Atenarium - Departmentarium

Progetto di Sistemi Middleware

Alberto Bacchelli - Luca Bigliardi

13 dicembre 2007

Indice

Capitolo 1

Introduzione

Atenarium e Departmentarium sono due componenti software che implementano un sistema Web distribuito per la gestione automatizzata delle pubblicazioni scientifiche di un ateneo. Departmentarium fornisce un'interfaccia web, accessibile da browser, che consente agli utenti afferenti al dipartimento di inserire e amministrare i report tecnici. Atenarium, contattando i singoli dipartimenti, fornisce anch'esso un'interfaccia web accessibile da browser che offre un catalogo di tutte le pubblicazioni e la possibilità di effettuare ricerche su di esso.

Nel **secondo capitolo** si tratta il processo produttivo affrontato per la creazione del sistema: vengono illustrate le funzionalità richieste, le scelte implementative, le soluzioni software adottate e le problematiche riscontrate durante lo sviluppo.
Nel **terzo capitolo** si descrive il sistema sviluppato: vengono presentati i dettagli architetturali e un esempio di installazione e configurazione.

Capitolo 2

Progetto

Il seguente capitolo presenta vari aspetti del processo produttivo che ha portato alla creazione del software descritto nel capitolo seguente. La prima sezione richiama le funzionalità richieste. La seconda sezione illustra le scelte implementative più significative che esulano dai requisiti. La terza sezione descrive le componenti software utilizzate e infine la quarta sezione presenta i maggiori problemi affrontati durante lo sviluppo.

2.1 Requisiti

Sviluppare un sistema Web che consenta ad un ateneo, suddiviso in dipartimenti, di automatizzare la gestione distribuita delle pubblicazioni scientifiche. Il sistema finale dovrà fornire due tipi di servizi:

- un servizio a livello di dipartimento che consenta al proprio personale scientifico di pubblicare rapporti tecnici;
- un servizio d'ateneo che combini i singoli servizi di dipartimento e che offra funzioni di ricerca e catalogazione dei rapporti tecnici.

2.1.1 Servizio di Dipartimento

Il servizio di dipartimento dovrà essere un servizio Web, accessibile da un comune browser, che consente di pubblicare un rapporto tecnico. Con riferimento alle specifiche di Ingegneria del Software[1], per pubblicare un rapporto tecnico è necessario:

- essere membro scientifico del dipartimento (professore, ricercatore, assegnista, dottorando, ospite);
- essere l'autore del rapporto tecnico;
- avere un numero progressivo che identifichi il rapporto tecnico (es. UBCLS-20006-01). Tale numero progressivo dovrà essere fornito dal servizio stesso automaticamente;
- fare l'upload del file pdf finale generato a partire da un codice sorgente (LaTeX o Word) fornito dal servizio stesso.

Il personale scientifico accede al servizio tramite login (id uguale al nome del professore/ricercatore/assegnista ecc... e password). Il servizio inoltre dovrà consentire all'autore del rapporto tecnico di sostituire un suo documento già pubblicato con una nuova versione e di pubblicare un documento che sia una raccolta di altri lavori (ad esempio, un professore è editor dei proceedings di una conferenza).

2.1.2 Servizio di Ateneo

Il servizio d'ateneo, accessibile da tutti, dovrà generare un catalogo di tutte le pubblicazioni dei vari dipartimenti e offrire una funzione di

[1]Disponibili alla pagina http://courses.web.cs.unibo.it/IngegneriaDelSoftware/-ProgettoDelCorso

ricerca di tutti i rapporti su un certo argomento. A tal proposito, il servizio di ateneo si dovrà interfacciare con i singoli servizi dipartimