built tecnology**: per progettare e costruire pagine HTML da inserire nel portale
- **User Management Technology**: per l'amministrazione degli utenti, al fine di mappare gli ID in direttori LDAP[4] (Lightweight Directory Access Protocol). Al fine di abilitare singoli login (finchè gli utenti posseggono differenti ID per ciascuna applicazione aziendale)
- **User Role Management Technology:** Per creare portali preformattati a seconda dei ruoli professionali ricoperti dai singoli utenti nelle mansioni aziendali.

- **Knowledge Management Platform**: fornisce accesso ad una organizzazione di dati non strutturati appoggiandosi sulle seguenti funzioni:
  - **Content Manager:** gestisce l'intero ciclo di vita di un documento, compresa la definizione dell'autore, della collocazione di memorizzazione, e della sua visualizzazione.
  - **Retrivial and classification:** Al fine di migliorare le ricerche dei documenti, questa funzione si preoccupa di classificare automaticamente i documenti strutturati e non.
  - **Collaboration:** tale servizio mira a colmare il gap comunicazionale fra diversi gruppi di lavoro. Per far questo mette a disposizione degli utenti sistemi sincronizzati come chat, conferenze on-line, e sistemi non sincronizzati come forum di discussione, dischi condivisi etc.. per scambiarsi idee, opinioni e collaborare nei processi.

[4] (Lightweight Directory Access Protocol). Proposto nel 1993 adottato dalla Ietf (Internet Engineering Task Force), il protocollo Ldap si posiziona come uno standard universale per l'accesso alle directory: Un protocollo Internet che consente l'accesso via browser a rubriche in cui sono elencate tutte le risorse di rete, compresi i nomi di gruppi di destinatari per la posta elettronica. Viene usato come interfaccia standard per rendere accessibili da Internet le diverse directory (liste di risorse e di persone) gestite dai vari sistemi operativi di rete e per scambiare informazioni tra queste ultime.

- **Business information Warehouse Platform**:
  - **Data Warehousing:** fornisce visualizzazioni comuni dei dati aziendali e mette a disposizione comodi tools per la progettazione e la gestione dei dati nel warehouse aziendale.
  - **Reporting and Analysis:** fornisce tools per la visualizzazione e l'interrogazione dei dati al fine di supportare l'utente nel suo compito decisionale.
  - **Planning and Simulation:** fornisce un ambiente per la simulazioni di scenari aziendali.
  - **Portal Integration and Information deployment:** funzioni per semplificare l'integrazione di servizi nel portale.
  - **Business Performance Management:** fornisce tools per la rappresentazione analitica e visuale degli scenari aziendali, al fine di monitorarli e seguirne gli andamenti.
  - **Analytical Applications**: si basa su diverse aree di business integra processi di business e fornisce predefiniti scenari e misurazioni al fine di confrontare e correggere i processi attualmente in atto.

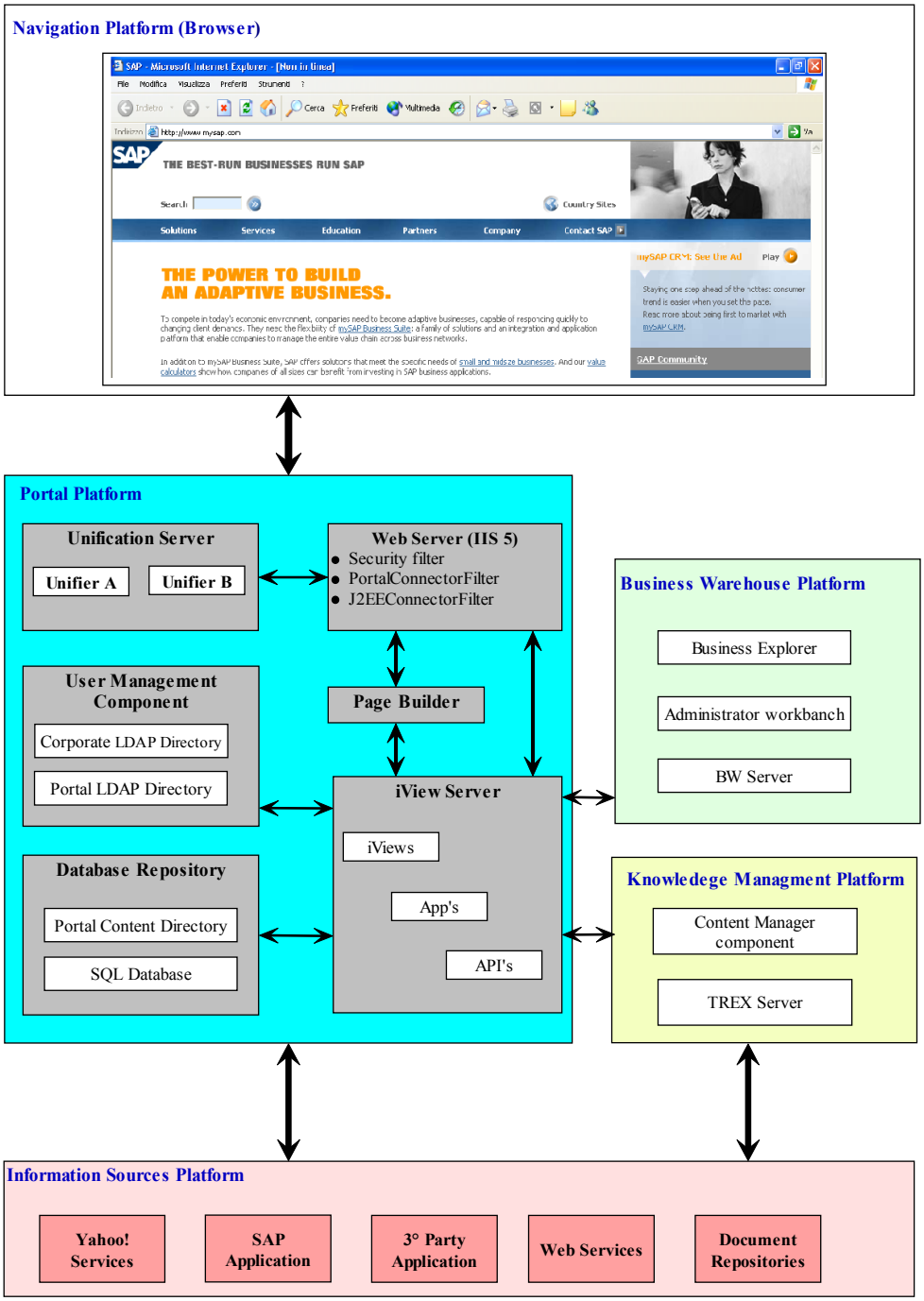


Figura 26: -SAP.com Enterprise Portal Function Architecture –

4.3.1 MySap Portal Architecture:

Come qualsiasi altra architettura distribuita vista fino ad ora, è possibile individuare due strati fondamentali: una piattaforma di navigazione definita "*Navigation platform"* ed una piattaforma vera e propria del portale chiamata "*Portal Platform"*. Considerato che la piattaforma di navigazione è costituita da un client con un browser standard che comunica con il portale attraverso richieste di tipo http o HTTPS, si è preferito studiare approfonditamente la Portal Platform la quale si suddivide in due livelli fondamentali:

4.3.1.1 Portal Platform: Middle layer

- **Web Server**: Microsoft Information Server (IIS 5.0) è il web Server utilizzato da MySap.com Enterprise Portal. In particolare il portale viene allocato in un host virtuale di nome 'SAPPortal' memorizzato, all'interno di un sottodirettorio dedicato del Server, di nome ….\Inetpub\wwwroot. Il Web Server venduto nel pacchetto di Sap , si arricchisce delle Internet Server Application Program Interface (ISAPI) , fornite ed installate dal programma MySap.com Enterprise Portal Setup Wizard che guida l'installazione del pacchetto SAP. Le principali ISAPI installate sono:
  - **Security Filter**: Il file securityfilter.dll installato, si preoccupa di controllare se gli utenti sono già stati autenticati nel portale. Se questi non hanno ancora eseguito il Login viene invocato un metodo di autenticazione definito BASIC, il quale chiede l'immissione di un nome-utente ed una password che saranno poi inoltrati al Corporate DirectoryServer (CDS) o ad un'altra autorità abilitata alle autenticazioni.
  - **Portal Connector Filter**: Il file PortalConnector.dll si preoccupa di capire se le richieste eseguite dagli utenti sono da inoltrare a specifici componenti del portale e nel caso appropriato reindirizzano tali richieste al componente dedicato
  - **J2EEConnectorFilter**: I J2EEConnectorFilter filtrano le richieste e controllano se queste devono essere processate dal J2EE engine ed in caso affermativo, le reindirizzano sulla porta J2EE che di default è la 8080.
- **Page Builder**: E' un servizio scritto in COM che estende le funzionalità del Web Server (IIS 5.0). Page Builder, come dice la parola stessa è un componente che permette la creazione delle pagine del portale, nelle quali vengono integrate e gestite varie finestre di visualizzazione (portlets) definiti da SAP come iView. Per far questo il Page builder suddivide il lavoro in due fasi:
  - **Step1:** carica inizialmente una pagina detta Portal Page nella quale non vengono inseriti gli iView, ma vengono posizionati i vari frame.
  - **Step2:** carica i singoli iView:
    - **passando per l'iViewServer** nel caso in cui l' iView aggiunga uno script ad un frame collocato sulla portal page iniziale. In questo caso i contenuti dell'iView sono reindirizzati al server il

quale si preoccupa della gestione dei contenuti e delle visualizzazioni al client.
        - **Bypassndo l'iViewServer** nel caso in cui l'iView aggiunga un URL al frame del portal page iniziale
- **Iview Server:** è un servizio sviluppato in COM che estende le funzionalità del Web Server (IIS 5.0); come visto sopra la sua funzione è quella di prelevare e fornire (fetch, catch) i dati degli iView al portal page builder. Nel MySap.com Enterprise Portal esistono due tipologie di iView:
    - **.NET iView,** solitamente scritte in ASP con un output getito in 'busdoc' o XML.
    - **Java iViews,** tipicamente scritti in Java/JSP. Iview Server dispone di un J2EE Application Server Engine (chiamato In-Q-My) che interpreta i Java iViews.

Il principale servizio fornito dall'iView Server è quello di processare le richieste degli iView provenienti dai client, memorizzarle in una cache, al fine di ottimizzare le prestazioni del portale, gestendo comunque le singole personalizzazioni degli utenti.

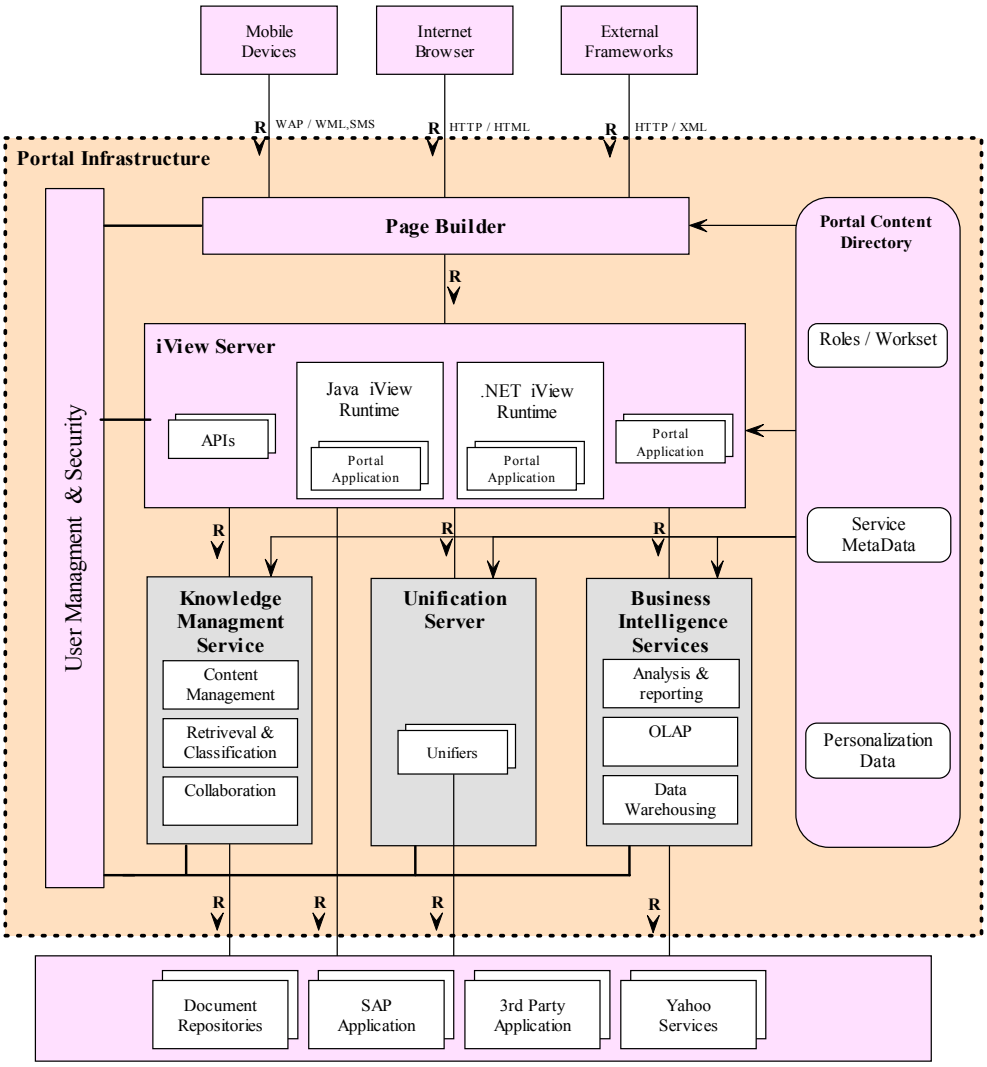


Figura 27: - MySap Portal Architecture –

Fonte: "mySAP Tecnology Portal infrastructure: people-Centric Collaboration" SAP WhitePaper

4.3.1.2 Portal Platform: Persistent layer

- **User Management**: Il componente in questione utilizza due tipologie di contenitori dati per immagazzinare i dati utenti. Entrambi i direttori sono basati sul Lightweight Directory Access Protocol (LDAP).
    - **Corporate LDAP Directory**: è un servizio sviluppato per recuperare le informazioni dell'utente e contiene:
        - Basic User Data
        - Basic Group Data
        - User/Group Assignment

  - Group Hierarchy

Qualsiasi utente che possiede un account nel LDAP aziendale, può accedere tramite log-in al portale.

- **Portal LDAP Directory**: è un servizio sviluppato per immagazzinare dati relativi agli utenti che vengono però riutilizzati solo dal portale, i dati contenuti sono:
  - Portal-Related User Properties
  - Portal-Related Group Properties
  - User/Group Properties
  - User/Group Role assignment

Queste informazioni possono essere immagazzinate in rami separati del Corporate LDAP directory server oppure su un dedicato LDAP corporate directory server.

- **Repository Database**:
  - **SQL Database**: possono essere utilizzati per immagazzinare i seguenti oggetti:
    - Roles
    - Worksets
    - Page Structures
    - Ipanel Definitions
    - Iview Data
    - User Personalization Information
  - **Portal Content Directory (PCD)**: sono direttori dedicati che funzionano da contenitori di dati persistenti. Anche in questi sottodirettori è possibile salvare informazioni riguardo i seguenti oggetti:
    - Roles
    - Worksets
    - Page Structures
    - Ipanel Definitions
    - Iview Data
    - User Personalization Information

    I Portal Content Directory sono utilizzati anche per mantenere memorizzati i dati riguardanti i role editor, e quindi mantengono informazioni come:
    - Role Mantenance
    - Role Migration (da sistemi esterni)
    - Import / Export of Role in objects
    - External Service migration

4.3.2 Flusso delle informazioni in mySAP.com Enterprise Portal

Come descritto nel SAP Technical Delivery intitolato "mySAP.COM Enterprise Portal Cookbook" scritto da Jude Lobo e disponibile in rete al sito www.sapgenie.com il flusso delle informazioni che vengono visualizzate dal portale segueno il seguente percorso:

1. Il web Browser (mySAP.com Enterprise Portal Client) manda un richiesta al server Web.
2. Il Web Server (con l'ausilio delle ISAPI) blocca la richiesta per l'autenticazione e la reindirizza al SecuityFilter.dll che invoca la Basic Autentication[5]. Il nome-utente e la password vengono spedite dal client al Corporate Directory Server for Autentication. Il PortalConnector.dll controlla che la richiesta sia una specifica del portale e la ridireziona al J2EE ConnectorFilter[6] per controllare se la richiesta debba essere processata da un J2EE engine. In tal caso la richiesta viene reindirizzata alla porta della J2EE che di default è la porta 8080.
3. Il PageBuilder, che rappresenta una estensione del Server Web, assembla la Portal Page iniziale e posiziona un iFrame per ogni singola funzione di iView.
4. L'iView Server riceve la richiesta dal Page Builder e richiama l'iView Application attraverso il suo URL, l'iView application legge i dati da una sorgente persistente e li processa e li spedisce il risultato all'iView Server mantenendo in cache una copia dei contenuti inviati.Compito dell'iView Server è anche di aggiungere uno script o un URL al relativo iFrame della Portal Page iniziale.
5. Il WebServer fornisce infine all'utente un FrameSet (cioè una parte di Frame) che gli è stata restituita dal Page Builder ed un biglietto di LogOn. A questo punto il WebBrowser mostra la pagina ricevuta e pubblica i dati ricevuti dall'iViewServer.

[5] Vedi capitolo 8 paragrafo 8.2 per approfondire la Basic Autentication.
[6] Vedi capitolo 7 paragrafo 7.4.3 per la configurazione degli Isapi filter.

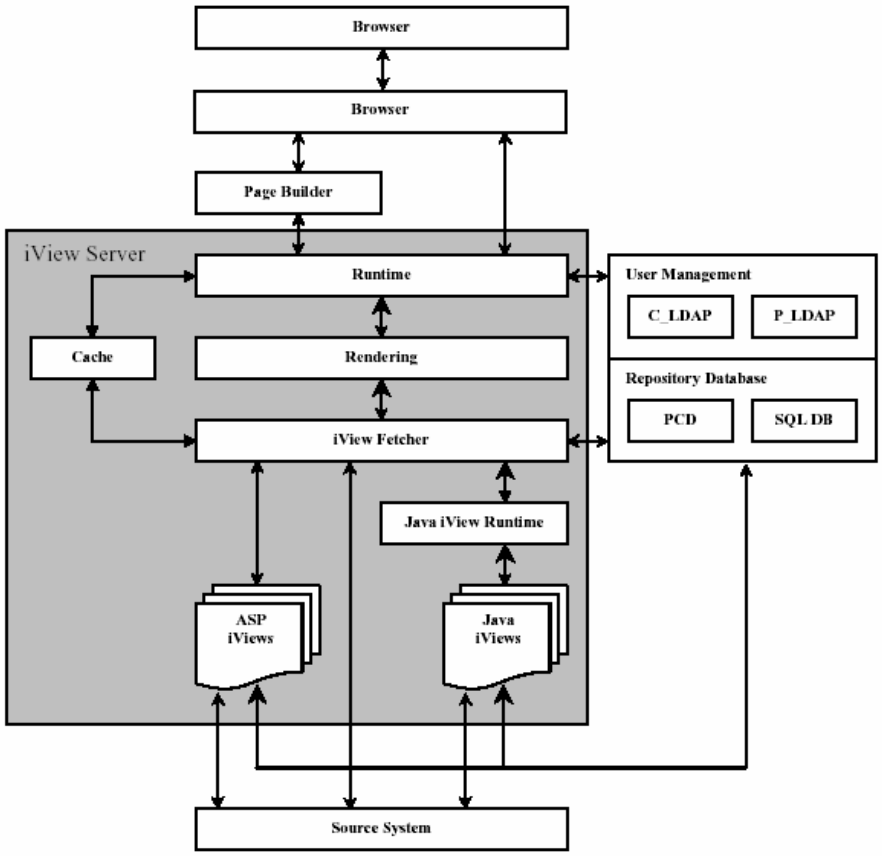


Figura 28: - Flusso delle Informazioni nel mySap Portal-

Fonte: "mySAP Tecnology Portal infrastructure: people-Centric Collaboration"

SAP WhitePaper

4.3.3 Prodotti inclusi in mySAP.com Enterprise Portal

A differenza della convenzione fin qui utilizzata e condivisa dalle aziende analizzate, per cui ci si è riferiti con il termine Enterprise Information Portal a tutta la tecnologia atta a creare un portale in grado di rappresentare un unico punto di accesso ai servizi ed alle informazioni di una struttura aziendale, Sap si differenzia sostituendo il termine EIP con quello più generico Enterprise Portal (EP). L'Enterprise Portal di Sap viene fornito in tre possibili versioni che prendono nomi specifici a seconda della loro specializzazione:

- mySap.com Enterprise Information Portal: Portale che fornisce informazioni agli utenti attraverso i servizi offerti dal portale Yahoo!. Tale applicazione è stata resa possibile grazie ad un contratto stipulato fra SAP e Yahoo!

- mySap.com Enterprise Collaboration Portal: Portale che fornisce agli utenti informazioni/applicazioni/servizi a seconda del ruolo ricoperto.
- MySap.com Enterprise Unification Portal: Portale che fornisce agli utenti informazioni/applicazioni/servizi a seconda del ruolo ricoperto ma dà la possibilità di raggiungere dati gestiti da sorgenti eterogenee permettendo di relazionarli fra loro.

4.4 Confronto fra i prodotti analizzati

L'analisi fatta evidenzia come i prodotti venduti dalle principali aziende mondiali produttrici di soluzioni informatiche nell'ambito degli Enterprise Information Portal, condividano una stessa base architetturale. La piattaforma J2EE rappresenta ormai per tutte le soluzioni web based uno standard de facto, grazie a benefici quali la scalabilità, e la flessibilità che si possono sfruttare grazie all'uso di un approccio J2EE oriented . Nella tabella riportata è possibile visualizzare con quali prodotti IBM, SAP e BEA riproducano una architettura simile alla J2EE ed in quale modo vengano integrate applicazioni capaci di estendere le funzionalità dei relativi Server Web in modo da creare una infrastruttura adatta allo sviluppo di un Enterprise Information Portal. Mentre SAP offre ai propri clienti il server di Microsoft (IIS 5.0), BEA ha sviluppato un intero server proprietario, ed IBM ha esteso le potenzialità di Tomcat, un Server Web ed http distribuito in modalità Open Source dal gruppo Jakarta. Altra particolarità da non sottovalutare è che IBM integra nella propria offerta anche il proprio database DB2.

Confronto fra le Architetture	IBM WebSphere	BEA WebLogic Portal	SAP mySAP.com Portal
Server	Apache Tomcat modificato da IBM	Bea WebLogic Server unico server attualmente certificato dalla Sun J2EE	Microsoft Information Server (IIS 5.0)
EIP FrameWork	Apache Jetspeed modificato da IBM	fornisce una soluzione proprietaria completamente realizzata da BEA	SapPortal Platforma soluzione proprietaria
Storing Data/Database	Integra DB2		

Figura 29: -Confronto Architetturale dei prodotti –

Se la base architetturale dei prodotti analizzati risulta simile, i servizi offerti dai vari EIP risultano invece differenziati in funzione della clientela a cui le aziende rivolgono i loro prodotti.

	Confronto fra i Servizi	IBM WebSphere	BEA WebLogic Portal	SAP mySAP.com Portal
TOOL di Gestione e Sviluppo	**Tool di amministrazione Server**		Console per la configurazione del Server	
	Tool di amministrazione Portale e/o Utenti	• Integrato in WebSphere Portal Server • WebSphere Personalization	• BEA WebLogic Foundation Service • Intelligence Administration	• User Managment Technology • User role managment Technology
	Tool di Sviluppo	EIP Informatio Integration Toolkit (studiato anche per i partner IBM)	BEA WebLogic Foundation Service	Page Built technology
TOOL di Gestione e Sviluppo	**Knowledge Managmnet**	• Information Mining (IBM) • WebCrawler Service (IBM) • EIP Information Integration(IBM)		Content & documentation
	Business Intelligence		Personalization & Interaction Managment	Business Information Warehouse Platform
	Accesso a sistemi proprietari (ERP);(SCM);(CRM)		Integration Services	Businness Warehouse
	Accesso a contenuti e/o servizi Web	Vengono forniti portlets configurabili per l'accesso a servizi/pagine remote	Integration Services	Contratto con Yahoo!
Soluzioni standard	**Applicazioni/Portlets Standard**	WebSphere integra una serie di applicazioni standard pronte all'uso		

Figura 30: -Confronto dei servizi offeri dai prodotti analizzati –

La soluzione proposta da IBM presenta un tool di gestione di amministrazione del portale e degli utenti ma quello che la caratterizza è una serie di tool avanzati per lo sviluppo di nuovi servizi. WebSphere Portal Server è un prodotto di tipo infrastrutturale, in grado di fornire ad eventuali terze parti, (dette partner) potenti applicativi che permettono lo sviluppo di nuovi portlet in grado di interfacciarsi anche con altre tipologie di prodotti, eventualmente proprietari. A fianco di questi potenti tool di sviluppo IBM offre una serie di servizi, indipendenti da qualsiasi altra struttura software, mirati allla gestione dei contenuti ed offre una piena integrazione con i prodotti venduti da Lotus Corporation.

Anche mySAP Portal offre ai propri clienti console dedicate all'amministrazione del portale e degli utenti, ma il vero valore aggiunto risiede nei potenti software di content aggregation che lavorano in piena sintonia ai warehouse aziendali e integrano le competenze di aziende come Yahoo! per la gestione dei contenuti dislocati nel

Web. Per raggiungere una tale integrazione di servizi con i warehouse aziendali, è chiaro che SAP ha concepito il portale cercando di fornire una sorta di continuità alle soluzione software già sviluppate come l'ERP. SAP vuole fornire un prodotto capace aprire al web le proprie soluzioni informatiche e quindi rivolge mySAP Portal al proprio parco clienti.

In più rispetto alle soluzioni di IBM e SAP BEA WebLogic Portal integra una console di amministrazione che include, oltre alla gestione del portale e degli utenti, anche la possibilità di configurare attraverso comode interfacce grafiche anche il server. La console di gestione di BEA è concepita inoltre con un approccio a delega come descritto nei paragrafi precedenti, il che rende il portale scalabile oltre che dal punto di vista architetturale anche dal punto di vista amministrativo. La soluzione proposta da BEA integra avanzati servizi di Business Intelligence e Work Flow del portale, studiati per fornire agli utenti mirate soluzioni ed incentivi a seconda delle esperienze da loro sostenute nel portale ed analizzate in background da regole implicite ed esplicite. La soluzione di BEA è mirata ad uno studio del navigatore del portale ed è quindi molto adatta per portali dedicati ai clienti.

Capitolo 5

Analisi e scelta delle tecnologie di Jetspeed: un particolare EIP OpenSource

Oltre alle soluzioni sviluppate e vendute dalle aziende presenti sul mercato, è possibile realizzare portali dinamici utilizzando anche software gratuiti scaricabili dalla rete: Jetspeed è l'implementazione di un Enterprise Information Portal scritto completamente in java ed XML sviluppato e distribuito, in modo open source, dal gruppo di ricerca internazionale di volontari uniti nell'Apache Software Foundation's Jakarta Project [*JAKARTA*]. Jetspeed, come un qualsiasi altro EIP, crea la rete di connessione fra risorse di tipo eterogeneo, e le rende accessibili agli utenti tramite il portale, il quale risulta accessibile sia via Web-Browser che attraverso la tecnologia WAP-phone. Jetspeed funge quindi da fulcro centrale (detto central hub) dove le informazioni, provenienti da sorgenti eterogenee, vengono organizzate e rese disponibili agli utenti.

Il dato che viene gestito attraverso Jetspeed è completamente indipendente dalla tipologia del contenuto, questo significa che i dati gestiti in XML, RSS[7], o SMTP disponibili anche su risorse esterne come database, web services o altro, possono venire integrati nei portali sviluppati con Jetspeed. L'attuale gestione del dato avviene attraverso l'uso del linguaggio XSL e la relativa pubblicazione sul portale viene gestita attraverso l'uso combinato di Java Server Pages (JSPs) e pagine HTML. Per la pubblicazione dei contenuti i portlets si avvalgono degli Element Construction Set (ECS) API che generano degli elementi di markup a partire da oggetti java. Gli ECS supportano il Wireless Markup Language (WML), l' HTML ed anche l'XML e tutto questo è integrato nella licenza d'uso di jetspeed scaricabile dal sito di Jakarta. Per la gestione e la pubblicazione dei contenuti risulterebbe più facile l'utilizzo di servlet-base template o web pubblishing come WebMacro, Velocity o Cocoon, scaricabili anch'essi in modo open source dal progetto Jakarta Apache. Jetspeed offre quindi dei Portal API per sviluppare piccole applicazioni in java, chiamate portlets, le quali trovano integrazione e funzionamento all'interno del portale.

[7] RSS è un particolare formato di XML usato per gestire i titoli delle pagine web. I costruttori di pagine web pubblicano queste tipologie di titoli per permettere agli utenti di ricercarli ed individuare i contenuti che gli interessano, una volta individuati i titoli interessanti, questi permettono un link al contenuto totale degli articoli.

5.1 Apache Jetspeed Portal Architecture

Come molti dei programmi sviluppati dal progetto Apache, anche Jetspeed è costruito sopra Turbine, una applicazione open source integrata nel file di download la quale si occupa dell'autenticazione degli utenti e del layout delle pagine. I portlets possono interagire con i servizi di Turbine attraverso i *RunData* object, mentre i sistemi integrabili o inclusi come Java Server Pages, Cocoon, Velocity e Web Macro forniscono i layout per le applicazioni dinamiche attraverso l'uso di XML/ XSLT e XSP nel caso di Cocoon.

Jetspeed può appoggiarsi su numerosi servlet runners e database ma la configurazione consigliata dagli sviluppatori dell'Apache project e quindi scelta anche in questa tesi, è di installare come servlet runner Tomcat 4.1 ed utilizzare come database Thomas Mueller's Hypersonic SQl, già unito nel pacchetto di Jetspeed. Il fatto che Tomcat 4.1 possa funzionare anche come Web Server, permette di non installare ulteriori Server HTTP.

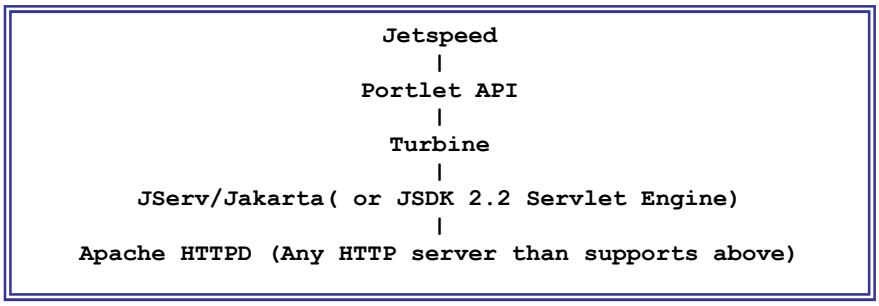


Figura 31: - Architettura di Jetspeed -

(http://jakarta.apache.org/jetspeed/site/application-development.html)

5.1.1 Portlet API

Le Jetspeed Portal API sono un insieme di interfacce che descrivono come un portlets deve interagire con il portlet container ossia con una Jetspeed web application. Le specifiche dei portlet vengono definite dalle Portlet Specification [JSR 168] e servono per definire i diversi componenti dei portali, le loro interazioni, i loro cicli di vita e la loro semantica. Ogni portlet che viene sviluppato con Jetspeed deve implementare direttamente l'interfaccia: org.apache.jetspeed.portal.Portlet oppure estendere una classe che la implementa. Grazie all'estensione di questa classe i portlets possono fruire di funzioni predefinite come accesso alle informazioni dei singoli utenti, interagire nelle portal windows, accedere alle informazioni dei web client, condividere informazioni con altri portlets, e metodi standard di memorizzazione delle informazioni in modo persistente. Proprio come le Servlet

API, le specifiche dei portlet permettono l'accesso ai sistemi di gestione aziendali senza imporre restrizioni sulle tipologie di protocolli utilizzati.

I portlets sono quindi speciali sottoclassi dell'HttpServlet, con proprietà che gli permettono facilmente di inserirsi ed agire nel contesto di un portale. I portlets possono essere assemblati all'interno di una vasta pagina anche con istanze multiple di uno stesso portlet ma con contenuti differenti a seconda della loro inizializzazione. Le portlet API forniscono interfacce standard capaci di dare servizi e di gestire in modo autonomo ogni portlet, in modo indipendente dall'intera infrastruttura del portale.

Le portlet API sono quindi un'estensione delle servlet API tranne il fatto che l'azione di alcune funzioni viene ristretta all'ambito del portlet e, dove è necessario, nel contesto del portale. Per vedere in dettaglio quali API vengono comunemente usate per la gestione di un portlet, si consiglia di proseguire nei successivi paragrafi.

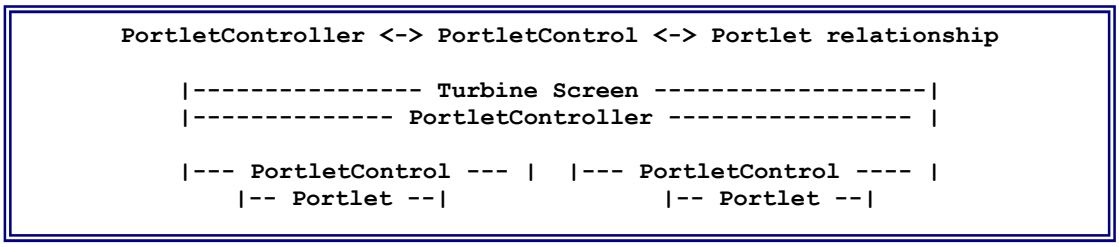


Figura 32: - Page Layout -

(http://jakarta.apache.org/jetspeed/site/application-development.html)

5.1.2 Portlet Interface

I frammenti di codice generati dai singoli portlet attraverso l'uso del metodo *getContent(),* vengono aggregati da Jetspeed il quale li inserisce nel contesto del portale. L'interfaccia del Portlet risulta essere il compromesso fra lo sviluppatore del portlet e le specifiche di Jetspeed, le quali condizionano i comportamenti e le interazioni dei singoli portlet nel contesto del portale.

Media Type

Il contenuto generato dal portlet dipende dai Media Type supportati dal browser. Il browser o più in generale il dispositivo che si connette al portale fornisce, nelle request, dei parametri che descrivono le caratteristiche del dispositivo stesso. Per esempio, un web browser fornisce nelle request dei parametri che descrivono quali MIME Types e linguaggi supporta. Il package Jetspeed recupera tutte queste informazioni e le racchiude in un oggetto java di nome JetspeedRunData. Uno

sviluppatore di applicazioni con Jetspeed può recuperare tali informazioni attraverso la classe *CapabilityMap* associata alla request del dispositivo: rundata.getCapabilityMap e controllando i media Type supportati.

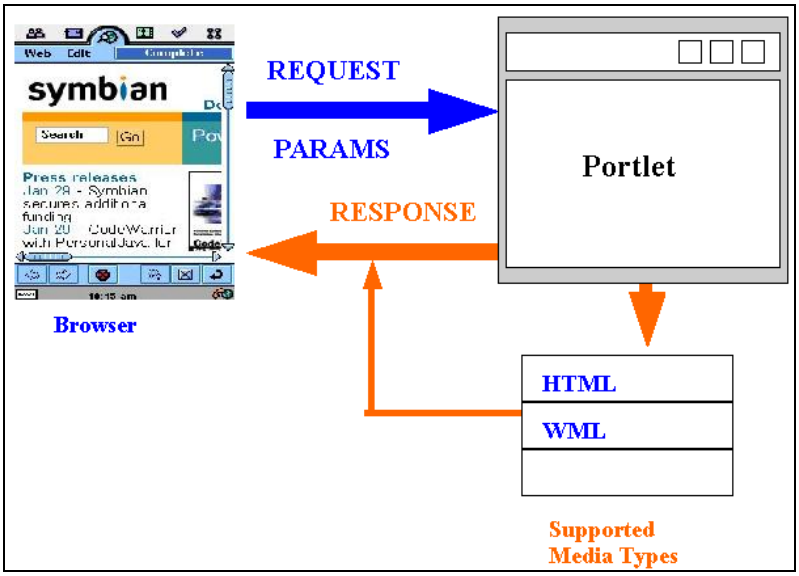


Figura 33: - Gestione dei Media Type – Fonte: [Tutorial] di Jetspeed

Nella Figura sotto viene riportato uno stralcio di codice che mostra come poter reperire attraverso la portlet Interface i media type supportati durante la generazione dei contenuti nel metodo *getContent()*.

```
  public ConcreteElement getContent(RunData rundata)
    {
        JetspeedRunData jrun = (JetspeedRunData) rundata;

        CapabilityMap map = jrun.getCapability();

        StringBuffer text = new StringBuffer();
        String mimeType = map.getPreferredType().toString();

        if (this.supportsType(map.getPreferredType()))
        {
            text.append("Supports preferred MimeType: " +
mimeType);
        }
        else
        {
            text.append("Doesn't support preferred MimeType: "
            + mimeType);
        }
    }
```

Figura 34: Codice di Esempio - Recupero Media Type supportati dal browser -

Ogni portlet definisce nel registry i media type che il dispositivo deve supportare, per poter visualizzarlo. La configurazione del portlet nel registry deve quindi definire dei tag <media-type>:

```
<portlet-entry name="HelloPortletInterface"
              hidden="false" type="instance" application="false">
        <meta-info>
            <title>Hello Portlet Interface</title>
            <description>JPortal Tutorial - Hello Portlet
Interface
             </description>
        </meta-info>

<classname>com.bluesunrise.jportal.portal.portlets.HelloPortletIn
terface
</classname>
        <parameter name="version" value="1.4b3" hidden="false"/>
        <media-type ref="html"/>
        <media-type ref="wml"/>
        <category>tutorial</category>
        <category>portlet</category>
    </portlet-entry>
```

Figura 35: Codice di Esempio - Configurazione dei media type nel file registry del portlet -

Attualmente i media-type supportati nel regstry sono quattro:

- HTML
- WML
- XML
- VXML

Ciclo di vita di un Portlet

Tutti i portlet possono essere automaticamente creati all'avvio di Jetspeed e posti nella cahe dei portlet: per far questo basta settare nel file principale di configurazione *jetspeedResource.properties*: `Autocreated.portlets=true`

```
########################################
# Automatic Portlet Creation           #
########################################
# Jetspeed can automatically create/instantiate all your Portlets
and #place them in the cache when Jetspeed starts up.

autocreate.portlets=false
```

Figura 36: -Configurazione di Automatic Portlet Creation-

Il comportamento che viene raccomandato dagli sviluppatori di questo EIP e quindi quello assunto di default dal programma, è di non creare il portlet finché questo non è acceduto da una PSML page, e quindi porre `Autocreated.portlets=false`

Risulta che, di default, il portlet viene creato ed inserito nella portlet cache, solo nel momento in cui viene richiesta una pagina nella quale è incluso. Al momento della creazione del portlet, viene chiamato il metodo *init()* e questo metodo è chiamato una sola volta durante l'intera vita di quel determinato portlet.

Il ciclo di vita (*life cycle*) di un portlet dipende da quanto tempo rimane nella cache, infatti la cancellazione del portlet dalla cache rappresenta l'effettiva eliminazione di questo dalla memoria del server e quindi non risulta più accessibile. Nel caso in cui un portlet venga rimosso dalla cache e poi successivamente richiesto da una pagina, allora verrà richiamato il relativo metodo init() il quale genererà nella cache una nuova istanza del portlet. Sia gli amministratori di sistema che i programmatori possono controllare il ciclo di vita dei portlet definendo il tempo di permanenza di questi nella cache.

Un portlet è gestito attraverso un ciclo di vita non definito e durante questo occorre decidere come inizializzare il portlet, come deve trattare e gestire le richieste dei contenuti e quale durata base di vita deve avere. Queste caratteristiche individuano le tre fasi fondamentali di un portlet:

1. Init
2. Render
3. Destroy

Oltre a queste tre fasi, Jetspeed ne supporta una ulteriore che non si manifesta direttamente sull'interfaccia del portale ma che si colloca fra la fase 1 e la fase 2 o meglio fra l'inizializzazione del portlet e la gestione dei contenuti e prende il nome di *processAction phase.* Le azioni dovrebbero essere gestite dalle action class le quali sono ereditate da Turbine, ma si infiltrano all'interno dell'architettura di jetspeed attraverso una accoppiamento non molto elegante fra Jetspeed e Turbine. Velocity portlets cerca di retificare questo difetto di architettura ma fondamentalmente la situazione non è cambiata e quindi la action processing phase non fanno parte delle interfacce dei portles (tratteremo in modo più approfondito questo problema nel paragrafo dedicato ai Velocity Portlet).

Possiamo quindi collegare le fasi con i loro metodi di interfaccia:

PHASE	METHOD
Init	Init
ProcessAction	TurbineAction
Render	GetContent
Destroy	--none--

Figura 37: - Fasi di un portle e relativi metodi -

Durante la fase Init, viene raccomandato di gestire qualsiasi tipologia di inizializzazione del portlet. Solitamente questo include anche la necessità di settare tutte quelle connessione ai database e a risorse che richiedono tempi di accesso

costosi, e a tutti quei parametri che, oltre ad essere condivisi, assumono uno stesso valore fra le varie istanze. La fase di render è chiamata su richiesta, ogni volta che viene richiesto il contenuto del portlet allora viene chiamato il metodo getContent .

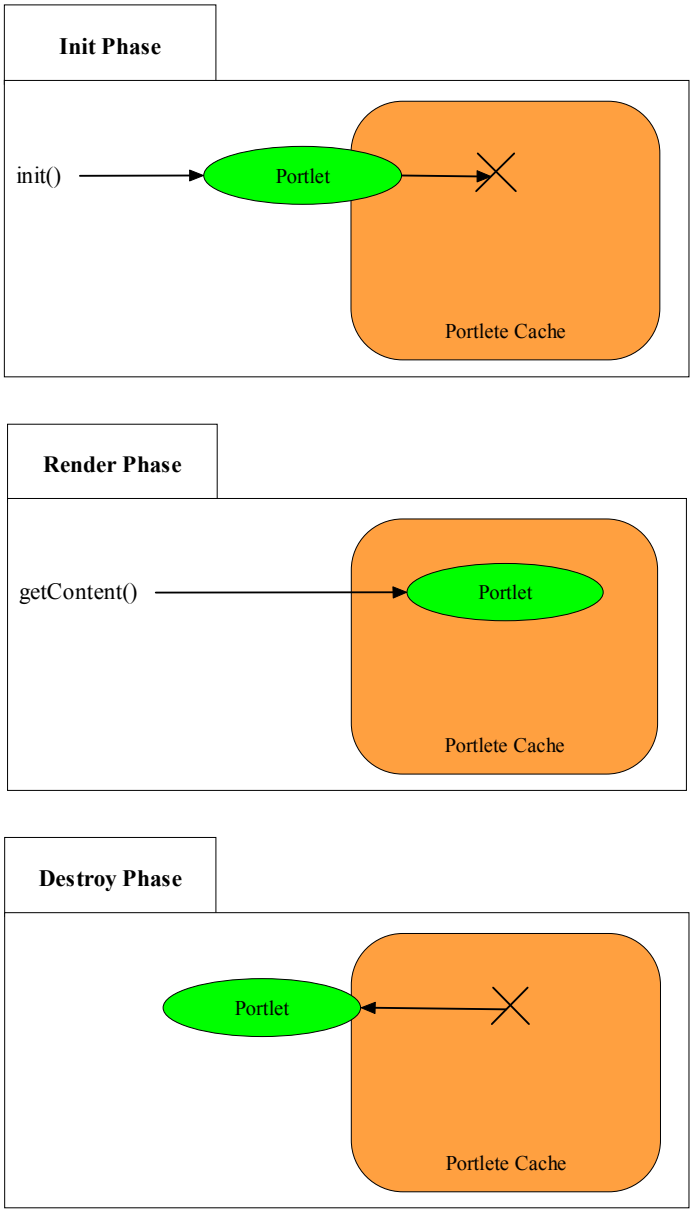


Figura 38: -Fasi di un portlet-

La fase di destroy dipende dalla durata del ciclo di vita del portlet. Sfortunatamente al portlet non viene mai notificato il momento in cui avverrà la sua cancellazione dalla cache e quindi la sua distruzione. Considerato che di default il portlet eredita dalla classe AbstractPortlet, risulta allora gestibile il ciclo di vita del

portlet, definendo nel *global portal setting* (file principale delle properties) il numero di millisecondi che il portlets può rimanere in cache.

```
########################################
# Portlet Cache                        #
########################################
# TimeToLive.default = number of milliseconds an unused portlet
will remain in cache.
#  Default 2700000 which is 45 minutes (45 * 60 * 1000)

services.PortletCache.classname=org.apache.jetspeed.services.port
letcache.JetspeedPortletCacheService
services.PortletCache.TimeToLive.default=2700000
```

Figura 39: -Esempio di configurazione life cycle-

Ogni qualvolta Jetspeed attraversa la fase di inizializzazione di un portlet, per prima cosa controlla se questo è già presente nella cache. Tale controllo viene eseguito per assicurarsi che la gestione del portlet sia affidata ad un unico metodo avente la capacità di differenziare i sucessivi portlet gli uni dagli altri senza dover istanziare copie della stessa classe. In particolare come la classe AbstractPortlet implementa il metodo *getHandle()*, anche la classe *AbstractInstancePortlet* definisce un metodo capace di creare nomi univoci da assegnare ai portlet in modo da poterli poi individuare univocamente nel futuro. Lo scopo di questo approccio è di poter dupplicare i portlets e differenziarli tenendo traccia dei relativi parametri tutto questo istanziando una sola volta la classe di riferimento. In questo modo, anche se sul portale accedono migliaia di utenti, non occorre istanziare migliaia di home-page personalizzate, ma basta differenziargli i parametri chiamando i metodi visti: tutto queste rende Jetspeed un EIP altamente scalabile.

Parametri di Init, Attributi e Titoli.

La portlet Interface permette all'utente di settare due tipologie di configurazione delle variabili: una per il portlet e una per le sue istanze. I parametri per il metodo *init()* sono definiti attraverso un file di *registry*, mentre gli attributi per le istanze sono definiti in un file PSML. I parametri di Init sono associati a tutti i portlet di una data classe e sono solo leggibili (*read-only*), mentre gli attributi sono associati ad ogni istanza della classe del portlet e possono essere modificati e resi persistenti.

Se si permette la visualizzazione dei parametri definiti nel *registry* e cioè i parametri di Init, allora questi sono visibili e modificabili dall'area di personalizzazione del portlet (*customize portlet*), richiamabile attraverso il bottone customize action button  . In particolare nell'area di customizzazione è anche possibile unire e modificare sia i parametri del metodo Init che gli attributi dell'istanza ma quando si va a salvare le modifiche apportate, queste vengono salvate solo sulla pagina corrente memorizzata nel risorse PSML e non viene modificato il

file di registry. Tale procedimento, assicura alle prossime nuove istanze di venire settate con i valori dei parametri definiti nel registry, ma permette all'utente che li ha personalizzati di riutilizzare la configurazione modificata.

Portlet mode

Una delle caratteristiche offerte da Jetspeed è la possibilità di selezionare, attraverso la menù bar del portlet, una delle sei tipologie di visualizzazione del portlet:

- View
- Customize
- Print_Friendly
- Info
- Minimize
- Maximize

View mode: La modalità di view è la modalità di default che viene applicata da Jetspeed per la visualizzazione normale dei portlet.

Customize mode: Nella visualizzazione di Edit fornita di default da Jetspeed, viene data la possibilità all'utente di visualizzare, al posto del contenuto del portlet, i parametri che lo caratterizzano. Tali parametri vengono caricati dal file di registry, e una loro eventuale modifica, viene memorizzata nel file PSML che identifica l'istanza del singolo portlet. Il metodo che implementa la customizzazione si chiama provideCustomization() ed è definito nella classe AbstractPortlet .

Maximize mode: Tale modalità, visualizza lo stesso contenuto della visualizzazione View allargando la finestra di visualizzazione del portlet all'intera pagina del portale.

Print Friendly mode: Tale modalità, visualizza lo stesso contenuto delle visualizzazione View, ma vengono eliminati tutti quelli che sono i controlli che circondano il portlet.

Oltre alle visualizzazioni analizzate è possibile scegliere, dalla menù bar, anche le modalità di **Info**, **Close** e **Minimize**. La particolarità che differenzia queste dalle precedenti, è che durante tali visualizzazioni, non vengono eseguite chiamate dirette ai portlet interessati; per esempio nella modalità minimize, vengono rese visibili alcune informazioni di esecuzione, ma non i contenuti del portlet. Ognuna delle visualizzazioni è selezionabile dalla menù bar del portlet, la quale presenta in ordine da sinistra a destra le seguenti modalità di visualizzazione: customize, print friendly, Info, Close, Minimize e Maximize .

Access Actions		
Icon	**Action**	**Description**
N/A	view	Allows to select a portlet in customizer and view its contents
	customize	Allows to customize a portlet once selected in profile
	info	Allows to view any additional information about a portlet
	maximize	Allows to view portlet in full screen mode
	minimize	Allows to minimize portlet (hide its content) and display its caption only
	close	Allows to temporarily close a portlet (hide its caption and content)
	print	Allows to display current portlet in "print friendly format" (without navigation and portlet control). Note that the default screen template/layout used may be overriden by setting action.print.template property in jr.props to your custom screen template.

Figura 40: - Menù bar ed Icone del portlet-

Sviluppare un portlet con Jetspeed

Quando si sviluppa un nuovo portlet da inserire in un portale implementato con Jetspeed occorre utilizzare la classe base AbstractPortlet memorizzata nel package org.apache.jetspeed.portlets. Utilizzando tale classe è possibile ottenere un portlet funzionante implementando (cioè sovrascrivendo) il metodo getContent(), il quale a sua volta implementa l'interfaccia RunData di Turbine. Attraverso i metodi get e set messi a disposizione dell'interfaccia degli oggetti RunData, è possibile accedere alle informazioni di esecuzione gestite da Turbine, come accesso ai cookie, accesso alle informazioni degli utenti ecc…. Il metodo getContent() ritorna un oggetto ConcreteElement della classe ECS, la quale rappresenta una estensione dei tag HTML o WML i quali formatteranno il contenuto del portlet in modo da poter essere rappresentato con un web browser o con dispositivi Wap.

```
Package it.esempio.portal.portlets;

import org.apache.jetspeed.portal.portlets.AbstractPortlet;
import org.apache.turbine.util.RunData;
import org.apache.ecs.ConcreteElement;
import org.apache.ecs.StringElement,

public class HalloWorldPortlet extended AbstractPortlet {
 public ConcreteElement getContent(RunData runData) {

 StringBuffer text=newStringBuffer();
JetspeedRunData jrun = new jetspeedRunData;

        switch (jrun.getMode())
{
case jrun.NORMAL:
text.append("MODE = VIEW");
break;
case jrun.CUSTOMIZE:
text.append("MODE = CUSTOMIZE");
break;
case jrun.MAXIMIZE:
text.append("MODE = MAXIMIZE");
break;
default:
text.append("MODE = UNKNOWN");
break;
}

 return(text.toString());
}
}
```

Figura 41: Codice di Esempio - Portlet realizzato con uso di ECS -

Una volta compilato e posizionato il file HalloWorldPortlet.class all'interno del relativo package it.esempio.portal.portlets posizionato nella webapps directory di Tomcat relativa a Jetspeed (WEB-INF/classes/ it.esempio.portal.portlets /Halloworld.class), e configurato nel file di registry (vedi capitolo 7) , il portlet è pronto per essere caricato nel portale.

Aggiungere un portlet in una pagina del portale.

Quando un portlet è stato

- compilato,
- inserito nel webapps directory di Tomcat dedicata a Jetspeed
- e debitamente configurato nel file di registry

è pronto per essere incluso in un pagina di un utilizzatore. Supponiamo allora di avviare Jetspeed (vedi capitolo 7), ed accedere al portale come utenti registrati (come vedremo è possibile utilizzare inizialmente come username "turbine" e come

Password “turbine”) ed attivare, cliccando sulla icona a matita posta sulla menù bar, la modalità di customizzazione. A questo punto selezionare il pannello da modificare (che solitamente si chiama “home”) e premere il tasto “aggiungi portlet”. A questo punto verrà visualizzato un elenco di portlet disponibili (definiti nel registry) integrabili sulla home page dell’utente (vedi Figura sotto), selezionare HalloWorldPortlet, cliccare in sucessione i tasti “applica”, “salva e applica”, poi il tasto “applica” nuovamente. A tal punto verrà visualizzata la pagina di partenza (detta visualizzazione normal) con integrato il portale scelto.

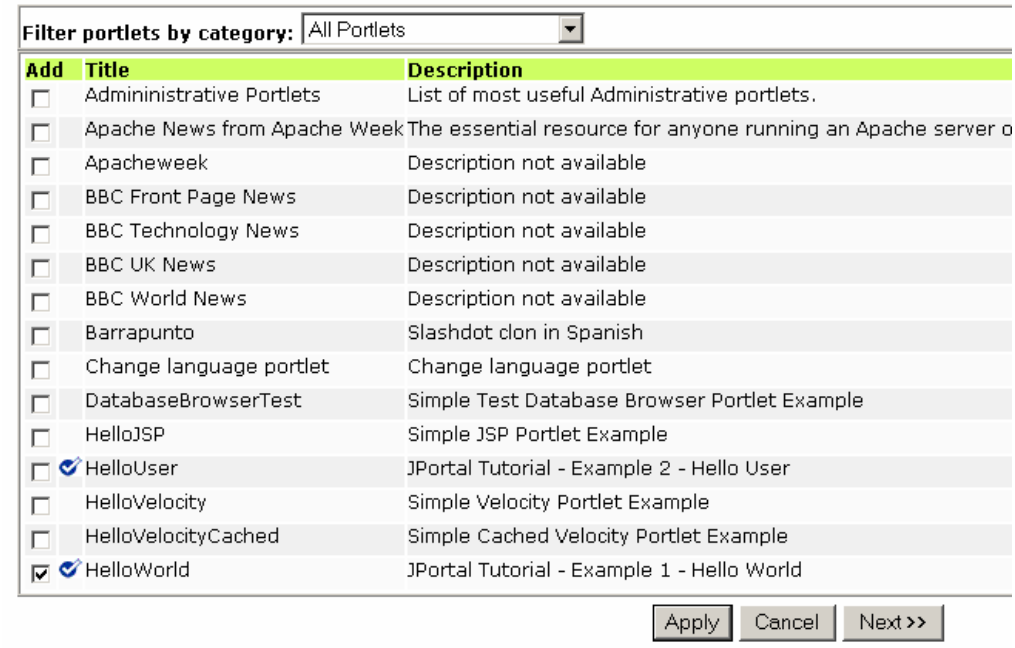


Figura 42: - Esempio di customize portlet (Aggiunta di un portlet) –

Quando viene inserito un portlet su una pagina di un utente, sul suo relativo file PSML viene aggiunta una entry, nella quale si assegna all’istanza del portlet aggiunta un identificativo chiamato *portlet instance id* che lo identifica da qualsiasi altro portlet del portale. Quindi si generano più istanze di una stessa classe (definita nel parent) ma vengono gestite indipendentemente le une dalle altre e le eventuali diversificazioni legate ai Layout o ad eventuali parametri modificabili attraverso la visualizzazione di customize (come il titolo del portlet) rimangono memorizzate sul file PSML relativo all’utente.

```
<entry id="P-f33cdc6793-10002" parent="IFrameController">
            <layout position="-1" size="-1">
                <property name="column" value="0"/>
                <property name="row" value="0"/>
            </layout>
</entry>
```

Figura 43:Codice di Esempio -Portlet New Entry nel file PSML -

5.1.3 Apache Turbine v.2.2

Turbine è un servlet based framework che permette agli sviluppatori java di costruire sicure applicazioni web cioè applicazioni nelle quali gli utenti possono utilizzare il loro Web Browser preferito per accedere in modo sicuro a servizi di *business logic*. Tutto il codice di Turbine è raccolto in un'unica locazione, suddiviso in moduli facilmente utilizzabili ed indipendenti fra loro, integrabili con altre applicazioni come Velocity, WebMacro ecc…. Anche questo codice, come Jetspeed viene rilasciato con la stessa licenza d'uso che permette agli utenti di creare utili siti web, senza preoccuparsi di modifiche apportate ai codici sorgenti. Turbine non è un application server, è un tool per costruire applicazioni web che può essere utilizzato in tre modi differenti

- Come un servlet framework con Turbine come controller
- Come un framework di comodo per il codice delle applicazioni
- Come un Object-Relational Tool

In ogni caso, basta unire alle API il codice fornito in turbine.jar file ed utilizzare le funzioni di cui si necessita.

Turbine è costituito di cinque moduli differenti ognuno dei quali svolge uno specifico compito nella struttura di Turbine. Allo scopoo di analizzare in dettaglio la struttura di Turbine , si analizzano questi moduli separatamente:

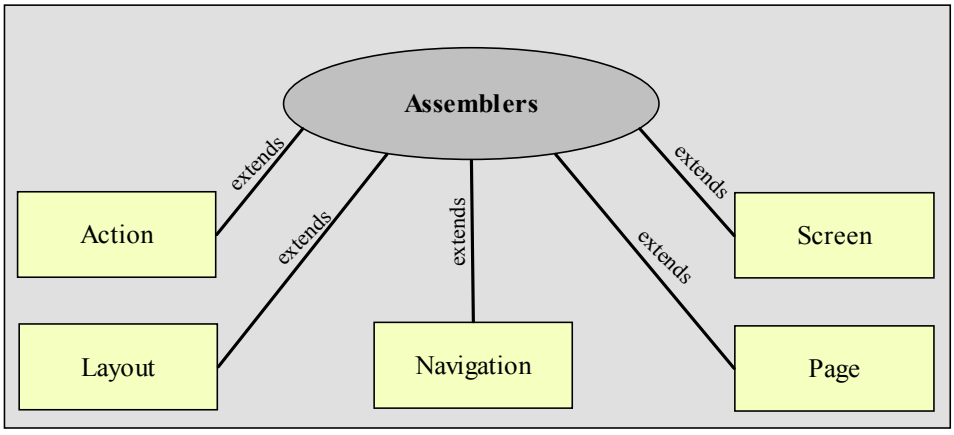


Figura 44: -Moduli di Turbine –

Fonte: "Turbine Specification" http://jakarta.apache.org/turbine

Action:

Il modulo Action rappresenta un insieme di istruzioni che svolgono mansioni precise e che possono essere chiamate fra la richiesta HTTP del client e la visualizzazione della pagina. Per esempio quando un utente esegue il submit di un form HTML, uno dei campi Hidden field (del codice HTML) contiene, in una variabile di nome Action, il nome di una procedura da chiamare alla quale vengono passate le informazioni raccolte nel modulo FORM appena compilato. Solitamente in questi casi la procedura andrà a confrontare i dati immessi nel form con i dati contenuti in un database e agirà a seconda degli eventi programmati. Il ciclo di istruzioni che avviene è il seguente:

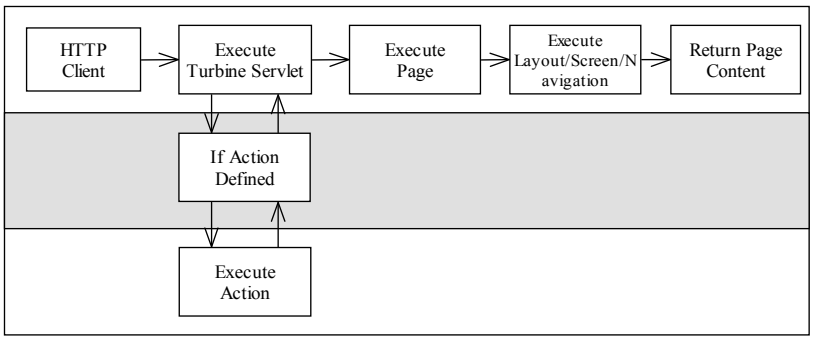


Figura 45: - Il funzionamento di Action –

Con questo metodo è possibile separare i processi da applicare ai dati spediti tramite una POST (e analogamente quelli di una GET) in moduli riusabili. Tale approccio permette di definire una sola funzione che viene richiamata tramite Action ogni volta che si necessita applicarla ai dati inviati tramite richieste http, è un metodo molto comodo per integrare procedure di business logic eseguite dagli EJB's in Turbine.

Page:

Il modulo Page è il primo modulo della catena che viene chiamato per generare delle Pagine. Il modulo Page controlla se esiste una Action definita nella richiesta e, in caso affermativo, tenta di lanciare tale procedura. Dopo l'esecuzione della Action il modulo Page si incarica di inoltrare la richiesta al Set Screen Object per il layout della pagina, e quindi tenta di eseguire il Layout object ritornato dal Set Screen Object. Importante è sottolineare il fatto che proprio il Set Screen Object è incaricato di leggere dal file *TurbineResource.properties* il valore di DefaultLayout per applicarlo al Layout.

Screen:

Il modulo Screen è essenzialmente considerato il "body" della pagina web. Il modulo di Layout esegue il modulo Screen. E' questo il modulo nel quale le pagine HTML vengono generate.

Navigation:

Un sito web presenta generalmente uno schema di navigazione che si divide in alto (top) e basso (bottom) i quale definiscono generalmente il titolo e le informazioni di chiusura della pagina Web. Il modulo di navigazione è eseguito da quello di Layout.

Layout:

Il modulo di Layout viene chiamato dal modulo di Page ed il suo compito è quello di definire il Layout fisico della pagina web. Generalmente vengono definite le localizzazioni delle barre di Navigation e della porzione di body (detta anche screen). Il modulo di Layout esegue il modulo Screen per costruire il body della pagina ed esegue il modulo di Navigation per costruire le porzioni relative alle barre di navigazione.

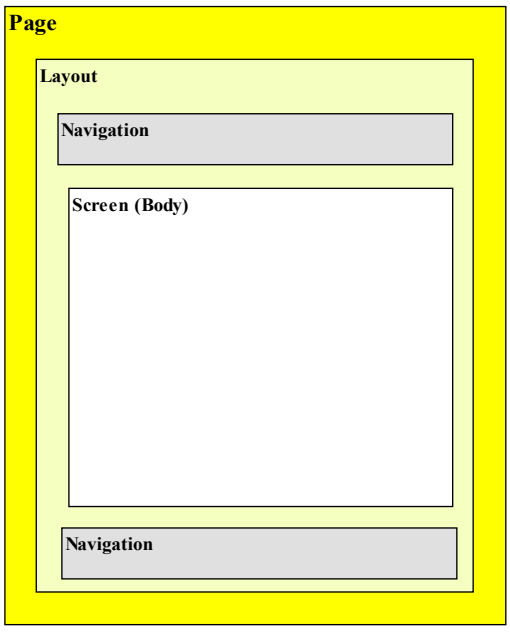


Figura 46: - Azione dei componenti su una pagina web -

5.1.4 Jetspeed e Turbine

Nel cuore di Turbine, si trova Turbine Servlet, il singolo punto di accesso alle applicazioni, e alle classi che formano il framework di interfaccia http. Queste sono le classi basilari che le applicazioni logiche devono estendere per inserire i framework necessari. Nel caso particolare Jetspeed estende le classi RunData e User fornite da Turbine e le rinomina rispettivamente come JetspeedRunData e JetspeedUser.

- **RunData interface:** Fornisce al programmatore l'accesso alle servlet request, state e ad altre informazioni riguardo la sessione, come il controllo degli accessi, le Turbine actions, cookies, servlet parameters (parsed) e all'utente corrente. Le informazioni che riguardano l'utente corrente sono accessibili attraverso i metodi forniti dalle classi getUser() o getJetspeedUser().
- **User interface:** Con un oggetto di tipo User è possibile accedere a tipiche informazioni degli utenti ed anche salvarle temporaneamente (per la durata della sessione) con i metodi getTemp() e setTemp(). I metodi da utilizzare per riportare informazioni riguardo gli utenti vengono riportati di seguito:

Metodi della classe USER	**Description**
AccessCounter	Tiene traccia di quante volte l'utente ha avuto accesso al portale.
Confirmed	Ritorna il valore restituito durante l'automatico sign up
CreateDate	La data di creazione di questo account
Email	Ritorna l'indirizzo e-mail dell'utente
FirstName	Ritorna il nome dell'utente
LastAccessDate	Ritorna l'ultima data di accesso dell'utente
LastLogin	Ritorna la data dell'ultimo login
LastName	Ritorna il cognome dell'utente
Password	Ritorna la password dell'utente
Perm	Memorizza in una hash table permanente nome/valore
Temp	Memorizza in una hash table Temporanea nome/valore
UserName	Ritorna lo UserName
hasLoggedIn	Indica se l'utente è attualmente loggato
incrementAccessCounter	Aumenta di uno il contatore degli accessi
isConfirmed	Indica se l'utente ha già dato conferma
removeTemp	Rimuove nome/valore dalla hash table

Proprietà della classe USER	Description
Disabile	L'account di un utente può essere disabilitato con l'uso di questa proprietà
UserId	È lo UserId
isNew	Indicatore booleano che indica se l'utente è già stato memorizzato o meno

La gestione del Layout e degli utenti sono i servizi principali utilizzati da Jetspeed per la creazione del portale e la gestione dei singoli portlets.

5.1.5 Servlet engine e Web Server : Apache Tomcat 4.0.6

Un portale sviluppato con Jetspeed, risulta essere a tutti gli effetti una applicazione web-based che necessita di un servlet engine per poter processare pagine ed applicazioni di tipo dinamico. A differenza dei prodotti venduti dalle aziende fino ad ora analizzate (IBM, BEA, SAP) che integrano già nel pacchetto di installazione il Servlet engine, Jetspeed viene rilasciato senza: spetta quindi allo sviluppatore eseguire le varie procedure atte all'installazione di un server capace di funzionare sia come Server Web che come Servlet Engine. Al fine di minimizzare i prodotti da installare, il consiglio fornito dagli sviluppatori di Jetspeed, è di installare L'EIP all'interno del server Apache Tomcat, applicativo Open Source in grado di processare sia richieste http (e quindi capace di funzionare come Server Web) sia di processare pagine di tipo dinamico (e quindi funzionare come Servlet Engine). Per configurare Tomcat in modalità stand-alone o in collaborazione con il server Microsoft IIS si consiglia di consultare il capitolo 7.

5.1.6 Database

Come descritto precedentemente, il file di Jetspeed include Thomas Mueller's Hypersonic SQL Database il quale risulta essere già configurato per funzionare integrato con Jetspeed e Tomcat. Nel caso si decidesse di seguire l'installazione di default e quindi di utilizzare Apache Tomcat come Server Web e come Servlet Engine, non sono necessarie configurazioni aggiuntive per integrare il database rilasciato. Viene comunque lasciata libertà agli sviluppatori di installare database differenti come Oracle, DB2, Sybase, MySQL o Postgres ma per configurarli occorre settare opportunamente il file TurbineResource.properties per specificare il posizionamento del server atto alla gestione del database.

5.2 Licenza di Apache Jetspeed

Jetspeed viene rilasciato in modalità OpenSource e questa è la licenza scaricabile dal sito: http://www.apache.org .

Figura 47: - Licenza di Jetspeed -

5.3 Scelta di Apache Jetspeed

Jetspeed è un Enterprise Information Portal ma a differenza dei prodotti forniti da IBM,SAP e BEA non possiede quei Tool amministrativi, Tool di sviluppo o servizi integrati capaci di rendere il portale facilmente configurabile, gestibile ed integrabile ad altri sistemi informativi. Gli unici servizi integrati in Jetspeed sono alcuni portlet, scaricabili come esempi dai tutorial, che non rappresentano assolutamente soluzioni pronte all'uso e quindi nemmeno commercializzabili. Jetspeed risulta essere quindi una solida base architetturale per un Enterprise Information Portal ma necessita di un grosso sforzo di progettazione e programmazione prima di poter integrare servizi ed applicativi capaci di dare un vero valore aggiunto agli utilizzatori.

Jetspeed può rappresentare un'opportunità per aziende di medio-grosse dimensioni produttrici di soluzioni informatiche proprietarie che vogliano aprire al web ed integrare in un EIP i sistemi informativi da loro prodotti, senza dover necessariamente avvalersi di un portale di terze parti e senza dover riprogrammare interamente le fondamenta architetturali di un EIP. Una scelta di questo tipo da parte di una azienda informatica, implica una prima fase di analisi di fattibilità del progetto, nella quale considerare oltre le potenzialità del laboratorio di ricerca e sviluppo interno, anche i tempi di realizzazione di una soluzione che possa alla fine rappresentare un effettivo prodotto concorrenziale capace di dare servizi diversificati rispetto le soluzioni fornite dai propri concorrenti. In una prima fase di sviluppo la realizzazione di soluzioni compatibili con le necessità del proprio parco clienti potrebbe rappresentare un modo per cominciare a commercializzare l'EIP come un servizio aggiuntivo alle soluzioni fino ad ora fornite.

Capitolo 6

Strumenti per la pubblicazione e la gestione dei contenuti

6.1 Velocity

Velocity è un tool opensource per la pubblicazione di pagine dinamiche anch'esso sviluppato dalla comunità di ricercatori volontari uniti nella Apache Software Foundation's Jakarta Project. Come definito sul sito ufficiale, [Velocity] è un java-based template engine. Permette ai disegnatori di pagine web di utilizzare metodi sviluppati con codice java. Grazie ad una integrazione completa dei servizi di template forniti da Velocity per le applicazioni di Turbine (web application framework), i disegnatori di pagine web possono lavorare in parallelo ai programmatori per sviluppare siti web, rispettando il Model View Controller [MVC].

L'architettura *Model-View-Controller (MVC)* costituisce un valido criterio di design di una applicazione J2EE per garantire la separazione tra funzionalità di business logic e accesso ai dati, dalla presentation logic. In particolare, risulta essere ideale per collocare gli oggetti individuati nella prima fase di object decomposition nei tre livelli logici di cui si compone, e consente ulteriori raffinamenti per arrivare a livelli sempre più vicini all'implementazione vera e propria:

- **Model** rappresenta i dati sui quali una applicazione si appoggia. Possiamo in linea generica identificare i model-objects con i business-objects di cui si è parlato a proposito della business logic. In un approccio EJB-Centric gli Enterprise Java Beans modellano questi oggetti e ne gestiscono gli aggiornamenti.
- **View** fornisce una rappresentazione dei dati contenuti nel Model in uno specifico formato. La View di una applicazione Web di tipo enterprise è, come si è detto, costituita principalmente da pagine dinamiche come le JSP. In particolare i dati modellati nel Model da ciascun EJB sono riflessi in un corrispondente JavaBean che ne fornisce la rappresentazione nel View consentendo alla JSP nella quale sono inseriti di visualizzarne il contenuto in modo semplice. E' importante sottolineare che tali JavaBean hanno compiti puramente di presentazione dei corrispondenti EJB, mentre a quest'ultimi spetta il compito di aggiornare i dati da essi incapsulati e gestire la business logic dell'applicazione.
- **Controller** fa da tramite tra il Model e il View, nel senso che si occupa di far sì che l'immagine presentata dal View sia coerente con i dati

contenuti nel Model e si occupa di convertire le interazioni che l'utente fa sulla View in corrispondenti eventi di aggiornamento sul Model. Dal momento che il Controller ha questo ruolo di coordinatore tra il Model e il View, esso è solitamente costituito da componenti che rientrano sia nel Web-tier che nel EJB-tier.

La potenzialità di Velocity è quindi quella di riuscire a separare il codice Java dalla pagina web permettendo così la creazioni di siti web facilmente manutenibili e quindi fornire una valida alternativa alle Java Server Page (JSP) o PHP.

Jetspeed include nel proprio motore (engine) i servizi di pubblicazione di Velocity, infatti molti dei controlli e controllori utilizzati per la creazione del layout del portale e di molti portlets è gestita da metodi di Velocity. Tale situazione permette di applicare il modello MVC all'intero sviluppo di un portale.

6.1.1 Introduzione all'uso di Velocity

Qualsiasi applicazione che si appoggia su Velocity richiede due parti: la prima è il template, che viene salvata con estensione .vm e la seconda corrisponde ad un programma sviluppato in java, salvato con estensione .java.

6.1.2 Velocity portlet

Un portlet sviluppato con la tecnologia di Velocity è composto da tipici componenti del modello MVC:

MVC Component	Velocity Component
Model	Java Objects put in the context
View	Template
Controller	Your Velocity action

Figura 48: -MVC Component & Velocity Component -

Il controller è rappresentato dalle classi di azione di Velocity (Velocity Action Class). La classe di base *VelocityPortlet* non necessita quasi mai di essere modificata. La Visualizzazione (view) è una pubblicazione (Templates) di Velocity, la quale genererà il contenuto del portlet, estraendo dinamicamente i contenuti dalle elaborazioni svolte dalle VelocityAction per inserirle poi nel contesto. Il metodo *getContent()* di un Portlet sviluppato con Velocity non necessita di essere modificato perché tutto il contenuto è generato dalla pubblicazione, proprio come progettato nel modello MVC.

Come specificato nel capitolo 5, in riferimento al ciclo di vita di un portlet, possiamo ora analizzare in dettaglio la fase chiamata ProcessAction, quella fase che si inserisce fra la fase Init e quella di pubblicazione nel ciclo di vita di un portlet.

PHASE	METHOD
Init	Init
ProcessAction	TurbineAction
Render	GetContent
Destroy	--none--

Figura 49: -Fasi di un portlet e relativi metodi-

I portlet sviluppati con la tecnologia di Velocity sono utilizzati solo per contenuti dinamici, non si ha infatti convenienza utilizzare tale template per lo sviluppo di pagine a contenuti statici.

Il funzionamento di base di Velocity consiste nel sostituire oggetti dinamici sviluppati con java, con Velocity template.

Per fare un esempio concreto riportiamo uno stralcio di codice prelevato dalla User Guide di Velocity (consultabile al sito: http://jakarta.apache.org/velocity/user-guide.html). Lo scopo di questo stralcio è costruire una pagina dinamica capace di visualizzare ad un paricolare utente gli abbonamenti da lui sottoscritti.

Una volta concordato fra tutti gli sviluppatori dell'azienda che:

- $customer si riferisce a tutte le informazioni pertinenti al cliente loggato,
- $mudsONSpecial si riferisce a tutte le tipologie di prodotti in sconto ,
- $flogger contiene metodi per il supporto alle promozioni,

è possibile implementare un pagina dinamica includendo questo statement VTL nella pagina web:

```
<HTML>
<BODY>
Hello $customer.Name!
<table>
#foreach( $mud in $mudsOnSpecial )
   #if ( $customer.hasPurchased($mud) )
      <tr>
        <td>
          $flogger.getPromo( $mud )
        </td>
      </tr>
   #end
#end
</table>
```

Figura 50: Codice di Esempio - Statement VTL-

6.1.3 Introduzione al Velocity templates language (VTL)

Lo scopo del Velocity Template Language (VTL) è di fornire una metodologia semplice e pulita per incorporare contenuti dinamici all'interno di pagine web. VTL utilizza *refernces* per inserire contenuti dinamici nel sito; anche le variabili rappresentano una tipologia di references. Le variabili utilizzate in questo linguaggio possono riferirsi ad oggetti definiti nel codice java oppure possono impostare il loro valore in base ad altri VTL statements. Per esempio il comando

#set($a = "Velocity")

inizia con il carattere # come tutti gli statement VTL ed imposta il valore di una variabile ad una costante utilizzando la direttiva di *set*. Tale direttiva utilizza una espressione racchiusa fra parentesi ( ) per assegnare un valore *value* ad una variabile *variable*. La variabile viene inserita nella parte sinistra dell'espressione mentre il valore ad essa assegnato occupa la parte destra ossia stà alla destra del segno di uguaglianza, utilizzato per separare variabile e valore. Nell'esempio sopra riportato, la variabile è $a mentre il valore è Velocity.

Quando un visitatore richiede la pagina web, il motore di pubblicazione di Velocity scorrerà all'interno della pagina web alla ricerca di tutti i caratteri #, e determinerà quali fra le linee di codice che iniziano con tale simbolo sono statement VTL e quali no.

Importante è ricordare che:

- **References** iniziano con $ e sono utilizzate per riferirsi a qualcosa
- **Dirctive** iniziano con # e solitamente compiono un'azione.

Sempre in riferimento all'esempio: *#set* è una directive e viene utilizzata per assegnare un valore, mentre, $a, può essere usata nella pubblicazione per mettere in output la costante "Velocity".

Una volta che ad una variabile è stato associato un valore, è possibile richiamare tale variabile in qualsiasi parte della pagina HTML, ma è anche possibile accedere a tutti i suoi metodi pubblici se questa variabile è un oggetto java. Al fine di creare pagine contenenti direttive VTL leggibili, viene consigliato ,dal manuale di Velocity, di iniziare tali direttive sempre a partire da una nuova linea di codice.

Velocity fornisce anche un insieme di comode direttive per effettuare controlli logici. Fra le direttive più comunemente utilizzate ricordiamo *#if* e la *#foreach*. Nell'esempio sopra riportato viene utilizzato il *#foreach* per scorrere ciclicamente un array, mentre *#if* viene utilizzato per richiamare un metodo pubblico e controllare quali abbonamenti ha sottoscritto il cliente. Al termine del controllo il nome dell'abbonato viene visualizzato in una tabella insieme ai suoi abbonamenti ed i suoi attributi.

6.1.4 Velocity Portlet in Jetspeed registry

Come ogni altro portlet, anche i portlet sviluppati con la tecnologia Velocity, devono essere integrati nel file di *registry* affinché possano essere visibili all'interno dell'EIP.

Quando si definiscono i Velocity portlet, occorre definire due parametri fondamentali: i template e gli action:

- i template definiscono le pubblicazioni di velocity le quali genereranno i contenuti del portlet (e devono essere salvati su file system in una sottodirettorio del Velocity paths)
- le action sono controllori, detti *controller,* che hanno la responsabilità di gestire le azioni, gli eventi e populating the context (e devono essere salvate in moduli convenzionali posizionati in un sottodirettorio dei portlets, nel quale vengono raccolte tutte le actions).

Importante è sottolineare il fatto che un portlet sviluppato in Jetspeed con la tecnologia di Velocity, può derivare sia dall'estensione della classe Velocity che di quella CustomizerVelocity, con la differenza che i portlet derivanti dalla prima classe, utilizzano una personalizzazione di default, mentre quelli che derivano dalla CustomizerVelocity possono definire anche una personalizzazione della loro visualizzazione.

```
<portlet-entry name="TutorialStockQuote1" hidden="false"
type="ref"
        parent="Velocity" application="false">
……
……
……
</portlet-entry>
```

```
<portlet-entry name="Wheather Portlet" hidden="false" type="ref"
        parent="CustomizerVelocity" application="false">
……
……
……
</portlet-entry>
```

Figura 51: -Esempi di Velocity Portlet nel registry-

6.1.5 La pubblicazione di Velocity (Velocity templates)

Al fine di analizzare come avviene una pubblicazione con l'uso della tecnologia di Velocity, si è deciso di analizzare concretamente un esempio. Supponiamo allora di voler pubblicare delle quotazione finanziarie in tempo reale, che vengono salvate in records riferiti, nell'esempio, col nome di $quotes. Analogamente anche i titoli

delle colonne, rappresentative dei valori contenuti in $quotes, vengono memorizzate in record di nome $coloumns.

```
<table>
 <tr>
    <td>
      <table border="true" cellspacing="1" cellpadding="3">
        <tr>
          #foreach ($column in $columns)
           #headerCell ($column)
          #end
        </tr>

        #foreach ($quote in $quotes)
        <tr>
          #entryCell ($quote.Symbol)
          #entryCell ($quote.Price)
          #entryCell ($quote.Change)
          #entryCell ($quote.Volume)
        </tr>
        #end
      </table>
    </td>
  </tr>
</table>
```

Figura 52: -Esempio di Velocity Template-

Come si nota dall'esempio di codice riportato, il primo ciclo individuato dalla riga #foreach ($column in $coloumns), ha lo scopo di visualizzare tutti i titoli delle colonne che conterranno i valori memorizzati nel record $quotes successivamente viene eseguito un secondo ciclo #foreach($quote in $quotes) per inserire i valori contenuti nei singoli array. Si nota come per l'inserimento dei valori contenuti nei record all'interno della pagina web siano state necessari due tipologie di macro. In particolare #headerMacro e #entryCell sono macro definite in un file di nome Velocimacro che viene condiviso con Jetspeed. Una caratteristica delle Velocimacro è quella di essere usate per definire frammenti di codice, poi richiamabili in altri templates e perfino in altre macro. Tutte le macro vengono salvate in file con estensione .vm.

In generale, la risoluzione dei templates si basa su diversi file di configurazione e diversi algoritmi di risoluzione. In particolare nel file *TurbineResource.properties,* posto nel sottodiretorio conf, viene inserito il localpath nel quale ricercare i Velocity Portlet templates.

```
# This parameter supports a comma separated list of directories
# Each directory is searched in order to find a template.
# This is useful for example, in defining application specific
templates in a separate structure from the jetspeed core
templates
# Note this needs to be set in 3 places - for the jsp loader and
vm loader in TR.p and for the template locator in JR.p
services.VelocityService.file.resource.loader.path = /WEB-
INF/templates/vm
```

Figura 53: -Configurazione del file TurbineResource.properties-

Per convenzione Jetspeed ricerca i Velocity Portlet templates nel Jetspeed deployment all'interno dei sottodirettori, a seconda delle necessità:

- /WEB INF/templates/vm/portlets/html per i portlets sviluppati in HTML
- /WEB INF/templates/vm/portlets/wml per i portlets sviluppati in WML

Settare il file di configurazione di Turbine, non è sufficiente, occorre aggiornare, con lo stesso path limitato alla root, anche il file JetspeedResource.properties.

```
########################################
# Template Locator Service             #
########################################
# The Template Locator is implemented as a Turbine service.
services.TemplateLocator.classname=org.apache.jetspeed.services.t
emplate.JetspeedTemplateLocatorService

# This parameter supports a comma separated list of directories
# Each directory is searched in order to find a template.
# This is useful for example, in defining application specific
templates in a #separate structure from the jetspeed core
templates
# Note this needs to be set in 3 places - for the jsp loader and
vm loader in TR.p #and for the template locator in JR.p
services.TemplateLocator.templateRoot=/WEB-INF/template
```

Figura 54: -Configurazione del file JetspeedResource.properties-

Per quanto riguarda gli algoritmi di risoluzione, il loro compito è quello di individuare i templates necessari. Questi algoritmi vengono localizzati in sottodirettori chiamati con le abbreviazioni del linguaggio utilizzato e della nazione di provenienza del client. I codici utilizzati per la creazione dei nomi dei direttori, seguono lo standard ISO-639 consultabile al sito: http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt .

Fino ad ora abbiamo visto come, noti riferimenti alle informazioni da visualizzare, fosse possibile pubblicare ed inserirle all'interno di un EIP. Rimane da analizzare come questi contenuti possano essere recuperati e resi leggibili alla fase di template del Velocity portlet.

6.1.6 Velocity action

Velocity action sono delle classi java con il compito di controllare la logica sottostante i contenuti: hanno il compito di ricercare, fornire o immagazzinare informazioni di tipo dinamico attraverso procedure di backend[8]. Ogni Velocity action, nota anche come controller, estende la classe VelocityPortletAction la quale implementa comodi metodi capaci di aiutare lo sviluppatore nella gestione dei contenuti. Come riportato anche nel paragrafo 5.2.3 raffigurante il ciclo di vita di un portlet, si nota come la fase processAction, precede la fase di render. Questo implica che, i metodi definiti nei controller, vengono invocati prima della pubblicazione dei contenuti, permettendo quindi, allo sviluppatore, di creare i contenuti dinamicamente a seconda delle richieste di visualizzazione ossia in base ai *portlet mode*. L'utilizzo di questo approccio permette un'agevole personalizzazione dei contenuti in base ai portlet mode richiesti.

Generalmente, i contenuti richiesti nel mode maximize è lo stesso di quello pubblicato nel normal mode, ma è data la possibilità allo sviluppatore di definire una personalizzazione della visualizzazione attraverso l'implementazione del metodo buildConfigureContext; per mantenere la personalizzazione di default, data da Jetspeed, basta non riscrivere tale metodo.

METHOD	PORTLET MODE
builNormalContext	View
buildConfigureContext	Customize (Edit)
buildMaximizedContext	Maximize

Figura 55: -Metodi e relativi modi di visualizzazione di un portlet -

Per motivi di completezza, si riporta lo stralcio del controller che si occupa della gestione dei contenuti nell'esempio delle quotazioni borsistiche.

```
public class TutorialStockQuoteAction1 extends
VelocityPortletAction
{
    private static final String SYMBOLS = "symbols";
    private static final String COLUMNS = "columns";
    private static final String QUOTES = "quotes";
    private static final String[] ALL_COLUMNS =
                {"Symbol","Price","Change","Volume"};

protected void buildNormalContext(VelocityPortlet portlet,
```

[8] Con il termine di procedure backend, ci si riferisce ad applicazioni che funzionano in modo trasparente all'utente.

```
                                        Context context,
                                        RunData rundata)
    {
        try
        {
            // Get reference to stock quote web service
            StockQuoteService service = (StockQuoteService)
                  TurbineServices.getInstance().
                      getService(StockQuoteService.SERVICE_NAME);

            // Retrieve portlet parameters
            String symbols =
PortletConfigState.getParameter(portlet,
                    rundata, SYMBOLS, "IBM,MSFT,ORCL,SUNW");

            String[] symbolArray = StringUtils.stringToArray(
                                                    symbols, ",");
            StockQuote[] quotes =
service.fullQuotes(symbolArray);

            // Place appropriate objects in Velocity context
            context.put(QUOTES, quotes);
            context.put(COLUMNS, ALL_COLUMNS);
        }
        catch (Exception e)
        {
            Log.error(e);
        }
    }
```

Figura 56: Codice di Esempio - VelocityAction –

Il TuttorialStockQuoteAction, ha il compito di trovare e fornire le quotazioni finanziarie da un servizio web, noto come *web service* e ritornare un array di quotazioni chiamato StockQuote. Per compiere queste azioni nel Velocity Action viene implementato un solo metodo di nome BuildNormalContext il quale fornisce l'array al contesto e lo mette a disposizione del template, il quale può estrarre le informazioni che gli interessano.

6.1.7 Velocity Action Event

Gli Action Event, gestiti dalla classe VelocityPortleAction, permettono di legare fra loro eventi che accadono sull'interfaccia utente, proprio come il premere di un pulsante o la compilazione di un di un form ed il relativo invio di dati.

Per analizzare in dettaglio il funzionamento di un Action Event è preferibile riportare un esempio nel qual vengono inseriti attraverso una form cinque dati di input i quali verranno salvati nella sessione corrente dell'utente e quindi non rimarranno in modo persistente nel registry.

Innanzitutto analizziamo l'inserimento del portlet nel file di configurazione di Jetspeed:

```
<portlet-entry name="CobiJonesPortlet" hidden="false" type="ref"
        parent="Velocity" application="false">
        <meta-info>
            <title>The Cobi Jones Portlet</title>
            <description>Tutorial showing an Action Event
Executing,
                dedicated to the man Cobi</description>
        </meta-info>

<classname>org.apache.jetspeed.portal.portlets.VelocityPortlet</c
lassname>
        <parameter name="template" value="cobi-jones-form"
hidden="true"
            cachedOnName="true" cachedOnValue="true"/>
        <parameter name="action"
value="portlets.CobiJonesPortletAction"
            hidden="true" cachedOnName="true"
cachedOnValue="true"/>
        <media-type ref="html"/>
        <url cachedOnURL="true"/>
        <category group="Jetspeed">tutorial</category>
    </portlet-entry>>
```

Figura 57: Codice di Esempio –Inserimento nel registry di un Velocity Portlet -

Riportiamo anche interamente il file controller il cui .class viene memorizzato nel sottodirettorio :
\WEB-INF\classes\com\bluesunrise\jportal\modules\actions\portlets
delle applicazioni del server.

```
package com.bluesunrise.jportal.modules.actions.portlets;

import org.apache.jetspeed.portal.portlets.VelocityPortlet;

// Turbine stuff
import org.apache.turbine.util.Log;
import org.apache.turbine.util.RunData;
import org.apache.turbine.services.TurbineServices;

// Velocity Stuff
import org.apache.velocity.context.Context;

// Jetspeed stuff
import org.apache.jetspeed.portal.portlets.VelocityPortlet;
import
org.apache.jetspeed.modules.actions.portlets.VelocityPortletActio
n;
import org.apache.jetspeed.util.PortletConfigState;
import org.apache.jetspeed.util.StringUtils;

/**
 * Tutorial 7 - Action Events
 *
 * @author <a href="mailto:taylor@apache.org">David Sean
Taylor</a>
 */

public class CobiJonesPortletAction extends VelocityPortletAction
{
```

```
    public static final String INPUT_FIRST_NAME ="firstname";
    public static final String INPUT_LAST_NAME  = "lastname";
    public static final String INPUT_POSITION   = "position";
    public static final String INPUT_CAPS       = "caps";
    public static final String INPUT_ACTIVE     = "active";
    public static final String PLAYER = "player";
    public static final String COBI_ERROR = "cobierror";


    /**
     * Build the normal state content for this portlet.
     *
     * @param portlet The velocity-based portlet that is being
built.
     * @param context The velocity context for this request.
     * @param rundata The turbine rundata context for this
request.
     */
    protected void buildNormalContext(VelocityPortlet portlet,
                                      Context context,
                                      RunData rundata)
    {
        try
        {
            Player player =
(Player)rundata.getUser().getTemp(PLAYER);
            if (null == player)
            {
                player = createNewPlayer();
                rundata.getUser().setTemp(PLAYER, player);
            }
            context.put(PLAYER, player);

            // make sure they don't sub Cobi!
            String cobiError =
(String)rundata.getRequest().getAttribute(COBI_ERROR);
            if (null != cobiError)
            {
                context.put(COBI_ERROR, cobiError);
            }
        }
        catch (Exception e)
        {
            Log.error(e);
            context.put(COBI_ERROR, e.toString());
        }
    }

    /**
     * Update the portlet state from the form.
     *
     * @param rundata The turbine rundata context for this
request.
     * @param context The velocity context for this request.
     */
    public void doUpdate(RunData rundata, Context context) throws
Exception
    {
        Player player =
(Player)rundata.getUser().getTemp(PLAYER);
        if (null == player)
```

```
        {
            player = createNewPlayer();
            rundata.getUser().setTemp(PLAYER, player);
        }

        String cobi =
rundata.getParameters().getString(INPUT_FIRST_NAME);
        String jones =
rundata.getParameters().getString(INPUT_LAST_NAME);
        if (!cobi.equalsIgnoreCase("Cobi") ||
!jones.equalsIgnoreCase("Jones"))
        {
            rundata.getRequest().setAttribute(COBI_ERROR,
                                              "Hey now, you cant
substitute Cobi with "
                                               + cobi + " " +
jones + "!");
        }
        player.setFirstName(cobi);
        player.setLastName(jones);

player.setPosition(rundata.getParameters().getString(INPUT_POSITI
ON));

player.setCaps(rundata.getParameters().getInt(INPUT_CAPS));

player.setActive(rundata.getParameters().getBoolean(INPUT_ACTIVE)
);

        rundata.getUser().setTemp(PLAYER, player);
    }

//////////////////////////////////////////////////////////////

    public Player createNewPlayer()
    {
      return new Player("Cobi", "Jones", "Midfielder", 150,
true);
    }

    public class Player
    {
        private String firstName;
        private String lastName;
        private String position;
        private int caps;
        boolean active;

        public Player(String firstName,
                      String lastName,
                      String position,
                      int caps,
                      boolean active)
        {
            this.firstName = firstName;
            this.lastName = lastName;
            this.position = position;
            this.caps = caps;
            this.active = active;
        }

        public String getFirstName()
```

```
        {
            return this.firstName;
        }

        public void setFirstName(String firstName)
        {
            this.firstName = firstName;
        }

        public String getLastName()
        {
            return this.lastName;
        }

        public void setLastName(String lastName)
        {
            this.lastName = lastName;
        }

        public String getPosition()
        {
            return this.position;
        }

        public void setPosition(String position)
        {
            this.position = position;
        }

        public int getCaps()
        {
            return this.caps;
        }

        public void setCaps(int caps)
        {
            this.caps = caps;
        }

        public boolean getActive()
         {
             return this.active;
         }

         public void setActive(boolean active)
         {
             this.active = active;
         }
    }
}
```

Figura 58: Codice di Esempio – Controller di un Velocity Portlet -

All'avvio del portlet, si ottiene il seguente risultato:

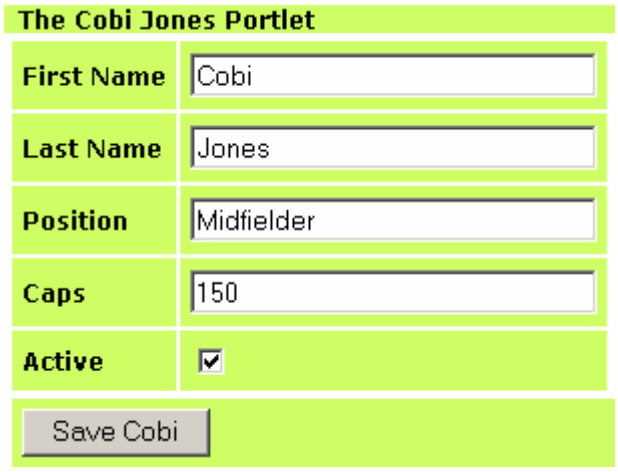


Figura 59: - Output del codice precedente -

Nel momento in cui si clicca il bottone "Save Coby", i valori contenuti nel form verranno inviati al Turbine controller servlet, il quale darà la possibilità di utilizzare i dati ricevuti alla classe action Java riportata sotto.

```
public void doUpdate(RunData rundata, Context context)
throws Exception
{
    Player player =
(Player)rundata.getUser().getTemp(PLAYER);
    if (null == player)
    {
        player = createNewPlayer();
        rundata.getUser().setTemp(PLAYER, player);
    }
    String cobi =
rundata.getParameters().getString(INPUT_FIRST_NAME);
    String jones =
rundata.getParameters().getString(INPUT_LAST_NAME);
    if (!cobi.equalsIgnoreCase("Cobi") ||
        !jones.equalsIgnoreCase("Jones"))
    {
        rundata.getRequest().setAttribute(COBI_ERROR,
               "Hey now, you cant substitute Cobi with "
                + cobi + " " + jones + "!");
    }
    player.setFirstName(cobi);
    player.setLastName(jones);

player.setPosition(rundata.getParameters().getString(INPUT_P
OSITION));

player.setCaps(rundata.getParameters().getInt(INPUT_CAPS));

player.setActive(rundata.getParameters().getBoolean(INPUT_AC
TIVE));

    rundata.getUser().setTemp(PLAYER, player);
}
```

Figura 60 Codice di Esempio - Action class -

Come detto anche in precedenza, Velocity Action definisce tre metodi standard: BuildNormalContext, BuildConfigureContext e BuildMaximizedContext, ai quali possono essere aggiunti anche altri action event. Nel nostro caso particolare si vogliono trasferire i dati inseriti nel form al portlet. Si possono ottenere i contenuti del form attraverso i parametri inviati dal rundata all'action event. Jetspeed inserisce tutti i parametri del form in un unico parametro di passaggio dal quale è possibile prelevare stringhe e dati da trasferire poi nel portlet. Proprio questa operazione viene svolta nel metodo doUpdate attraverso le seguenti istruzioni:

```
String cobi= rundata.getParameters().getString(INPUT_FIRST_NAME);
…
player.setActive(rundata.getParameters().getBoolean(INPUT_ACTIVE)
);
```

Nel caso esaminato, i dati sono memorizzati in una pagina di sessione, attraverso dei metodi forniti da Jetspeed come:

```
rundata.getUser().setTemp(PLAYER, player);
……
Player player = (Player)rundata.getUser().getTemp(PLAYER);
```

Per settare il giusto action listener, occorre intervenire sul file di configurazione del Velocity template ed impostare il giusto link; si riporta allora uno stralcio del file con estensione .vm:

```
<form method="post"
action="$jslink.setAction("portlets.CobiJonesPortletAction")">
  <div align="left">
    <table bgcolor="#ffffff" cellpadding="5">

      #if ($cobierror)
        ............
        ............
      #end

      <tr>
        #formCell ("First Name" "firstname" $player.FirstName)
      </tr>
      <tr>
        #formCell ("Last Name" "lastname" $player.LastName)
      </tr>
      <tr>
        #formCell ("Position" "position" $player.Position)
      </tr>
      <tr>
        #formCell ("Caps" "caps" $player.Caps)
      </tr>
      <tr>
        #formCheckBox2 ("Active" "active" $player.Active)
      </tr>
  </table>
```

```
  <table bgcolor="#ffffff" cellpadding="5" width="100%">
      <tr>
        <td align="$ui.buttonAlignment"
bgcolor="$!{skin.TitleBackgroundColor}">
        <input type="submit" name="eventSubmit_doUpdate"
value="Save Cobi"/>
      </td>
      <td>
         $!msg
      </td>
    </tr>
  </table>

</div>
</form>
```

Figura 61: Codice di Esempio – Configurazione dell'Action Listener nel Velocity template -

L'istruzione $jslink , si occupa di linkare insieme il form con la corrente pagina del portale, mentre il metodo setAction, specifica l'azione che governerà il portlet. Il tasto di submit, sarà quello che , quando premuto, dovrà far scatenare il traseferimento dei dati dal form al portlet. L'istruzione: input type="submit" name="eventSubmit_doUpdate" è quella che si preoccupa di collegare l'azione compiuta sull'interfaccia rappresentata dalla pressione del bottone, con il richiamo del metodo doUpdate.

Importante notare come, per individuare i campi di input, si siano utilizzati i macro standard di input di Jetspeed ed inoltre come, per catturare tali dati, si siano utilizzate le istruzioni di #formcell. Un dato da non sottovalutare è la necessità di chiamare l'attributo nello stesso modo con cui lo si tratta nel java ActionListener.

Nell'ultima parte del metodo doUpdate, si dimostra come si possano gestire anche eventuali eccezioni passando dei semplici parametri, come delle stringhe, fra l'action event al metodo BuildNormalContext. Tale metodo controllerà la presenza di una eccezione ed in caso di presenza la pubblicherà attraverso l'istruzione:

```
// make sure they don't sub Cobi!

String
cobiError=(String)rundata.getRequest().getAttribute(COBI_ERR
OR);
if (null != cobiError)
{
       context.put(COBI_ERROR, cobiError);
}
```

Figura 62: Codice di Esempio –Gestione delle eccezioni -

Tale messaggio di errore verrà captato dal template il quale lo pubblicherà in questo modo:

```
#if ($cobierror)
      <tr>
        <td colspan="2">
          <table bgcolor="red">
            <tr>
              <td>
                $cobierror
              </td>
            </tr>
          </table>
        </td>
      </tr>
      #end
```

Figura 63: - Output dell'eccezione gestita dal template -

6.2 JSP

Come già analizzato approfonditamente nel capitolo riguardante la piattaforma J2EE, la Java Server Pages (JSP) è una tecnologia basata sul linguaggio java sviluppata dalla Sun Microsystem allo scopo di sviluppare applicazioni per il lato server. Le pagine JSP non sono altro che pagine HTML con speciali tags contenenti codice Java in grado di fornire contenuti dinamici. Le pagine JSP offrono una robusta piattaforma per lo sviluppo di applicazioni web in quanto sono

- Multipiattaforma
- Componenti riusabili grazie all'utilizzo di JavaBeans ed EJB
- Sono in grado di separare i contenuti dal codice

In modo analogo a quanto visto per i Velocity Portlet, è possibile sviluppare portlets con la tecnologia JSP, rispettando sempre la MVC design.

6.2.1 JSP Portlet

Anche un portlet JSP è costituito dei principali componenti MVC:

MVC Component	JSP Component
Model	Java Object put in as request attributes
View	Template
Controller	Your JSP Action

Figura 64: - componenti MVC di un portlet sviluppato con tecnologia JSP -

Il compito di controllore (*controller*) viene svolto dalla JSP Action C lass. La classe base JspPortlet non dovrebbe essere quasi mai modificata. La visualizzazione della pagina è la JSP Template la quale genera il contenuto del portlet estraendo informazioni dinamicamente dagli attributi richiesti *request attribute*. In aggiunta alle variabili standard (request, response,session,ecc…) sono disponibili anche le seguenti variabili:

Variabile	Tipo	Descrizione
rundata	RunData	Si riferisce alle richieste degli oggetti rundata
js_peid	String	Identificatore univoco del portlet
link	JspLink	Istanza di un oggetto JspLink
portlet	Portlet	Riferimento a questo oggetto portlet

Figura 65: - Variabili di un portlet -

Tali variabili possono essere recuperate attraverso delle request attributes come segue:

```
String jspeid = (String) request.getAttributes("js_peid");
```

Il metodo *getContent()* della classe JspPortlet non dovrebbe mai essere modificato. Tutto il contenuto viene generato attraverso i template, proprio come previsto dal modello MVC.

Anche nel caso dei portlet sviluppati con la tecnologia JSP, il ciclo di vita del portlet è contraddistinto da una fase ulteriore che si inserisce fra la fase di Init e quella di Render:

Phase	Method
Init	Init

ProcessAction	JSP Portlet Action and Action Event
Render	Template is called by Jsp Portlet
Destroy	-- none --

Figura 66: - Fasi di un portlet sviluppato con tecnologia JSP -

6.2.2 JSP Portlet in the Registry

Come un qualsiasi altro portlet da integrare in un EIP sviluppato con Jetspeed, occorre definire il JSP portlet nel file di *registry*.

```
<portlet-entry name="HelloJSP" hidden="false" type="ref"
               parent="JSP" application="false">
               ……
               <parameter name="template" value="hello.jsp"
hidden="true"/>
        ……
        <parameter name="action" value="portlets.hello.jsp"
hidden="true"/>
    </portlet-entry>
```

Figura 67: - Definizione del JSP portlet nel file di configurazione –

Quando si definisce un portlet sviluppato in JSP occorre definire il ***templat***e le ***actìn*** . In particolare il template definisce le JSP template che si occupano della pubblicazione dei contenuti (ossia è la MVC view); mentre la Action rappresenta il controllore, ossia quell'elemento responsabile della gestione degli eventi e della gestione degli attributi nella request..

Il template dovrebbe essere salvato in un sottodirettorio di un JSP Templates path, mentre le action sono convenzionalmente situate nel sottodirettorio dei portlets del direttorio di root delle action .

6.2.3 JSP Template

Analogamente a quanto fatto per il Velocity template, riportiamo un esempio di codice del tutorial di Jetspeed, nel quale si prepara una pagina JSP per pubblicare in modo dinamico il valore di alcune quotazioni recuperate da un Web Services:

```
<%@ taglib uri='/WEB-INF/templates/jsp/tld/template.tld'
prefix='jetspeed' %>

<%@ page import = "org.apache.turbine.util.Log" %>
<%@ page import = "org.apache.jetspeed.webservices.finance.
stockmarket.StockQuote" %>
```

```
<%
try{
    StockQuote[] quotes = (StockQuote[])
request.getAttribute("quotes");
    String[] columns = (String[])
request.getAttribute("columns");
    String jspeid = (String) request.getAttribute("js_peid");
%>

<FORM METHOD="POST">
  <INPUT TYPE="hidden" NAME="js_peid" VALUE="<%=jspeid%>">
  Enter symbol(s) separated with commas: <input name="symbols"
type="TEXT"><INPUT TYPE="SUBMIT" NAME="refresh" VALUE="Get
Quotes">
</FORM>
<table>
  <tr>
    <td>
      <table border="true" cellspacing="1" cellpadding="3">
        <tr>
          <%for (int i = 0; columns != null && i <
columns.length; i++) {%>
            <TH><%=columns[i]%></TH>
          <%}%>
        </tr>

          <%for (int j = 0; quotes != null && j < quotes.length;
j++) {%>
        <tr>
          <TD><%=quotes[j].getSymbol()%></TD>
          <TD><%=quotes[j].getPrice()%></TD>
          <TD><%=quotes[j].getChange()%></TD>
          <TD><%=quotes[j].getVolume()%></TD>
        </tr>
          <%}%>
      </table>
    </td>
  </tr>
</table>
<%} catch (Exception e) {
    Log.error(e);
    return;
}%>
```

Figura 68: Codice di Esempio - Pagina JSP per la pubblicazione di quotazioni borsistiche -

Nel codice riportato è importante notare il parametro recuperato attraverso la variabile *jspeid*: Il valore contenuto neela variabile identifica unicamente il portlet e quindi identifica unicamente il form che esercita la richiesta nel contesto del portale anche nel caso di istanze multiple di uno stesso portlet.

6.2.4 JSP Template resolution

Di default Jetspeed ricerca le JSP templates nel sottodirettorio /WEB-INF/templates/jsp/portlets/html per i portlets sviluppati in html, nel direttorio /WEB-INF/templates/jsp/portlets/wml per i portlets sviluppati con tecnologia WML. Analogamente a quanto visto per il Velocity template, anche per le pubblicazioni con tecnologia JSP occorre configurare il file *Turbine.resources.properties* e porre:

```
services.JspService.template = /WEB-INF/templates/jsp, /WEB-
INF/my-templates/jsp
```

Si possono specificare percorsi multipli, separandoli da una virgola, proprio come riportato sopra.

Occorre configurare inoltre anche il file *jetsppedResource.properties* come riportato nell'esempio sotto:

```
services.TemplateLocator.templateRoot = /WEB-INF/my-
templates, /WEB-INF/my-templates
```

L'algoritmo dedicato alla risoluzione delle pubblicazioni è un servizio che autonomamente ricercherà i file necessari alle pubblicazioni secondo i parametri configurati.

6.2.5 JSP Portlet Actions

Le JSP portlet Actions sono delle classi Java nelle quali gli sviluppatori possono inserire il codice atto alla gestione logica dei dati. In queste classi è possibile inserire ogni tipo di controllo di back-end necessario a recuperare o memorizzare in modo trasparente all'utente informazioni di tipo dinamico. Questi portlet actions possono essere utilizzati anche come edit bean per popolare i contenuti delle pagine JSP. Per completezza ed analogia ai paragrafi precedenti riportiamo un codice di esempio di un JSP Portlet Action:

```
public class TutorialStockQuoteAction8 extends
JspPortletAction
{
    private static final String SYMBOLS = "symbols";
    private static final String COLUMNS = "columns";
    private static final String QUOTES = "quotes";
    private static final String[] ALL_COLUMNS =
{"Symbol","Price","Change","Volume"};

    /**
     * Build the normal state content for this portlet.
     *
     * @param portlet The jsp-based portlet that is being
built.
     * @param rundata The turbine rundata context for this
request.
     */
    protected void buildNormalContext(Portlet portlet,
                                      RunData rundata)
    {
        try
        {
            // Get reference to stock quote web service
            StockQuoteService service = (StockQuoteService)
TurbineServices.getInstance().
```

```
                getService(StockQuoteService.SERVICE_NAME);

            // Retrieve portlet parameters
            String symbols = (String)
PortletSessionState.getAttributeWithFallback(portlet,
rundata, SYMBOLS);

            // Request stock quote(s) from the stock quote
web service
            String[] symbolArray =
StringUtils.stringToArray(symbols, ",");
            StockQuote[] quotes =
service.fullQuotes(symbolArray);

            // Place appropriate objects in jsp context
            rundata.getRequest().setAttribute(QUOTES,
quotes);
            rundata.getRequest().setAttribute(COLUMNS,
ALL_COLUMNS);
        }
        catch (Exception e)
        {
            Log.error(e);
        }
    }
}
```

Figura 69: Codice di Esempio -JSP Portlet Action -

Questo portelt recupera da un servizio web delle quotazioni e le pubblica attraverso una pagina JSP. Come esempio è stato rieditato solo il metodo buildNormalcontext(). Questo metodo ritorna un array di oggetti StockQuote ogniqualvolta l'utente richiede un determinato StockSymbol.

```
String symbols = (String)
PortletSessionState.getAttributeWithFallback(portlet,
rundata, SYMBOLS);
```

Il codice sopra riportato recupera i parametri del portlet secondo questo algoritmo:

- Se i parametri sono presenti nella request li ricava da questa e li memorizza nella sessione altrimenti
- Se i parametri sono già nella sessione, li recupera da questa altrimenti
- Se i parametri sono presenti nell'istanza del portlet cioè nel file PSML relativi all'utente li recupera da questa altrimenti
- I parametri vengono recuperati dal file di configurazione del portlet nel file di registry.

6.2.6 JSP Event

Le action Event permettono di gestire le interazioni dell'utente attraverso user interface (come il submit di una form o il click di un pulsante) e collegare a queste azioni determinati eventi. Per far questo occorre:

- Specificare il nome del JspPortleAction in un hidden input field chiamato "action"
- Definire un pulsante con un nome = "action"
- Dichiarare un gestore degli eventi (event handler) nel JspPortletAction

Per esempio:

```
<form method="post"  action="<jetspeed:dynamicUri/>">
 <input type="hidden" name="js_peid" value="<%=jspeid%>">
 <input type="hidden" name="action" value="
portlets.myJspPortletAction "/>
 <input type="submit" name="eventSubmit_doUpdate" value="Save"/>
```

```
public void doUpdate(RunData rundata, Portlet portlet) throws
Exception
{
 // action logic goes here
}
```

Capitolo 7

Implementazione della piattaforma J2EE: Tomcat e Jetspeed

In questo capitolo viene introdotta la particolare implementazione della piattaforma J2EE utilizzata per la creazione di un EIP.

7.1 Packing and deployment

Proprio come consentito dalla J2EE platform, Jetspeed è concepito come applicazione composta da componenti riusabili, i quali sono raggruppati in insiemi, chiamati *moduli,* a seconda delle loro caratteristiche e funzionalità comuni. Prima di continuare l'analisi dell'architettura risulta quindi fondamentale analizzare quali tipologie di moduli si possono incontrare:

- MODULI EJB: Un modulo EJB è l'unità più piccola di EJB della quale si può fare il deployment. Tale modulo è impacchettato in un file EJB JAR, cioè un file standard Java JAR con estensione .jar che contiene:
  1. le classi Java degli EJB e le loro remote e home interface. Se si tratta di un entity bean deve essere compresa anche la classe della primary key;
  2. tutte le altre classi Java dalle quali dipendono gli EJB che non siano già comprese nella J2EE platform;
  3. un EJB deployment descriptor, tipicamente chiamato ejb-jar.xml che deve essere contenuto in una particolare directory di nome META-INF.

  E' da notare che un EJB JAR file differisce da un normale JAR file perché nel primo è contenuto anche il deployment descriptor.
- MODULI WEB: Un modulo Web è l'unità più piccola di risorse Web della quale si può fare il deployment. Tale modulo è impacchettato in un file Web ARchive (WAR), un file standard Java JAR con estensione .war che contiene:
  1. le classi Java delle servlet e le classi dalle quali queste dipendono, eventualmente impacchettate a loro volta in un normale file JAR;
  2. le pagine JSP e le classi Java dalle quali dipendono;
  3. documenti statici, come ad esempio le pagine HTML, le immagini, i file sonori, ecc. ;
  4. le applets;

5. un Web deployment descriptor, tipicamente chiamato web.xml e contenuto in una speciale directory chiamata WEB-INF.

- MODULI APPLICATION CLIENT: Un modulo application client è impacchettato in un file JAR con estensione .jar e contiene:
  1. le classi Java che implementano il client;
  2. un Application client deployment descriptor che descrive gli EJB e le risorse esterne referenziate dall'applicazione.

La J2EE non specifica tools per effettuare il deployment di un application client, o meccanismi per installarlo. Alcune piattaforme J2EE sofisticate possono consentire il deployment dell'application client direttamente nel J2EE server e metterlo così automaticamente a disposizione di alcuni clients intranet. Altre piattaforme J2EE possono invece richiedere che il deployment dell'application client sia manualmente effettuato su ciascuna macchina client.

Il processo di assemblaggio dei componenti nei moduli sopra indicati e dei moduli in enterprise applications è detto *packaging*, mentre il processo di installazione e personalizzazione di una applicazione in un certo contesto operativo è detto invece *deployment*.

Tali processi sono fondamentali per poter utilizzare gli applicativi, e sono talmente importanti che occorrono meccanismi standard di configurazione.

Gli strumenti standard utilizzati per semplificare i processi di packing e deployment si ritrovano anche in Jetspeed ed in particolare sono:

- File JAR come package standard per i moduli e le applicazioni
- File XML-based chiamati *deployment descriptor* per configurare tali moduli.

In particolare, una volta decompresso il file sorgente in una opportuna directory che chiameremo per semplicità JETSPEED_HOME, si possono osservare le seguenti sottodirectory che assumono ruoli importanti per la configurazione, la personalizzazione e l'utilizzo di Jetspeed.

- \bin : contiene il file jetspeed.jar e tutte le classi java già compilate nei rispettivi .class raccolte in un package.
- \build : contiene i file necessari alla configurazione e alla compilazione di Jetspeed per trasferirlo come applicazione all'interno del direttorio preposto nel server web.
- \lib : contiene tute le librerie, compresse nei formati .jar, atte al funzionamento di Jetspeed.

7.2 Java Development Kit

Dal sito della Sun: http://java.sun.com/j2se si è scaricato gratuitamente j2sdk1.4.0.02 un kit pronto per lo sviluppo di applicazioni java, contenete anche una java virtual machine. Come richiesto dalle specifiche di installazione, è stato

necessario definire nelle *proprietà* di *risorse del computer* una variabile di ambiente di nome JAVA_HOME e definirgli come valore il nome assoluto del direttorio in cui è stato installata la j2sdk *<javat_home>*.

7.3 Ant

Ant [Ant] è un tool di build. In altre parole, come definito dal gruppo di Apache sulla home page del progetto, Ant è una sorta di "make" senza le stranezze del make. Più specificatamente Ant è un programma Java che esegue dei comandi che possono essere del tipo: "compila questi file", "crea questa directory", "crea questo jar con i file di questa directory", "esegui javadoc su questi sorgenti"...e così via. Ant ci viene in aiuto quando c'e' l'esigenza di automatizzare tutti quegli step necessari per ottenere il prodotto finale di un progetto di grandi dimensioni, a partire dai nostri sorgenti. Ad esempio, se stiamo sviluppando una Enterprise Application, per ottenere il nostro file .ear dobbiamo compilare i sorgenti della nostra applicazione, creare un file jar (con tanto di descrittori) per ogni ejb, creare un war con tutte le nostre jsp e tutte le classi di supporto messe al posto giusto, creare infine l'ear che contiene il war e tutti gli ejb. Ora, fare a mano tutto ciò è tedioso ed è soggetto ad errori. E' per questo che su molti progetti viene utilizzato Ant: opportunamente istruito, Ant può svolgere tutti questi compiti in modo automatico.

Ant è scritto completamente in Java ed è quindi multipiattaforma. Ant è un programma che si lancia "a riga di comando", non ha quindi un'interfaccia grafica. I comandi che Ant esegue sono letti da un file xml, di solito chiamato build.xml. In questo file possiamo scrivere i comandi che Ant deve eseguire, tramite l'utilizzo di opportuni tag.

La home page del progetto Ant è la seguente: http://jakarta.apache.org/ant. L'ultima versione di Ant attualmente scaricabile dal sito è la 1.5.1. Una volta scaricato il programma occorre scompattate il file in una directory a piacimento e settare le variabili di environment ANT_HOME e JAVA_HOME. La prima deve puntare alla directory che contiene Ant, mentre JAVA_HOME punta alla directory in cui abbiamo installato il JDK. Naturalmente occorre ricordarsi di aggiornare la variabile di ambiente PATH. Ad esempio, sotto Unix

```
export ANT_HOME=/usr/local/ant
export JAVA_HOME=/usr/local/jdk-1.2.2
export PATH=${PATH}:${JAVA_HOME}/bin:${ANT_HOME}/bin
```

mentre sotto Windows:

```
set ANT_HOME=c:\ant
set JAVA_HOME=c:\jdk1.2.2
set PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin
```

Una volta settati questi parametri non resta altro che aprire una finestra shell di dos, portarsi nel direttorio da "buildare" (contenente il file build.xml) e lanciare il comando Ant con gli eventuali target.

7.4 Servlet engine Tomcat

Tomcat, come visto anche in precedenza, è un servlet container che offre il supporto per lo standard Servlet 2.2 e per lo standard JSP 1.2. In particolare nella release 4.1.18, installata in questa specifica architettura, implementa un nuovo servlet container (chiamato Catalina) che si differenzia architeturalmente dalle precedenti versioni e supporta le specifiche per le Servlet 2.3 e JSP 1.2. Tomcat 4.1.18 è scritto interamente in Java ed è messo a disposizione open source dal progetto Jakarta al sito http://jakarta.apache.org/tomcat/index.html .

7.4.1 Configurazione e installazione di Tomcat stand alone

Una volta scaricato il file eseguibile *Tomcat-4.1.18.exe*, è sufficiente mandarlo in esecuzione ed automaticamente verrà individuata la J2SDK ed un direttorio in cui installarsi. Tomcat verrà installato in un direttorio che chiameremo per comodità <*Tomcat_home*>. Come richiesto dalle specifiche di installazione, occorre definire nelle *proprietà* di *risorse del computer* una variabile di ambiente di nome CATALINA_HOME e assegnargli come valore il nome assoluto del direttorio in cui è stato installato tomcat <*Tomcat_home*>.

In <*Tomcat_home*> si possono osservare le seguenti sottodirectory che assumono ruoli importanti per la configurazione e l'utilizzo di Tomcat:

- /bin : contiene i file eseguibili di sistema che servono per far partire e per far terminare il server;
- /conf : contiene i file di configurazione del server, compreso il file server.xml di cui si è accennato;
- /lib : contiene tutte le librerie necessarie al funzionamento del server;
- /logs : contiene i file di log di sistema;
- /webapps : contiene tutte le applicazioni Web di cui viene fatto il deployment; queste devono essere organizzate con la struttura gerarchica prevista dalle specifiche Servlet 2.2 e possono eventualmente essere impacchettate sottoforma di file WAR;
- /work : è una directory in cui temporaneamente vengono riportati alcuni file ad uso delle applicazioni, come ad esempio le pagine JSP compilate.

7.4.2 Avvio del server Tomcat StandAlone

Una volta installato e configurato, Tomcat deve essere avviato mediante i file eseguibili presenti nella directory TOMCAT_HOME/bin.

Lanciando il file startup.bat (o l'equivalente startup.sh per piattaforme Unix) il server viene avviato. A questo punto è possibile avviare un browser e puntarlo sull'URL di esempi http://localhost:8080/examples per testare la corretta insatallazione di Tomcat. Se tutto è andato a buon fine dovrebbe comparire l'applicazione Web di esempio che viene fornita assieme al server.

7.4.3 Configurazione e installazione di Tomcat con IIS

Considerato che IIS non è in grado di eseguire Servlets e Java Server Pages (JSPs), può capitare di dover configurare Tomcat in modo da collaborare con IIS. Installando opportuni filtri in IIS e opportuni file in Tomcat è possibile ridirigere servlet e JSP request da IIS a Tomcat le quali saranno gestite da quest'ultimo e fornite al client.

Come pubblicato sul sito di Apache http://jakarta.apache.org/tomcat/tomcat-4.1-doc/jk2/jk/iishowto.html IIS-Tomcat redirector sono stati sviluppati e testati su:

- WinNT4.0-i386, Win98, Win2K, WinXP.
- IIS 4.x,5.x
- Tomcat3.2.x, Tomcat3.3.x, Tomcat 4.0.x, Tomcat 4.1.x, Tomcat5.

In questa tesi si è configurato IIS-Tomcat redirector per WinXP professional, Tomcat4.1.18, IISv.5.

Innanzitutto partiamo dal presupposto di avere installato IIS server sulla porta 80 (testarne il funzionamento andando su http://localhost) e Tomcat in modalità stand Alone sulla porta 8080 (vedi paragrafo precedente). Per installare gli ISAPI Redirector si necessita di quattro file disponibili in rete al sito http://www.getnet.net/~rbarr/TomcatOnIIS/default.htm.

- isapi_redirect.dll IIS server plug-in (disponibile anche sul sito Jakarta)
- workers.propertie[9] un file che descrive gli Host(s) e le porte usate da Tomcat .
- uriworkermap.properties un file che mappa gli indirizzi URL che dovranno essere processati da Tomcat (disponibile un esempio alla fine del paragrafo).
- iis_redirctor.reg per registrare il file isapi_redirect.dll sotto windows (disponibile una versione di esempio alla fine del paragrafo)

[9] Questi quattro file sono stati scaricati dal sito http://www.getnet.net/~rbarr/TomcatOnIIS/default.htm

l'installazione include due parti fondamentali:

- Configurare ISAPI Redirector default e quindi testare il funzionamento degli esempi inclusi in Tomcat chiedendo di processarli al server IIS
- Aggiungere, nella configurazione di tomcat, nuovi URL (oltre alla directory di esempio) in modo da poter aggiungere direttori virtuali da gestire con l'ausilio di Tomcat tramite la ridirezione.

Configurare ISAPI Redirector:

Supponiamo di aver copiato il file isapi_redirect.dll in <Tomcat_home>/bin, i file di properties nel direttorio <Tomcat_home>/conf/ntiis opportunamente creato ed il file iis_redirector.reg in <tomcat_home>/conf.

Editare il file workers.properties

- Settare la variabile *workers.tomcat_home* al valore del Catalina directory (per esempio: <Tomcat_home>)
- Settare la variabile workers.java_home al direttorio contenente la java virtual machine (per esempio <java_home>/bin).

Editare il file isapi.redirector.reg

- Log_file deve puntare a <Tomcat_home>\\logs\\iis_redirector.log
- Worker_file deve puntare a <Tomcat_home>\\conf\\ntiis \\workers.properties
- Worker_mount_file deve puntare a <Tomcat_home>\\conf\\ntiis \\uriworkermap.properties
- Salvare il file e caricarlo (mandarlo in esecuzione, cliccando due volte sopra il nome del file carica il file di registry in windows).

Configurare IIS

- Aprire IIS Admin e il web site di default
- Aggiungere un virtual directory di nome "jakarta", farla puntare al direttorio <tomcat_home>/bin e dargli i diritti "executable". (Per aggiungere una virtual directory, cliccare con il tasto destro del mouse sopra al web-site di default, scegliere "New", scegliere "Virtual Directory" e settare il nome ed il direttorio).
- Aggiungere <tomcat_home>/bin/isapi_redirect.dll come filtro ISAPI.

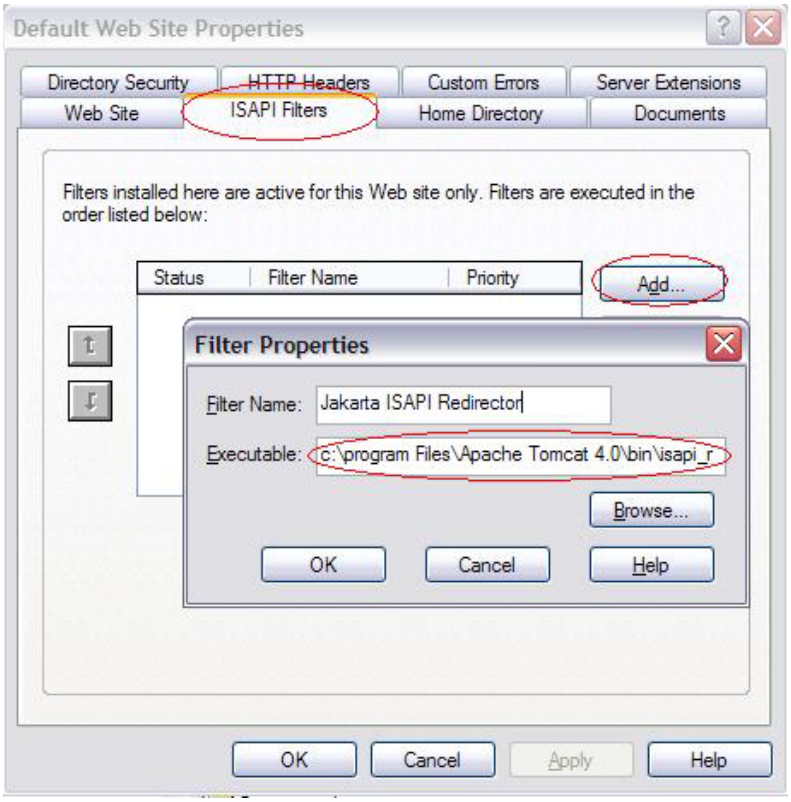


Figura 70: - IIS ISAPI Filter -

Per aggiungere questo filtro occorre cliccare con il pulsante destro sul web site di default e scegliere proprietà. Selezionare aggiungi Filtro ISAPI e cliccare su Aggiungi. Selezionare il filtro isapi_redirector.dll e salvare. Fare il reboot del sistema, riaprire IIS Admin, cliccare col tasto destro del mouse sul web site e controllare che nel ISAPI Filters Tab il filtro Jakarta sia stato attivato e che quindi abbia una freccia verde di fianco). Se il filtro è stato attivato allora dovrebbe funzionare e per testare basta aprire il browser e chiedere alla porta 80 (server IIS) di aprire gli esempi di Tomcat: http://localhost/examples/jsp/dates/date.jsp.

Aggiungere una virtual directory personalizzata

Ora che l'ISAPI Filter è correttamente installato è possibile processare le pagine JSP attraverso il server IIS a patto che queste siano state inserite in opprtune virtual directory. Per inserire files JSP in virtual directory occorre completare altri due step:

- Definire la directory per Tomcat, in modo che le risorse siano accessibili via:http://localhost:8080/my -directory
- Definire il nuovo direttorio nell'ISAPI Redirector filter, in modo da rendere le risorse accessibili via: http://localhost/my-directory

STEP 1: Per aggiungere una virtual directory in Tomcat, occorre modificare il file server.xml allocato nel direttorio <*Tomcat-home*>/Conf ed aggiungere un simple element xml.

Posizionarsi all'interno del file dentro Il Server element (<SERVER…)

Quindi portarsi all'interno del service element (<SERRVICE….)

Poi all'interno del engine element (<ENGINE….)

Poi all'interno dell'host element (<ELEMENT….)

Una volta dentro a questi element xml aggiungere un simple context element come quello sotto riportato:

<Context path="/my-directory" docBase="C:/my-directory" debug="0" />

Dove *path* è il nome del direttorio virtuale (con il quale richiamare la risorsa attraverso il browser e *docBase* punta al direttorio fisico.

Quando questo lavoro è fatto correttamente le risorse posizionate nel direttorio my-directory saranno visualizzabili via http://localhost:8080/my-directory.

STEP 2: Per definire il direttorio all'interno nell'ISAPI Redirector occorre aprire il file uriworkermap.properties posizionato nel direttorio <tomcat_home>/conf/ntiis ed aggiungere due linee del tipo:

/my-directory=$(default.worker)
/my-directory/*=$(default.worker)

Dopo aver salvato questo file fare il reboot della macchina. All'avvio sarà possibile visualizzare le risorse posizionate in c:/my-directory via http:://localhost/my-directory.

```
#
# Default worker to be used through our mappings
#
default.worker=ajp13

#
# Sites to be redirected to Tomcat
#
/examples=$(default.worker)
/examples/*=$(default.worker)
/webdav=$(default.worker)
/webdav/*=$(default.worker)
/tomcat-docs=$(default.worker)
/tomcat-docs/*=$(default.worker)
/manager=$(default.worker)
/manager/*=$(default.worker)
```

Figura 71: - Codice di esempio uriworkermap.properties –

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\Jakarta
Isapi Redirector\1.0]
"extension_uri"="/jakarta/isapi_redirector.dll"
"log_file"="C:\\Program Files\\Apache Tomcat
4.0\\logs\\iis_redirector.log"
"log_level"="emerg"
"worker_file"="C:\\Program Files\\Apache Tomcat
4.0\\conf\\ntiis\\workers.properties"
"worker_mount_file"="C:\\Program Files\\Apache Tomcat
4.0\\conf\\ntiis\\uriworkermap.properties"
```

Figura 72: - Codice di esempio isapi_redirector.reg -

7.5 Jetspeed

7.5.1 Configurazione e installazione di Jetspeed

Jetspeed è l'implementazione di un EIP scritto completamente in Java ed XML disponibile in modalità OpenSource on line su Jakarta, un sito che distribuisce una numerosissima quantità di progetti OpenSource e freeware. La versione 1.4b3 installata in questa particolare architettura è scaricabile sia come file sorgente che come file già compilato e quindi come pacchetto war al sito http://jakarta.apache.org/build/jakarta-jetspeed/release/v1.4b3 .

Per l'installazione a partire dal codice sorgente (jetspped-1.4b3-src.zip), occorre :

- decomprimerlo all'interno di un direttorio dedicato che indicheremo col nome di<*jetspeed_home*>
- configurare il file di build: *build.xml* posto nel direttorio <*jetspeed_home*>/build. (Nel caso della versione di jetspeed 1.4b3 installata, non è stato necessario modificare tale file di configurazione)
- aprire un finestra shell di dos, portarsi nel direttorio <*jetspeed_home*>/*build* e lanciare il comando *ant war* il quale genererà un file di nome *jetspeed.war* nel direttorio di <*jetspeed_home*>/*bin*.
- copiare il file jetspeed.war nel direttorio *webapps* di tomcat.

Per l'installazione a partire dal codice binario (jetspped-1.4b3-war.zip), basta :

- scompattare il file jetspeed-1.4b3-war nel direttorio *webapps* di tomcat.

7.5.2 Avvio di Jetspeed

Una volta eseguita l'installazione, basterà attivare, come descritto precedentemente, Tomcat in modo stand alone e quindi avviare un web browser puntandolo sull'URL di default: http://localhost:8080/jetspeed. Se tutto è andato a buon fine dovrebbe comparire un esempio di default di un EIP costruito con jetspeed. In tale portale, sono inoltre già registrati due ipotetici account, con i quali poter sperimentare tipologie di accesso differenziate al portale.

- **Utente1**: Login: turbine **Password**: turbine
- **Utente2**: Login: admin **Password**: jetspeed

Una volta avviato il server, il pacchetto *Jetspeed.war* salvato nella webapps directory di Tomcat, creerà un direttorio di nome Jetspeed, contenete altri sottodirettori, di fondamentale importanza per la configurazione del portale e dei portlet. Vediamo in dettaglio i principali sottodirettori contenuti in *<Tomcat_home>/webapps/Jetspeed/:*

- /WEB INF : contiene tutti i file necessari all'esecuzione del portale.
- /WEB INF/PSML: contiene i file di configurazione dei singoli utenti, dei gruppi di utenti e della configurazione della pagina iniziale di default del portale.
- /WEB INF/LIB: contiene le librerie Java in formato JAR che Jetspeed utilizza;
- /WEB INF/DB: contiene altre directory con files relativi al database Hypersonic fornito insieme a Jetspeed.
- /WEB INF/CLASSES : contiene tutti i package delle applicazioni e quindi dei portlet utilizzabili nel portale.
- /WEB INF/CONF: contiene i file di configurazione del portale
- /WEB INF/LOG: contiene i vari file di log generati dal server ;
- /WEB INF/TMP: è una directory di lavoro che contiene file temporanei;

7.5.3 File di configurazione di Jetspeed

Una volta installato Jetspeed, è possibile configurare il portale utilizzando appositi file posizionati nella sottocartella /WEB-INF/CONF. Analiazziamo in questi capitoli come configurare le principali proprietà di Jetspeed.

Creare un Custom Porperty file

E' possibile configurare per qualsiasi sito specifiche d'ambiente personalizzate, creando un apposito custom property file per esempio: *my.properties* e salvarlo in */WEB-INF/conf*. Creare un file di configurazione personalizzato permette aggiornamenti futuri di Jetspeed più pratici, in quanto i file di configurazione di default non vengono modificati equindi una eventuale loro modifica non interviene direttamente sulle configurazioni del portale. Tale particolarità è stata aggiunta a partire dalla versione 1.4b2. Un esempio di configurazione personalizzata viene riportata qui sotto e si nota come vengano soprascritti dei parametri contenuti nei file

TurbineResources.properties,
JetspeedResourece.properties,
JetspeedSecurity.properties,

```
#
########################################################################
##
#
#  T u r b i n e R e s o u r c e s . p r o p e r t i e s :
#
#
########################################################################
##

# ----------------------------------------------------------------
---
#
#  L O G S
#
# ----------------------------------------------------------------
---
services.LoggingService.default = debug

# ----------------------------------------------------------------
---
#
#  S E R V I C E S
#
# ----------------------------------------------------------------
---
services.ResourceService.classname =
org.apache.jetspeed.services.resources.JetspeedResourceService

#
########################################################################
##
#
#  J e t s p e e d R e s o u r c e s . p r o p e r t i e s :
#
#
########################################################################
##

########################################
# Registry Service                     #
########################################
services.Registry.refreshRate = 60

########################################
# Portlet Usage Service                #
```

```
########################################
services.PortletStats.enabled = true

########################################
# Customization
########################################
customizer.preview.enable = true

########################################
# New User Registration mail support    #
########################################
automatic.logon.enable = true

#
######################################################################
###
#  J e t s p e e d S e c u r i t y . p r o p e r t i e s :
#
#
######################################################################
##

services.JetspeedSecurity.password.expiration.period = 90

# ------------------------------------------------------------
---
#
#  A D D I T I O N A L  P R O P E R T I E S
#
# ------------------------------------------------------------
---
# The full path name to an additional properties file. Properties
in
# this file will be included in this property set. Duplicate name
# values will be replaced, so be careful.
#
# Default: none
# ------------------------------------------------------------
---

include = TurbineResources.properties
```

Figura 73: - File di configurazione Jetseed personalizzato -

- Il default logg-In è settato a “debug”,
- il file di registry viene aggiornato ogni minuto,
- il servizio di logg-In dei portlet è attivato
- viene abilitata la funzione di preview dei portlet in modalità di customizzazione,
- viene abilitato il logon automatico
- le password devono essere cambiate ogni 90 giorni.

Importante notare come fra questi elementi di configurazione , ce ne siano due obbligatori:

- service.ResourceService.classname= org.apache.jetspeed.services.resources.JetspeedResourceService
- include = TurbineResources.properties

Il primo comando è quello che permette la realizzazione di un file di configurazione personalizzato, mentre il secondo include nel file di personalizzazione i file di default di Turbine.

Per far in modo che Jetspeed legga tale file di configurazione all'avvio, occorre modificare il file il web app descriptor (*web.xml*) posto nel sottodirettorio *WEB-INF/* come sotto riportato:

```
<web-app>
    <servlet>
        <servlet-name>
           jetspeed
        </servlet-name>

        <servlet-class>
            org.apache.turbine.Turbine
        </servlet-class>

        <init-param>
            <param-name>properties</param-name>
            <param-value>WEB-INF/conf/my.properties</param-value>
        </init-param>

        <init-param>
              <param-name>resources</param-name>
            <param-value>
org.apache.jetspeed.services.resources.JetspeedResourceService
</param-value>
        </init-param>
    </servlet>
    ....
</web-app>
```

Figura 74: - Web app descriptor -

Nei file *.properties viene data la possibilità di utilizzare $(variable) in sostituzione dei parametri costanti: per esempio è possibile inserire assegnare un valore alla variabile confRoot e poi riutilizzarlo con $(confRoot).

```
confRoot = /WEB-INF/conf
...
services.URLManager.url = ${confRoot}/datasources.properties
...
services.Registry.mapping=${confRoot}/registry.xml
...

oppure

defaultRefresh = 60
...
services.Registry.refreshRate = ${defaultRefresh}
...
refresh.portlet.default = ${defaultRefresh}
```

Configurare il Layout e le barre di navigazione

L'aspetto dei portali sviluppati con Jetspeed è controllato dai Layout e dai Navigation Template, mentre un Layout Manager viene utilizzato per la generazione del portale. Jetspeed supporta 2 tipologie di Layout Manager una sviluppata in JSP e l'altra in Velocity. Entrambe le tipologie di Layout compiono le stesse operazioni e la decisione di quale utilizzare dipende solo dalle preferenze dei programmatori i quali, una volta deciso, dovranno specificare la scelta nel parametro:

services.TemplateService.default.extension
nel file *TurbineResource.properties*

Il Layout sviluppato in JSP (.JSP) viene utilizzato nella versione 1.3b1 di Jetspeed, mentre le successive versioni (1.3a2, 1.4b1, 1.4b2, 1.4b3) utilizzano di default un Layout sviluppato con Velocity (.vm).

I file utilizzati da Velocity Layout Manager sono:

- /WEB INF/Template/vm/navigation*language/country*/default.vm Per pubblicare il logo nell'angolo alto sinistro del portale, e le barre top,bottom e left di navigazione.

- /WEB INF/Template/vm/navigation/*language/country*/top.vm: definisce la top navigation bar a seconda che l'utente abbia superato o meno la fase di autenticazione nel portale.

- /WEB INF/Template/vm/navigation/*language/country*/left.vm: definisce la left navigation bar per tutti gli utenti.

- /WEB INF/Template/vm/navigation/*language/country*/bottom.vm definisce la barra di navigazione bassa, posta a fondo pagina comune a tutti gli utenti.

Analizzando questi file è possibile individuare come i valori che vengono esposti sulle barre di navigazione ed i settaggi di configurazione siano recuperati attraverso particolari variabili che devono essere opportunamente settate nel file, posto nel sottodirettorio: WEB-INF/conf, di nome *JetspeedResources.properties.*

```
########################################
# Navigation Bar customization         #
########################################
# Top navigation bar
#   topnav.enable    - Display the left navigation bar
#   topnav.vm        - VM file name for the top nav, in
templates/vm/navigations/html
#   topnav.logo.file - file name of the logo relative to
<jetspeed_home>. Do not use with logo.url
#   topnav.logo.url  - URL of logo. Useful when using a common
company logo that is on a different server
#   topnav.user_login.enable - Display login prompts on
navigation bar.  If false then login nust be via login portlet
#   topnav.user_creation.enable - Display "create user" prompts
on navigation bar.  Requires topnav.user_login.enable=true
topnav.enable=true
topnav.vm=top.vm
topnav.logo.file=images/jetspeed-logo.gif
topnav.logo.url=
topnav.user_login.enable=true
topnav.user_creation.enable=true

# Left Navigation bar
#   leftnav.enable    - Display the left navigation bar
#   leftnav.vm        - VM file name for the left nav, in
templates/vm/navigations/html
#   leftnav.width     - Keep the left edge of the content from
moving as the width of the content varies
leftnav.enable=true
leftnav.vm=left.vm
leftnav.width=10%

# Bottom Navigation bar
#   bottomnav.enable    - Display the Bottom navigation bar
#   bottomnav.vm        - VM file name for the bottom nav, in
templates/vm/navigations/html
bottomnav.enable=true
bottomnav.vm=bottom.vm
```

Figura 75: -Configurazione delle navigation bar attraverso il file JetspeedResoureces.properties -

Dallo stralcio di codice riportato, si vede come sia possibile:

- Attivare o disattivare le barre attraverso le rispettive configurazioni di:
  topnav.enable=true/false
  leftnav.enable=true/false
  bottomnav.enable=true/false
- Modificare il logo del portale attraverso la modifica della variabile tonav.logo.file,
- Attivare o disattivare la sezione di riconoscimento del log-IN sulla top-bar oppure attivare o disattivare il form di login attraverso i rispettivi comandi:
  topnav.user_login.enable=true/false
  topnav.creation_login.enable=true/false
- Decidere la larghezza (in percentuale della left bar) con leftnav.width=0%-100%

Ed altri servizi di Layout riguardo le barre di navigazione.

Localizzazione

Ognuna delle configurazioni sopra riportate può essere adattata a seconda della lingua e del paese del client che visualizza il portale. Basti osservare i direttori sopra citati per notare che il direttorio *language* e quello *country* erano scritti in corsivo, questo per sottolineare il fatto che a seconda delle caratteristiche del client è possibile personalizzare le barre di navigazione. Dettagliatamente, si possono personalizzare tutti i direttori come actions, layouts, screens, navigation, pages, che sono definiti nel root path directory dei moduli gestiti da Turbine. Nel file di configurazione TurbineResources.properties è possibile configurare un default language ed un default country.

```
# ------------------------------------------------------------
----
#
#  L O C A L I Z A T I O N  S E R V I C E
#
# ------------------------------------------------------------
----

# Default ResourceBundle and language/country codes used by the
# TurbineLocalizationService.
#
# this bundle is searched first - so name your override file here
#locale.default.bundle=
#
# this is a comma separated list of bundles that are searched to
find a resource
# after the above default bundle has been checked, if its
specified
# should be fine to leave this setting as is

locale.default.bundles=org.apache.jetspeed.modules.localization.J
etspeedLocalization
locale.default.language=en
locale.default.country=US
```

Figura 76: - Stralcio di TurbineResource.properties –

Analogamente è possibile (ma non obbligatorio) configurare i file PSML relativi ai gruppi ed agli utenti con la stessa metodologia ed in questo caso i file vengono riposti in sottodirettori basati sui “language & ccountry code abbrevation”. Tali abbreviazioni rispettano le specifiche ISO-639 per il language code, mentre le specifiche ISO-3166 per l’uso del language ed il country code abbrevation contemporaneamente.

```
user
   |-- david
        |-- html
             |-- fr             // french language
```

```
                    |-- FR        // France country-code
                    |-- BE        // Belgium country-code
```

Figura 77: - Esempio di PSML Localization-

Configurare un portlet nel file di registry con il PSML

I nuovi portlet compilati e memorizzati nei relativi package della web application, devono essere configuranti manualmente con l'utilizzo del Portal Structure Markup Language (PSML): una sorta di dialetto dell'XML. Jetspeed utilizza PSML per descrivere la configurazione interna, i portlet disponibili e le configurazioni degli utenti. I portlet disponibili in Jetspeed vengono dichiarati in un portlet registry tramite l'utilizzo di PSML, in particolare ogni portlet viene dichiarato attraverso un "portlet entry element" che dice al portale come istanziare tale servizio. Ogni portlet entry deve avere una classe da cui ereditare e deve specificare nel particolare in che modo il portlet deve essere istanziato:

- *Instance*: Questa tipologia di entry portlet rappresenta uno stand alone portlet, che ha tutte le informazioni che gli occorrono per la sua esecuzione.
- *Abstract*: L'abstract portlet invece non possiede tutte le informazioni che gli occorrono, quindi un *ref* portlet deve provvedere a fornirgli i dati mancanti.
- *Ref*: Un ref portlet costruisce sopra un altro portlet entry, è quindi possibile creare delle catene di ref portlet che si costruiscono una sull'altra, aggiungendo informazione.

Il file di configurazione di tutti i portlets utilizzabili nel portale è *demo-portlets.xreg* ed è posizionato nella WEB-INF/conf directory. Ogni utente possiede invece due file di configurazione personali nei quali vengono indicati quali portlet sono utilizzati o sono stati settati per quel determinato utente. I file di configurazione si chiamano entrambi default.psml ma sono memorizzati in due diversi sottodirettori relativi all'utente: rispettivamente in WEB-INF/PSML/<username>/HTML per la configurazione dei portlet attraverso browser HTML e WEB-INF/PSML/<username>/WML per la configurazione di quelli raggiungibili attraverso tecnologia WAP.

Per chiarezza riportiamo un esempio di configurazione di un portlet nel file di registry.

```
<?xml version="1.0" encoding="UTF-8"?>
<registry>
    <portlet-entry name="HelloWorld" hidden="false"
type="instance" application="false">
        <meta-info>
            <title>HelloWorld</title>
            <description>Secondo Esempio di portlet
Luca</description>
        </meta-info>

<classname>it.gruppopro.jportal.portlets.iframe.HelloWorld;</clas
sname>
        <media-type ref="html"/>
        <url cachedOnURL="true"/>
    </portlet-entry>
</registry>
```

Figura 78: - Esempio di configurazione di un portlet con PSML -

Name=”HelloWorld” determina il nome del portlet

Hidden=”false” indica che il portlet è visibile

Type=”instance” indica che è un portlet stand alone

<classname>it.gruppopro.jportal.portlets.iframe.HelloWorld;</classnameIndica da quale classe viene istanziato il portlet

<media-type ref="html"/> Indica quali media type sono supportati

Oltre a queste informazioni di base è possibile aggiungere anche un titolo al portlet ed una descrizione attraverso l’uso dei <meta-info> , <title> e <description> tag.

Configurare la pagina di default in Jetspeed

Considerata la mancanza di tool amministrativi del portale, anche la semplice configurazione della pagina di default nella quale tutti gli utenti non autenticati hanno accesso, risulta una operazione che deve essere eseguita manualmente. Il file di configurazione che descrive i portlet integrati nella pagina di default e la loro disposizione è posto nel direttorio /WEB-INF/psml/user/anon/html/default.psml. Per poter modificare questo file, o si interviene manualmente con l’uso del PSML oppure il tutorial di Jetspeed consiglia di avviare il portale, creare un utente di comodo che chiameremo *comodo*, adattare la pagina di questo utente fittizio (con l’utilizzo della visualizzazione di customize) a seconda delle esigenze e quindi copiare il file creato da jetspeed nel direttorio /WEB-INF/psml/user/comodo/html/default.psml al posto di quello posizionato nel direttorio prima citato /WEB-INF/psml/user/html/anon/ e quindi riavviare il portale. Se tutte le operazioni sono eseguite correttamente si otterrà una nuova pagina di default.

Come detto precedentemente è possibile anche intervenire manualmente sul file default.psml posto in WEB-INF/psml/user/html/anon. In questo caso occorre analizzare i vari elementi del PSML che servono per configurare le pagine del portale e il loro layout. Il risultato di un file di configurazione viene riportato sotto:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<portlets id="100"
xmlns="http://xml.apache.org/jetspeed/2000/psml">

    <metainfo>
        <title>Welcome Page</title>
    </metainfo>
    <control name="TabControl"/>
    <controller name="CardPortletController"/>

    <portlets id="101">
        <metainfo>
            <title>Basic Tutorials</title>
        </metainfo>
    </portlets>

    <portlets id="102">
        <metainfo>
            <title>Advanced Tutorials</title>
        </metainfo>
    </portlets>

    <portlets id="103">
        <metainfo>
            <title>Jetspeed Portlets</title>
        </metainfo>
    </portlets>

    <portlets id="104">
       <controller name="OneColumn"/>

        <metainfo>
            <title>Referenced Portlets</title>
        </metainfo>
        <reference path="group/apache/media-
type/html/page/default"/>
        <reference path="group/Jetspeed/media-
type/html/page/default"/>
    </portlets>

</portlets>
```

Figura 79: - File di Configurazione della pagina di default -

Nel file di configurazione compaiono i seguenti tag:

- <*portlets*> definisce una nuovo pane all'interno del portale
- <*controller*> definisce una tipologia di layout .
- <*control*> definisce un tipo di menù

Per ogni opzione di menù occorre definire un pane ossia una raccolta di portlets che verranno visualizzati insieme. Anche se, i portlets vengono raccolti in uno stesso pane, possono comunque avere layout personalizzato. Un pane viene specificato come riportato sotto ed il <title> tag viene usato per visualizzare un titolo al pane.

```
    <portlets id="101">
        <metainfo>
            <title>Basic Tutorials</title>
```

```
        </metainfo>
    </portlets>
```

Figura 80: Definizione di un Pane –

I *Control*s sono decorazioni che si possono applicare attorno ad un pane o ad un singolo portlet; Jetspeed prevede due tipologie di controls per i pane:

- *TabControl:* che posiziona le opzioni del menù nella parte alta della pagina
- *MenùControl:* che posiziona le opzioni del menù sul lato sinistro della pagina.

I *controllers* definiscono invece il layout dei pane nel senso che decidono il numero di colonne e di righe in cui questi devono essere suddivisi. Nel caso di layout dei portlets raccolti in uno stesso pane, il controller può assumere i seguenti valori:

- One Coloumn – Tutti portlets vengono posizionati uno sotto all'altra in un'unica colonna del pane.
- Single Row – Tutti i portlets definiti nel pane vengono affiancati su una stessa riga.
- Three Column – I portlet vengono posizionati su tre colonne del pane ed è possibile selezionare anche in percentuale quanta larghezza del pane ogni colonna occupa: (25/50/25) oppure (33/33/33).
- Two Column – I portlets vengono posizionati su due colonne del pane ed è possibile selezionare la larghezza , in percentuale, occupata da oni colonna: (50/50) oppure (25/75) oppure (75/25).

Quindi occorrerà definire all'interno di un pane anche i relativi portlets da caricare e con quale tipologia di layout posizionarli. Il risultato della definizione completa del pane 103 potrebbe quindi essere:

```
<portlets id="103">
        <metainfo>
            <title>Jetspeed Portlets</title>
        </metainfo>

        <controller name="TwoColumns"/>

        <entry parent="BBCFrontPage">
            <layout>
                <property name="column" value="0"/>
                <property name="row" value="0"/>
            </layout>
        </entry>

        <entry parent="WeatherPortlet">
            <layout>
                <property name="column" value="0"/>
                <property name="row" value="1"/>
            </layout>
            <parameter name="weather_city_info"
value="US/AZ/Phoenix"/>
            <parameter name="weather_style" value="infobox"/>
        </entry>
    </portlets>
```

Figura 81 : -Configurazione di un pane –

Configurare la pagina di default per un nuovo utente

La pagina di default che viene assegnata quando un nuovo utente si aggiunge (registrandosi) al portale viene copiata dal file /WEB-INF/psml/user/turbine/html/default.psml, quindi per modificare la pagina di default per un nuovo utente occorre comportarsi come per la modifica dell pagina di default (descritta) nel paragrafo sopra, con la sola accortezza di modificare il file default.psml posto nel direttorio /WEB-INF/psml/user/turbine/html/ invece che quello posto in /WEB-INF/psml/user/anon/html/.

Capitolo 8

Sviluppo e deployment di un caso di studio : la facoltà di Ingegneria

A tutti gli effetti il sito della facoltà di Ingegneria di Modena può essere paragonato ad un qualsiasi sito aziendale nel quale, varie tipologie di utenti, utilizzano le informazioni ed i servizi messi a disposizione dalla facoltà a seconda dei relativi diritti di accesso. Il sito della facoltà presenta quindi, sparsi fra le pagine del sito web, vari servizi come la posta elettronica, la possibilità di inserire news di facoltà, news dirette agli studenti, accesso alla rete intranet, accesso alla personalizzazione delle pagine docenti, che rappresentano un insieme di funzionalità utili ma sparse nel contesto del portale. Un docente per esempio che voglia inserire una news nel sito della facoltà di Ingegneria deve quindi portarsi sul sito relativo alla facoltà, cliccare sul pulsante Login, inserire i dati di Autenticazione, fare il submit e quindi accedere al servizio. Nel caso in cui, sempre lo stesso ipotetico docente, voglia modificare anche i dati relativi alla pagina web personale, deve portarsi su un secondo sito dedicato, ripetere un'altra fase di autenticazione (magari con nome utente e password diverse) e quindi accedere al servizio. Basti pensare che tali operazioni possono essere di routine e iterate più volte in uno stesso giorno per immaginare la scomodità del meccanismo.

Per quanto analizzato un EIP applicato al contesto della Facoltà di Ingegneria potrebbe rappresentare una comoda soluzione per accedere alle informazioni ed ai servizi di ateneo in modo personalizzato e immediato.

Scopo di questo caso di studio è quindi di integrare in un EIP il sito della facoltà di Ingegneria e creare, per ogni utente, un' unica pagina di accesso ai servizi che possa essere personalizzata a seconda delle singole esigenze. In particolare si è previsto di sviluppare una applicazione che, opportunamente settata, possa fare il login automaticamente ai singoli servizi, in base ai dati di accesso dei singoli utenti. L'applicazione sviluppata, richiede l'inserimento di alcuni dati relativi al servizio (come per esempio nome utente, password, protocollo di comunicazione, url relativo al servizio…) durante la prima connessione. Nelle successive fasi di Login al portale il portlet creato permetterà all'utente di accedere ai servizi da lui onfigurati durante le sessioni precedenti in modo trasparente, ossia senza dover intervenire direttamente durante le fasi di autenticazione.

In questo capitolo viene presentata l'analisi e l'implementazione del portlet atto a svolgere il servizio sopra citato, mettendo in evidenza le difficoltà ed i limiti che si possono incontrare nel lavorare con un prodotto OpenSource quale Jetspeed.

8.1 Jetspeed Portal Security

Lo scopo del portal Security è quello di autenticare gli utenti al portale ed autorizzare loro l'accesso ad eventuali sorgenti di dati o ad eventuali servizi. Gli utenti devono potersi registrare o essere registrati da un amministratore ed, a seconda dei ruoli a loro assegnati, vengono attivati o meno gli accessi alle varie risorse. Vediamo gli oggetti che gestiscono la sicurezza in Jetspeed:

Interfaccia	Descrizione
JetspeedUser	Definisce gli attributi minimi di un utente nel sistema portale
Role	Definisce gli attributi minimi dei ruoli nel sistema portale
Group	Definisce gli attributi minimi di un gruppo nel sistema portale
Permission	Definisce gli attributi minimi dei permessi nel sistema portale

Il deployment di default di Jetspeed include già Hypersonic SQL database nel quale vengono definiti alcuni utenti, ruoli, gruppi, e permessi per facilitare un primo approccio alla gestione del portale, ma viene anche incentivato il cambio del database, verso un prodotto più robusto, nel caso si dovessero creare soluzioni da commercializzare. Nel caso di studio intrapreso, ci siamo accontentati del database fornito dal deployment di default di Jetspeed. Il default security service di Jetspeed utilizza Jakarta [Torque] ma a partire dalla versione 1.4b3 di Jetspeed, installata in questa tesi, si è passati alla tecnologia delle LDAP Security Service. Connettendosi come amministratore del portale: (nomeUtente=*Admin*, password=*Jetspeed*) è possibile verificare come Jetspeed fornisca un buon numero di portlet atti alla gestione di utenti, gruppi, ruoli e permessi. Nella sezione Security dell'amministratore, vengono infatti forniti comodi tool per la gestione delle security:

- User Browser: permette di aggiungere/modificare/cancellare utenti e associare a questi relazioni fra User-Group e User-Role.
- SecurityRole/Group/Security Browser : forniscono aggiunta/cancellazione/modifica per la gestione dei ruoli, gruppi e permessi.

Le associazioni fra User e Group non vengono utilizzate di default dal Security System, ma tali associazioni possono essere create per instaurare nel portale dei security check in funzione dei ruoli. Per esempio si possono creare dei gruppi di utenti, come nel nostro caso docenti, che possono accedere a servizi vietati invece al gruppo studenti.

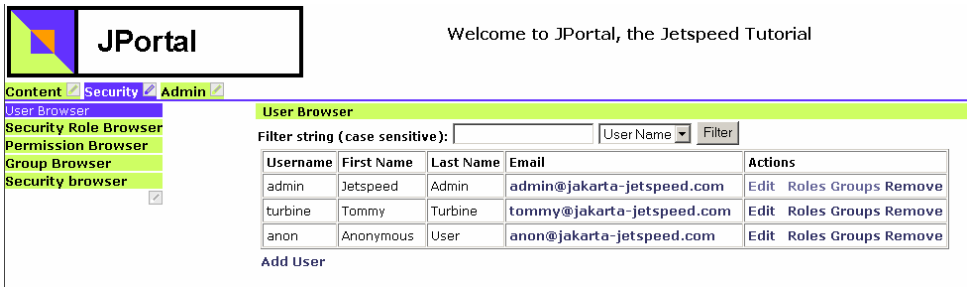


Figura 82: - Portlet per la gestione Utenti -

Per proteggere eventuali risorse con security constranits, occorre intervenire nel Security registry *security.xreg* posizionato nel sottodirettorio */WEB-INF/conf/*. Nell'esempio riportato sotto vengono rilasciati tutti i diritti di accesso agli utenti con ruolo account manager o admin, ma si limita il diritto di accesso alla sola lettura agli utenti con ruolo guest.

```
<security-entry name="requires-accountManager">
        <meta-info>
            <title>Account Manager</title>
            <description>Grant full access to Account
Manager Role, read access to Support Role.</description>
        </meta-info>
        <access action="*">
            <allow-if role="accountManager"/>
            <allow-if role="admin"/>
        </access>
        <access action="view">
            <allow-if role="guest"/>
        </access>
    </security-entry>
```

Figura 83: - Esempio di assegnazione dei diritti di accesso ad un portlet -

8.2 HTTP e Autenticazioni

Quando un web client richiede ad un servizio web l'autenticazione di un utente, viene mandata una richiesta HTTP verso un url, che rappresenta un programma eseguibile, con l'aggiunta di uno Username, una Password ed eventuali parametri extra che chiameremo parametri estesi. In base al posizionamento di tutti questi parametri, nel contesto della richiesta http, nascono varie tipologie di autenticazioni. In base alla tipologia di richiesta http, sono state analizzate le varie tipologie di autenticazione:

TIPO DI AUTENTICAZIONE	HTTP REQUEST GET	HTTP REQUEST POST
NO Authentication	Invio dei soli parametri estesi URL?name1=value1&name2=value2....	Invio dei soli parametri estesi attraverso una form opportunamente settata.
BASIC Authentication	il nome utente e password prima dell'url: Username:Password@URL?name1=value1&name2=value2....	XXXXXXXXX
FORM Authentication	Username e Password seguono l'url URL?j_Username=Username&J_password=password&name1=value1&name2=value2....	Invio della richiesta attraverso un form di autenticazione e quindi attraverso l'header della richiesta http

Figura 84: -HTTP e Autenticazioni –

8.2.1 http No Authentication (method get/post)

A volte una risorsa web non necessita di un vero e proprio riconoscimento dell'utente ma richiede solo di settare determinati parametri (parametri estesi), i quali dovranno essere spediti dal client al server web tramite una request http. In questo caso il client può inviare i parametri sia con il metodo get, e quindi inserendoli nella stringa url secondo questa struttura:

URL= url?name1=vallue1&name2=value2…

Oppure può inviarli al server web mediante una POST, ossia tramite la sottoscrizione di una form. In tal caso i parametri inviati non vengono inseriti direttamente nella stringa dell'URL ma vengono inseriti nell'Header della richiesta inviata dal client.

8.2.2 http Basic Authentication (method get)

Il meccanismo di autenticazione basic definito dalle specifiche http/1.1 si basa sull'invio di uno username una password ed eventuali parametri estesi. Questa tipologia di autenticazione è poco utilizzata e viene eseguita solitamente da parte del client con l'utilizzo del metodo get ossia, inserendo i parametri direttamente nella stringa dell'url secondo questa struttura:

URL= Username:Password@url?name1=vallue1&name2=value2…

Il server web controllerà la possibilità per l'utente di accedere al servizio o all'eventuale risorsa protetta e risponderà al client.

8.2.3 http Form Authentication (method get/post)

Anche il meccanismo di autenticazione Form si basa sull'invio di uno Username, una password ed eventuali parametri estesi. Questa è la tipologia di autenticazione più utilizzata e se il metodo della richiesta è una get allora tali parametri saranno inseriti dal client nella stringa URL secondo la seguente struttura:

URL=url?loginUserParamName=Username&
passwordparamName=password&name1=vallue1&name2=value2…

Ma se il client volesse rendere illeggibili i parametri inviati al server allora è consigliato l'uso di una post ossia di una richiesta generata attraverso la sottoscrizione di una form come quella rappresentata nella figura sotto, dove la action della form prenderà il valore dell'URL ossia del programma eseguibile che controllerà i valori spediti.

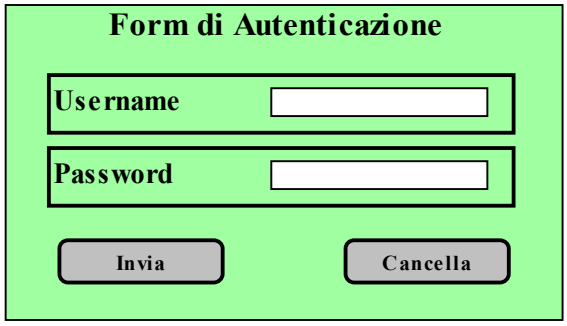


Figura 85: - Form di Autenticazione -

8.3 Implementazione dell'IFrameController

Il Portlet creato (chiamato IframeController) estende il metodo getContent del portlet IframPortlet fornito con il deployment di Jetspeed v.1.4b3 e ne sfrutta i metodi messi a disposizione. Quando il portlet controller invoca il metodo getContent() dell' IframeController questo verifica che l'utente sia loggato al portale. Se l'utente è anonimo allora viene configurato il valore inserito in UrlPublic come sorgente dell'Iframe Controller che sarà visualizzato nel Portlet. Nel caso in cui

l'utente si sia invece autenticato al portale vengono caricati tutti i parametri definiti nel registry e si verifica che l'utente abbia già settato nelle sessioni precedenti i parametri privati quali Username, Password, UrlPrivate e tipologia di Autenticazione. Se questi non sono ancora stati inseriti allora si pubblica nel portlet un Avviso statico, altrimenti a seconda del metodo http utilizzato dalla request si inizia l'autenticazione (trasparente all'utente) presso il sito indicato nel parametro UrlPrivate.

Nel caso di una richiesta ***http get*** viene creato un l'URL contenete i valori relativi all'autenticazione e viene settato come sorgente dell'Iframe[10].

Nel caso di una richiesta ***http post***, viene spedita una richiesta, contenente i dati dell'autenticazione, ad una pagina JSP di nome post.jsp posta nel direttorio */pro* della webApplication. Tale pagina preleva i dati contenuti nella richiesta inviata dal portlet e costruisce un form dinamicamente, ponendo nella action l'URL passato dal portlet nella variabile di nome sito. Tale pagina JSP sarà poi settata dal portlet come sorgente per l'Iframe. Quando l'utente dovrà accedere ad un servizio che richiede una autenticazione attraverso una richiesta http con metodo post, dovrà necessariamente cliccare su un pulsante che la pagina dinamica avrà creato il quale determinerà l'action del form dinamico.

[10] L'elemento IFRAME permette agli autori di inserire un frame all'interno di un blocco di testo. Inserire un frame in linea all'interno di una sezione di testo è molto simile ad inserire un oggetto attraverso l'elemento OBJECT: entrambi permettono di inserire un documento HTML nel mezzo di un altro, essi possono entrambi essere allineati con testo circostante, ecc.

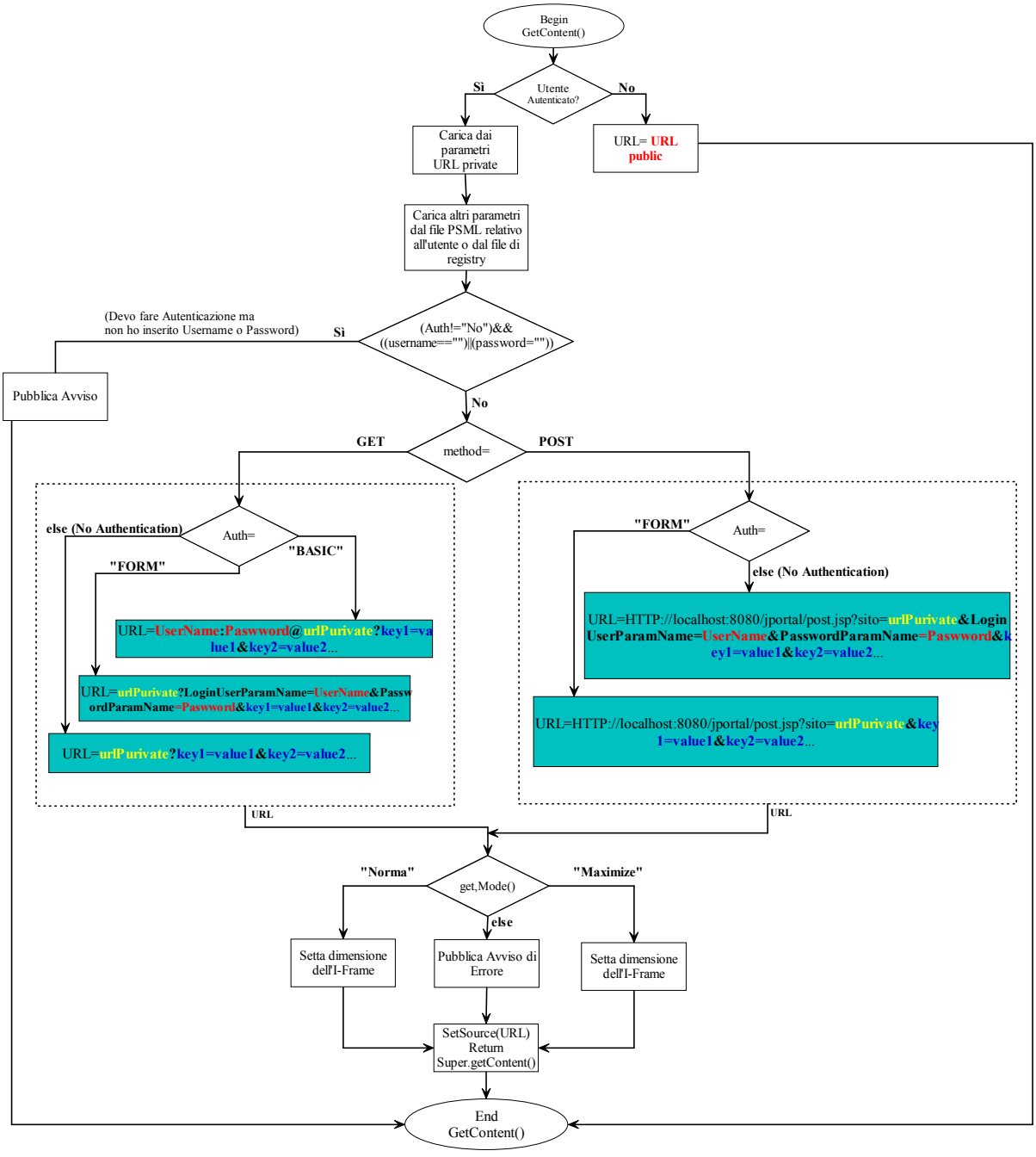


Figura 86: - Diagramma di Flusso dell' IFramePortlet -

8.3.1 Codice IframeController

IFrameController.java

Created with JBuilder

```
package it.gruppopro.jportal.portlets.iframe;

/**
 *
Title: IframeController.java

 *
Description:

 *
Copyright: Copyright (c) 2003

 *
Company:

 * @author Bonzagni Luca
 * @version 1.0
 */


import org.apache.jetspeed.portal.portlets.IFramePortlet;
import org.apache.turbine.util.RunData;
import org.apache.ecs.ConcreteElement;
import org.apache.ecs.StringElement;
import org.apache.jetspeed.om.security.BaseJetspeedUser;
import org.apache.jetspeed.services.rundata.JetspeedRunData;
import org.apache.jetspeed.portal.PortletConfig;
import java.util.*;
//
import org.apache.jetspeed.om.security.JetspeedUser;
import org.apache.jetspeed.modules.actions.portlets.CustomizeAction;


public class IFrameController extends IFramePortlet
{
  protected String auth;
  protected String url;
  protected String sito;
  protected String protocol;
  protected String method;
  protected String avviso="Primo LOGIN: si prega di settare nome-Utente, password ed
eventuali variabili per la autenticazione.  Per inserire tali valori occorre attivare
la visualizzazione CUSTOMIZE. IMPORTANTE inserire le variabili nel formato:
key1=value1||key2=value2...";

  protected String loginUserParamName="user";
  protected String passwordParamName="password";
  protected String loginUser;
  protected String password;
  protected String userParams;
  protected Properties extendedParams = new Properties();



  /**
     * Riedita il file getContent
     * @param runData, value The runData and his value
   **/

  public ConcreteElement getContent (RunData runData)
```

```
  {

     JetspeedRunData jrun = (JetspeedRunData) runData;
     JetspeedUser user= jrun.getJetspeedUser();

     String name=user.getUserName();

     if("anon".equalsIgnoreCase(name))
     {
       //utente non loggato viene portato sull'url pubblico
       //nel quale potrà fare l'autenticazione normale
       url=this.getPortletConfig().getInitParameter("urlPublic");
       sito=url;
     }
     else {
       //utente loggato viene portato sull'url privato per l'autenticazione
       //e si controlla che l'utente abbia già nome utente e password settati

       url=this.getPortletConfig().getInitParameter("urlPrivate");

       if(this.getPortletConfig().getInitParameter("loginUserParamName") != null)

loginUserParamName=this.getPortletConfig().getInitParameter("loginUserParamName");
       if(this.getPortletConfig().getInitParameter("passwordParamName") != null)

passwordParamName=this.getPortletConfig().getInitParameter("passwordParamName");


       //Se l'utente possiede già nome utente e password per accedere al servizio
allora prosegue,
       //altrimenti viene invocata una jsp che chiede di inserire tali dati e li
memorizza
       //nei parametri di init.
       loginUser= this.getPortletConfig().getInitParameter("nomeUtente");
       password= this.getPortletConfig().getInitParameter("passUtente");
       auth= this.getPortletConfig().getInitParameter("Auth");
       method= this.getPortletConfig().getInitParameter("Method");


       if ("No".equalsIgnoreCase(auth)) {
         //Abbiamo un accesso senza richiesta di autenticazione
         //System.out.println("Auth uguale a No");
       }else {
         //System.out.println("Auth diverso a No");
         if ((loginUser=="" ) || (password=="") || ("".equalsIgnoreCase(auth))) {
           //Significa che c'è una autenticazione da svolgere e manca username o
password
           //Oppure che bisogna ancora settare i parametri allora viene visualizzato
           //un messaggio Standard che spiega come settare i parametri
           // url, nome utente, password, e variabili estese
           return new StringElement(avviso);
         }
       }


       //Costruisce il link in base al tipo di metodo della chiamata
       if("post".equalsIgnoreCase(method)) {
         //Metodo POST
         sito=getLinkUrlPost(jrun).toString();
       } else {
         //Metodo GET
         sito=getLinkUrlGet(jrun).toString();
       }

     }


     //Controllo la visualizzazione del Portlet e ne setto i parametri dell'Iframe
     switch (jrun.getMode())
     {
```

```
        case JetspeedRunData.NORMAL:

          //Visualizzazione normale del portlet
          this.setHeight(this.getPortletConfig().getInitParameter("normal_height"));
          this.setWidth(this.getPortletConfig().getInitParameter("normal_width"));
          break;

        case JetspeedRunData.MAXIMIZE:

          //Visualizzazione allargate del portlet

this.setHeight(this.getPortletConfig().getInitParameter("maximized_height"));
          this.setWidth(this.getPortletConfig().getInitParameter("maximized_width"));
          break;

       default:
         return new StringElement("ERRORE");
     }

     //utilizzo un metodo di IFramePortlet per settare site come url
     //da inserire nell'IFrame Portlet
     this.setSource(sito);
     return super.getContent(runData);
}


/**
    * Accede con metodo get al servizio richiesto
    * In base alla tipologia di autenticazione(Basic/Form/NO AUth)
    * @param jrun, value jrun and his value
    *   */

public StringBuffer getLinkUrlGet(RunData jrun)
{

  StringBuffer s = new StringBuffer();

  //Gestione dei parametri extra**********************************
  //li separo e li aggiungo all ExtendedParams
  StringTokenizer st;
  String parametro;
  String key;
  String value;

  userParams=this.getPortletConfig().getInitParameter("extendedParams");
  if(userParams !=null) {
    st=new StringTokenizer(userParams, "||");
    while (st.hasMoreElements())  {
      parametro= st.nextToken();
      int index=parametro.indexOf('=');
      if (index>0) {
        key=parametro.substring(0, index);
        value=parametro.substring(index+1);
        this.addExtendedParam(key,value);
      }
    }
  }
  //Fine gestione dei parametri extra*****************************


  //A seconda della tipologia di autenticazione e del protocollo
  //richiesto, viene creata la richiesta
  if(this.getPortletConfig().getInitParameter("Auth")!=null)  {
    auth=getPortletConfig().getInitParameter("Auth");
  }

  if(this.getPortletConfig().getInitParameter("Protocol")!=null)  {
    protocol=getPortletConfig().getInitParameter("Protocol");
  }
```

```
  if (auth.equalsIgnoreCase("BASIC"))
  {
    //BASIC Authentication
    //Metodo GET costruisco l'URL
    s.append(protocol + "://"+loginUser + ":" + password + "@" + url);

    Enumeration enum = extendedParams.keys();
    boolean firstParam = true;
    while (enum.hasMoreElements()) {
      String ke = (String) enum.nextElement();
      if (firstParam) {
        s.append("?" + ke + "=" + extendedParams.getProperty(ke));
        firstParam = false;
      } else {
        s.append("&" + ke + "=" + extendedParams.getProperty(ke));
      }
    }
  } else
  {
    if (auth.equalsIgnoreCase("FORM"))
    {
      //FORM authentication
      //Metodo GET costruisco l'URL
      s.append(protocol + "://" + url + "?" + loginUserParamName + "=" + loginUser +
"&" + passwordParamName + "=" + password);

      Enumeration enum = extendedParams.keys();
      while (enum.hasMoreElements()) {
        String ke = (String) enum.nextElement();
        s.append("&" + ke + "=" + extendedParams.getProperty(ke));
      }


    } else {
      //NO Authentication
        s.append(protocol + "://" + url);

        Enumeration enum = extendedParams.keys();
        boolean firstParam = true;
        while (enum.hasMoreElements()) {
          String ke = (String) enum.nextElement();
          if (firstParam) {
            s.append("?" + ke + "=" + extendedParams.getProperty(ke));
            firstParam = false;
          } else {
            s.append("&" + ke + "=" + extendedParams.getProperty(ke));
          }
        }
    }
  }
  return(s);
}



/**
   * Richiede la creazione di una Form Dinamica
   * @param jrun, value jrun and his value
   *   */
public String getLinkUrlPost(RunData jrun)
{

  //Inizializzo una variabile di comodo
  StringBuffer s = new StringBuffer();

  //Gestione dei parametri extra*********************************
  //li separo e li aggiungo all ExtendedParams
  StringTokenizer st;
```

```
  String parametro;
  String key;
  String value;

  userParams=this.getPortletConfig().getInitParameter("extendedParams");
  if(userParams !=null) {
    st=new StringTokenizer(userParams, "||");
    while (st.hasMoreElements())  {
      parametro= st.nextToken();
      int index=parametro.indexOf('=');
      if (index>0) {
        key=parametro.substring(0, index);
        value=parametro.substring(index+1);
        this.addExtendedParam(key,value);
      }
    }
  }
  //Fine gestione dei parametri extra*****************************

  //A seconda della tipologia di autenticazione e del protocollo
 //richiesto, viene creata la richiesta
 if(this.getPortletConfig().getInitParameter("Auth")!=null)  {
   auth=getPortletConfig().getInitParameter("Auth");
 }

 if(this.getPortletConfig().getInitParameter("Protocol")!=null)  {
   protocol=getPortletConfig().getInitParameter("Protocol");
  }


  if (auth.equalsIgnoreCase("BASIC")||auth.equalsIgnoreCase("FORM")) {
    //METODO POST , AUTH= BASIC oppure FORM
    //Invio la richiesta di creazione di un Form Dinamico alla pagina JSP
    //che risiede sul server

    //Nel caso post.jsp risieda su Localhost oppure nel server Universitario apollo
    s.append("http://localhost:8080/jportal/pro/post.jsp?sito="+protocol+"://"+url);

//s.append("http://localhost:8080/lucaportal/pro/post.jsp?sito="+protocol+"://"+url);

//s.append("http://apollo.ing.unimo.it/jportal/pro/post.jsp?sito="+protocol+"://"+url
);


    s.append("&"+ loginUserParamName +"="+loginUser);
    s.append("&"+ passwordParamName +"="+password);
    Enumeration enum = extendedParams.keys();
    while (enum.hasMoreElements()) {
      String ke = (String) enum.nextElement();
      s.append("&"+ ke +"="+extendedParams.getProperty(ke));
    }
  } else  {
    //NO METHOD
    //Creo Il Form senza Nome Utente e Password ma solo con i parametri estesi
    //Nel caso post.jsp risieda su Localhost oppure nel server Universitario apollo
    s.append("http://localhost:8080/jportal/pro/post.jsp?sito="+protocol+"://"+url);

//s.append("http://localhost:8080/lucaportal/pro/post.jsp?sito="+protocol+"://"+url);

//s.append("http://apollo.ing.unimo.it/jportal/pro/post.jsp?sito="+protocol+"://"+url
);


    Enumeration enum = extendedParams.keys();
    while (enum.hasMoreElements()) {
      String ke = (String) enum.nextElement();
      s.append("&"+ ke +"="+extendedParams.getProperty(ke));
    }
  }
  return(s.toString());
```

```
}
```

```
  /**
   * Sets the name and the value
   * @param name, value The name and his value
   **/
  public void addExtendedParam(String name, String value) {
    extendedParams.setProperty(name, value);
  }
```

IFrameController.java

Created with JBuilder

8.3.2 Codice pagina JSP: “post.jsp”

```
<%@ page language="java"
         import="java.util.Enumeration"
         session="false"
%>


<%
      Enumeration e = request.getHeaderNames();

      while (e.hasMoreElements()) {
        String name = (String)e.nextElement();
        String value = request.getHeader(name);
      }

      e = request.getParameterNames();

      while (e.hasMoreElements()) {
        String name = (String)e.nextElement();
        String value = request.getParameter(name);
        if("sito".equalsIgnoreCase(name))
        {
              out.println(" <form method='POST' action='"+ value
+"' id='form1' name='form1'>");
         }
       }

      e = request.getParameterNames();
      while (e.hasMoreElements()) {
        String name = (String)e.nextElement();
        String value = request.getParameter(name);
        if("sito".equalsIgnoreCase(name))
        {

        }
        else
        out.println(" <input type='hidden' name='"+ name +"'
value='" + value + "'> ");
      }
      out.println("<input type='submit' value='Accedi'>");
      out.println("</form>");

%>
```

8.4 Servizi Universitari Integrati

Con l'uso dell'IframeController si sono riusciti ad integrare in un'unica pagina chiamata *services* del portale i seguenti servizi universitari dedicati ai docenti:

- Amministrazione Pagina Personale
- E-mail personale
- Inserimento News di facoltà
- Inserimento News Studenti
- Accesso rete Intranet
- Accesso a servizi senza autenticazione come lista newsgroup

Chiaramente per ogni servizio si sono create istanze dello stesso portlet caratterizzate da parametri nel registry adatti alla tipologia di autenticazione al server. Di seguito riporto una lista dei parametri necessari alla configurazione dell'IFramePortlet nel file di registry:

Parametro	Valore assegnato nel registry
Normal_height	Dimensione in percentuale dell'altezza della finestra del portlet in visualizzazione normale.
Normal_width	Dimensione in percentuale della larghezza della finestra del portlet in visualizzazione normale.
Maximized_height	Dimensione in percentuale dell'altezza della finestra del portlet in visualizzazione massimizzata.
Maximized_weidhth	Dimensione in percentuale della larghezza della finestra del portlet in visualizzazione massimizzata.
loginUserParamName	Nome della variabile che trasporta il valore dello Username
PasswordParamName	Nome della variabile che trasporta il valore della Password
NomeUtente	XXX
PassUtente	XXX
Auth	Tipologia di autenticazione (BASIC,FORM/No)
urlPrivate	URL che analizza l'autenticazione
urlPublic	URL sul quale ridirigere gli utenti non loggiati al portale
extendedParams	Parametri estesi vanno inseriti come: Name1=Value1||Name2=Value2……
Protocol	http/https
Method	Metodo della request inviata dal client

Figura 87: -Tabella dei parametri da configurare in IFramePortlet-

8.4.1 Amministrazione Pagina Personale

Per accedere al servizio di Amministrazione pagina Personale sono stati settati i seguenti parametri:

Parametro	Valore assegnato nel registry
Normal_height	100%
Normal_width	100%
Maximized_height	500%
Maximized_weidhth	100%
loginUserParamName	Login
PasswordParamName	password
NomeUtente	XXX
PassUtente	XXX
Auth	FORM
urlPrivate	ares.ing.unimo.it/formFacolta/validazione_luca.asp
urlPublic	www.ing.unimo.it
extendedParams	submit1=Invia
Protocol	http
Method	get

Figura 88: -Parametri per l'accesso al servizio Amministarzione pagina docente -

La relativa configurazione del portlet nel registry file è la seguente:

```
<portlet-entry name="IFrameController2" hidden="false"
      type="instance" application="false">
      <meta-info>
         <title>IFrameAmministrazione</title>
         <description>Portlet per la amministrazione delle pagine docenti</description>
      </meta-info>
      <classname>it.gruppopro.jportal.portlets.iframe.IFrameController</classname>
      <parameter name="normal_height" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="normal_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_height" value="500%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="loginUserParamName" value="login" hidden="true"
```

```
      cachedOnName="true" cachedOnValue="false"/>
    <parameter name="passwordParamName" value="password"
      hidden="true" cachedOnName="true" cachedOnValue="false"/>
    <parameter name="nomeUtente" value="" hidden="false"
      cachedOnName="true" cachedOnValue="false"/>
    <parameter name="passUtente" value="" hidden="false"
      cachedOnName="true" cachedOnValue="false"/>
    <parameter name="Auth" value="FORM" hidden="true"
      cachedOnName="true" cachedOnValue="false"/>
    <parameter name="urlPrivate"
      value="ares.ing.unimo.it/formFacolta/validazione_luca.asp"
      hidden="false" cachedOnName="true" cachedOnValue="false"/>
    <parameter name="urlPublic" value="www.ing.unimo.it"
      hidden="true" cachedOnName="true" cachedOnValue="false"/>
    <parameter name="extendedParams" value="submit1=Invia"
      hidden="true" cachedOnName="true" cachedOnValue="false"/>
    <parameter name="Protocol" value="HTTP" hidden="true"
      cachedOnName="true" cachedOnValue="false"/>
    <parameter name="Method" value="get" hidden="true"
      cachedOnName="true" cachedOnValue="false"/>
    <media-type ref="html"/>
    <url cachedOnURL="false"/>
  </portlet-entry>
```

8.4.2 Servizio E-mail

Il servizio di e-mail universitaria, utilizza HTTPS: un protocollo molto simile all'http che consente la trasmissione sicura (criptata) di documenti o pagine HTML riservate. Le pagine riservate sono solitamente memorizzate in chiaro sul server, vengono criptate solo durante la trasmissione al client, per impedirne la lettura da parte di estranee (man-in-the-middle) e decriptate nel computer del client per permettergli la visualizzazione. Al fine di poter decriptare le pagine, occorre che il client possegga una chiave pubblica chiamata (PKC) Public Key Certificate rilasciata da un ente certificatore. Jetspeed supporta questa tecnologia, anche se presenta alcuni problemi nella gestione del certificato. Quando si visualizza una pagina criptata, Jetspeed innesta un ciclo che per più volte richiede al browser di visualizzare un "Avviso di protezione" che ci avverte della tipologia di certificato che si stà per acettare. A parte questo piccolo inconveniente, il servizio funziona efficientemente.

Parametro	Valore assegnato nel registry
Normal_height	100%
Normal_width	100%
Maximized_height	500%
Maximized_weidhth	100%

loginUserParamName	Imapusser
PasswordParamName	pass
NomeUtente	XXX
PassUtente	XXX
Auth	FORM
urlPrivate	mail.unimo.it/horde/imp/redirect.php
urlPublic	www.ing.unimo.it
extendedParams	mailbox=INBOX||folders=||port=143||server=mail.unimo.it|| namespace=INBOX||new_lang=it_IT||actionID=105|| button=Entra||realm=unimo.it||maildomain=unimo.it|| redirect_url=||protocol=imap
Protocol	HTTPS
Method	POST

Figura 89: - Parametri per l'accesso allla mail universitaria -

La configurazione nel file di registry relativa al portlet dedicato all'accesso al servizio postale universitario è la seguente:

```
<portlet-entry name="IFrameController6" hidden="false"
      type="instance" application="false">
      <meta-info>
         <title>IFramePosta</title>
         <description>Portlet per accedere alla posta Universitaria</description>
      </meta-info>
      <classname>it.gruppopro.jportal.portlets.iframe.IFrameController</classname>
      <parameter name="normal_height" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="normal_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_height" value="500%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="loginUserParamName" value="imapuser"
         hidden="true" cachedOnName="true" cachedOnValue="false"/>
      <parameter name="passwordParamName" value="pass" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="nomeUtente" value="" hidden="false"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="passUtente" value="" hidden="false"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="Auth" value="FORM" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="urlPrivate"
```

```
            value="mail.unimo.it/horde/imp/redirect.php" hidden="true"
            cachedOnName="true" cachedOnValue="false"/>
          <parameter name="urlPublic" value="www.ing.unimo.it"
            hidden="true" cachedOnName="true" cachedOnValue="false"/>
          <parameter name="extendedParams"

value="mailbox=INBOX||folders=||port=143||server=mail.unimo.it||namespace=INBOX||new_lang=it
_IT||actionID=105||button=Entra||realm=unimo.it||maildomain=unimo.it||redirect_url=||protocol=imap
"
            hidden="true" cachedOnName="true" cachedOnValue="false"/>
          <parameter name="Protocol" value="HTTPS" hidden="true"
            cachedOnName="true" cachedOnValue="false"/>
          <parameter name="Method" value="post" hidden="true"
            cachedOnName="true" cachedOnValue="false"/>
          <media-type ref="html"/>
          <url cachedOnURL="true"/>
        </portlet-entry>
```

8.4.3 Inserimento News

Il sito di facoltà, permette l'inserimento di news universitarie suddivise in "Notizie ed Eveneti" e "Notizie agli studenti", in particolare per ognuna di queste due tipologie di News il sito della facoltà prevede due tipologie di autenticazione. L'IframePortlet permette all'utente di autenticarsi su uno qualsiasi di questi servizi, settando opportunamente il valore di un parametro esteso di nome domain.

Parametro	Valore assegnato nel registry
Normal_height	100%
Normal_width	100%
Maximized_height	500%
Maximized_weidhth	100%
loginUserParamName	User
PasswordParamName	pass
NomeUtente	XXX
PassUtente	XXX
Auth	FORM
urlPrivate	notizie.unimo.it/campus/controller/AccessHandler
urlPublic	www.ing.unimo.it
extendedParams	Domain=INGMO[11]

[11] domain=INGMO (Inserisce news in Notizie ed Eventi)
domain=INGMOST(Inserisce news in Notizie agli studenti)

Protocol	http
Method	get

Figura 90: - Parametri per l'accesso all'Inserimento News -

La configurazione nel file di registry relativa al portlet dedicato inserimento di News in ambito universitario è la seguente:

```
<portlet-entry name="IFrameController5" hidden="false"
      type="instance" application="false">
      <meta-info>
         <title>IFrameNotizie</title>
         <description>Portlet per inserimento notizie in ambito universitario</description>
      </meta-info>
      <classname>it.gruppopro.jportal.portlets.iframe.IFrameController</classname>
      <parameter name="normal_height" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="normal_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_height" value="500%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="loginUserParamName" value="user" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="passwordParamName" value="pass" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="nomeUtente" value="" hidden="false"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="passUtente" value="" hidden="false"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="Auth" value="FORM" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="urlPrivate"
         value="notizie.unimo.it/campus/controller/AccessHandler"
         hidden="false" cachedOnName="true" cachedOnValue="false"/>
      <parameter name="urlPublic" value="www.ing.unimo.it"
         hidden="true" cachedOnName="true" cachedOnValue="false"/>
      <parameter name="extendedParams" value="domain=INGMO"
         hidden="false" cachedOnName="true" cachedOnValue="false"/>
      <parameter name="Protocol" value="HTTP" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="Method" value="get" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <media-type ref="html"/>
      <url cachedOnURL="true"/>
   </portlet-entry>
```

8.4.4 Accesso alla Rete Intranet

La rete intranet rappresenta un accesso a vari servizi disponibili in ambito universitario. In particolare l'autenticazione a questo servizio segue i parametri sotto riportati:

Parametro	Valore assegnato nel registry
Normal_height	100%
Normal_width	100%
Maximized_height	500%
Maximized_weidhth	100%
loginUserParamName	FldUser
PasswordParamName	FldPassword
NomeUtente	XXX
PassUtente	XXX
Auth	FORM
urlPrivate	www.ing.unimo.it/intranet/login.asp
urlPublic	www.ing.unimo.it
extendedParams	Action=logIn
Protocol	http
Method	post

Figura 91: - Parametri di accesso alla rete Intranet-

Configurazione del portlet nel registry.

```
<portlet-entry name="IFrameController4" hidden="false"
      type="instance" application="false">
      <meta-info>
         <title>IFrameIntranet</title>
         <description>Permette di accedere alla intranet Universitaria</description>
      </meta-info>
      <classname>it.gruppopro.jportal.portlets.iframe.IFrameController</classname>
      <parameter name="normal_height" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="normal_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_height" value="500%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="loginUserParamName" value="FldUserId"
```

```
    hidden="true" cachedOnName="true" cachedOnValue="false"/>
  <parameter name="passwordParamName" value="FldPassword"
    hidden="true" cachedOnName="true" cachedOnValue="false"/>
  <parameter name="nomeUtente" value="" hidden="false"
    cachedOnName="true" cachedOnValue="false"/>
  <parameter name="passUtente" value="" hidden="false"
    cachedOnName="true" cachedOnValue="false"/>
  <parameter name="Auth" value="FORM" hidden="true"
    cachedOnName="true" cachedOnValue="false"/>
  <parameter name="urlPrivate"
    value="www.ing.unimo.it/intranet/login.asp" hidden="false"
    cachedOnName="true" cachedOnValue="false"/>
  <parameter name="urlPublic" value="www.ing.unimo.it"
    hidden="true" cachedOnName="true" cachedOnValue="false"/>
  <parameter name="extendedParams" value="Action=LogIn"
    hidden="true" cachedOnName="true" cachedOnValue="false"/>
  <parameter name="Protocol" value="HTTP" hidden="true"
    cachedOnName="true" cachedOnValue="false"/>
  <parameter name="Method" value="post" hidden="true"
    cachedOnName="true" cachedOnValue="false"/>
  <media-type ref="html"/>
  <url cachedOnURL="true"/>
</portlet-entry>
```

8.4.5 Accesso a servizi senza autenticazione: lista dei Newsgroup

Questo servizio non necessita una autenticazione vera e propria, è possibile accedere a questo servizio conoscendo semplicemente il suo URL. Considerata la riservatezza dei dati viene comunque utilizzato un protocollo HTTPS. I parametri settati per questo accesso sono sotto riportati:

Parametro	Valore assegnato nel registry
Normal_height	100%
Normal_width	100%
Maximized_height	500%
Maximized_weidhth	100%
loginUserParamName	
PasswordParamName	
NomeUtente	
PassUtente	
Auth	No
urlPrivate	sparc20.ing.unimo.it/mailman/listinfo
urlPublic	www.ing.unimo.it

extendedParams	
Protocol	https
Method	post

Figura 92: - Parametri di accesso per i NewsGroup -

Configurazione del portlet nel file di registry:

```
<portlet-entry name="IFrameController3" hidden="false"
      type="instance" application="false">
      <meta-info>
         <title>IFrameHTTPS</title>
         <description>Portlet che elenca le NewsGroup di Ingegneria</description>
      </meta-info>
      <classname>it.gruppopro.jportal.portlets.iframe.IFrameController</classname>
      <parameter name="normal_height" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="normal_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_height" value="500%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="maximized_width" value="100%" hidden="true"
         cachedOnName="true" cachedOnValue="true"/>
      <parameter name="loginUserParamName" value="login" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="passwordParamName" value="password"
         hidden="true" cachedOnName="true" cachedOnValue="false"/>
      <parameter name="nomeUtente" value="" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="passUtente" value="" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="Auth" value="No" hidden="false"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="urlPrivate"
         value="sparc20.ing.unimo.it/mailman/listinfo" hidden="false"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="urlPublic" value="www.ing.unimo.it"
         hidden="true" cachedOnName="true" cachedOnValue="false"/>
      <parameter name="extendedParams" value="" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="Protocol" value="HTTPS" hidden="false"
         cachedOnName="true" cachedOnValue="false"/>
      <parameter name="Method" value="post" hidden="true"
         cachedOnName="true" cachedOnValue="false"/>
      <media-type ref="html"/>
      <url cachedOnURL="true"/>
   </portlet-entry>
```

8.5 Interfaccia utente

Quando l'utente si connette all'Enterprise Information Portal sviluppato ed installato nel server universitario sotto il direttorio (jportal), inizialmente non viene riconosciuto come utente registrato e quindi di default, viene caricata la pagina definita nel parametro URL Public (In questo caso il sito della facoltà di Ingegneria).

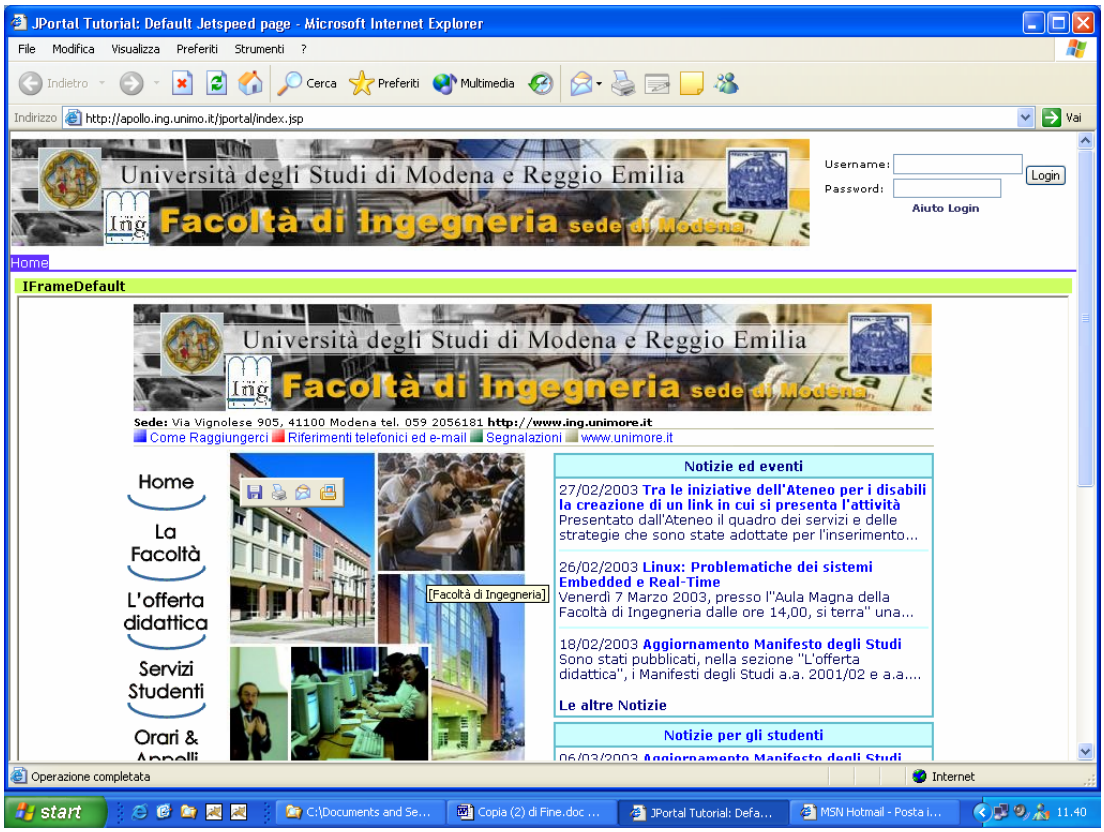


Figura 93: - JPORTAL: Interfaccia utente Pagina di Default -

Una volta che l'utente ha eseguito l'autenticazione ed è stato riconosciuto dall'EIP attraverso la fase di personalization, l'enterprise Information Portal carica sulla sua pagina personalizzata i servizi e le informazioni ad esso dedicate. Nella Figura sotto viene riportata la pagina di accesso di un eventuale docente registrato al portale installato in ambito universitario: si notano i vari servizi di Posta, Amministrazione ecc.. che vengono integrati nella pagina personalizzata dal docente grazie all'uso del portlet sviluppato durante la tesi: IfameController.

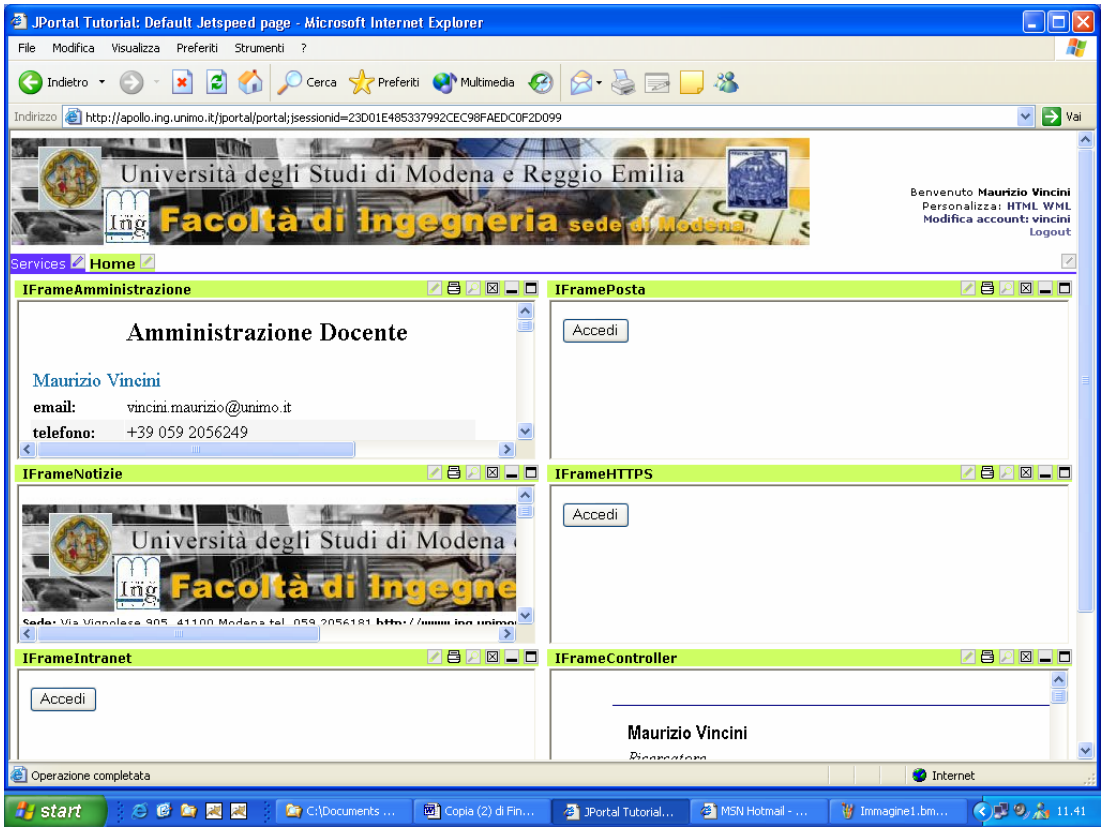


Figura 94: - JPORTAL: Interfaccia Docente -

Conclusioni e sviluppi futuri

Durante lo sviluppo della tesi si è potuto constatare che scegliere di sviluppare un EIP con l'ausilio di prodotti commerciali o con l'uso di prodotti OpenSource pone lo sviluppatore di fronte a due scenari completamente diversi. In particolare se si necessita di un EIP corredato da applicativi e servizi standard è sicuramente consigliabile optare per un prodotto commerciale, ma se si necessita di modificare l'EIP per integrarlo con sistemi aziendali proprietari e non standardizzati ecco che la scelta si complica.

Scegliere EIP di tipo commerciale per realizzare nuovi portlet capaci di integrare servizi e sistemi informatici proprietari in Enterprise Information Portal può significare una notevole diminuzione sui tempi di realizzazione dei nuovi applicativi. La possibilità infatti di riadattare componenti informatici già creati e la possibilità di utilizzare avanzati tool di sviluppo e tool amministrativi grafici per creare e gestire i servizi del portale permettono di agevolare le fasi relative allo sviluppo. Scegliere un EIP commerciale implica comunque accettare oltre ai vincoli dettati da eventuali scelte progettuali, già assunte dagli sviluppatori del software acquistato, l'obbligo di sostenere costi fissi dovuti alle licenze d'uso. Questi fattori condizionano negativamente le potenzialità e la competitività dei nuovi servizi sviluppati. Considerato inoltre che anche l'utilizzo di prodotti commerciali non elimina una pesante fase di analisi per integrare eventuali sistemi informativi preesistenti, è immediato pensare che per interfacciare software proprietari nel contesto di un Enterprise Information Portal possa risultare conveniente implementare servizi attraverso tecnologie OpenSource.

Una scelta Open Source può rappresentare infatti, per quelle aziende come come Gruppo Pro, che dispongono di avanzati laboratori di sviluppo, una notevole riduzione dei costi, dovuti all'assenza di licenze. Sviluppare prodotti a partire da un OpenSource quale Jetspeed implica però uno sforzo maggiore per la realizzazione e successivamente per la manutenzione di nuovi applicativi. Jetspeed rappresenta un prodotto che ha tutte le potenzialità per realizzare avanzati servizi nell'ambito degli Enterprise Information Portal ma non è corredato dai tool e dai componenti informatici riutilizzabili che caratterizzano i prodotti commerciali analizzati.

A fronte di queste problematiche si è scelto di sviluppare come test, un portlet capace di integrare nel contesto di un EIP, dei servizi Web preesistenti situati all'interno del sito della facoltà di Ingegneria. Lo sviluppo del portlet ha necessitato una fase di analisi delle tecnologie, dell'architettura J2EE e delle API utilizzate da Jetspeed, quindi la realizzazione di classi Java, file di tipo Xml e Psml e la realizzazione di pagine dinamiche attraverso l'uso della tecnologia JSP. Il portlet realizzato permette oggi l'integrazione di servizi in ambito Web con o senza autenticazioni realizzati con tecnologia

- HTML
- ASP
- PHP

- JSP

E supporta i protocolli di comunicazione

- HTTP
- HTTPS

In ambito aziendale la mia tesi potrà essere utilizzata per integrare, nel contesto di un Enterprise Information Portal sviluppato con Jetspeed, servizi accessibili tramite autenticazione e non. La capacità del portlet sviluppato, di interfacciarsi agli standard più comuni di autenticazione, permette di integrare, nel contesto di un EIP, servizi preesistenti, magari sviluppati in soluzioni informatiche proprietarie quali gli ERP. Le potenzialità offerte da Jetspeed, di configurare i portlet per dispositivi operanti con tecnologia WAP, permetterebbero di aprire al mondo wireless tutte quelle risorse oggi raggiungibili attraverso Internet.

Nel contesto universitario L'EIP installato può essere migliorato al fine di creare un accesso ai servizi universitari oltre che al personale docente anche agli studenti e a tutti quegli utenti che rientrano in un qualche modo ad agire nei processi universitari. In particolar modo sarebbe possibile configurare gruppi di utenti (Docente/Studente/Aziende convenzionate ecc...) e fornirgli i relativi diritti di accesso ai servizi a loro dedicati. Tale configurazione permetterebbe di diversificare la visualizzazione delle informazioni e delle applicazioni universitarie in base alla tipologia di utente connesso informatizzando e personalizzandole varie procedure di gestione.

Attualmente l'Enterprise Information Portal risulta quindi essere un punto di accesso personalizzato e dinamico per la visualizzazione delle informazioni e fruizione di servizi raggiungibili attraverso la rete, ma grazie agli attuali studi di estrazione dati da documenti HTML, l'EIP potrebbe evolversi in un punto di accesso personalizzato alle risorse finalizzato alla estrapolazione dati. Tali sviluppi futuri permetterebbero quindi di accedere, visualizzare ed utilizzare le informazioni in rete in modo integrato ai programmi ed applicativi comunemente presenti nei nostri computer.

Bibliografia

[data Warehouse]: Paolo Tiberio, "Tecnologia dei Sistemi Informativi" dispense del corso di Sistemi Informativi .

[InfoWorld] : L'articolo riassuntivo riguardo il report di Merril Lynch è disponibile al sito:http://www.infoworld.com/cgi-bin/displayStory.pl?/features/990125eip.html

[JSRs] : Java Specification Request Comunità http://www.jcp.org/aboutJava/communityprocess/ accepted.html

[J2EE] : "Java 2 Enterprise Edition" documentazione disponibile al sito http://java.sun.com/j2ee/docs.html. J2EE SDK scaricabile al URL http://java.sun.com/j2ee/download.html . Un ottimo tutorial per sviluppatori di applicazioni J2EE è disponibile al URL http://java.sun.com/blueprints/index.html .

[SUN] : Sito ufficiale di Java della Sun Microsystem : http://java.sun.com.

[SPECEJB] : Specifiche degli Enterprise Java Beans scaricabili al URL http://java.sun.com/products/ejb/docs.html

[IBM] : Sito ufficiale: http://www.ibm.com

[BEA] : Sito ufficiale:http://www.bea.com

[BEA] : Sito ufficiale italiano: http://www.it.bea.com

[JAKARTA] : Sito ufficiale: http://jakarta.apache.org

[JSR 168] : Java Specification Request: http://www.jcp.org/en/jsr/detail?id=168

[Tutorial] : Tutorial di Jestspeed dedicato ai portlet consultabile al sito: : http://www.bluesunrise.com/jetspeed-docs/JetspeedTutorial.htm

[Velocity] : Sito ufficiale http://jakarta.apache.org/velocity

[MVC] "Model View Controller - J2EE Design patterns" documentazione al URL http://java.sun.com/blueprints/patterns/j2ee_patterns/model_view_controller/

[Ant] : Ant è un tool di build rilasciato in modalità OpenSource dal gruppo di ricerca Apache scaricabile al sito http://jakarta.apache.org/ant

[Torque] : jakarta Torque, è un persistence Layer, per maggiori informazioni vedere il sito ufficiale: http://db.apache.org/torque/

Clive Finkelstein, Peter Aiken,: "Building Corporate Portals with XML" McGraw-Hill, Sept 1999.

Clive Finkelstein, Alan Perkins: "Enterprise Portal Engineering" Visible System Corporation Sito: http://visible.com.au

"Humingbird™ EIP" White Paper : http://www.humingbird.com

Gary J. Asmus "Enterprise Information Portal",Innovative Emergency Managment ® Inc. White Paper : http://ieminc.com

Andrea Cervellati: "Portale Web di Facoltà: progetto e implementazione di home page dei docenti mediante architettura J2EE" tesi di laurea in Ingegneria Informatica Università di Modena.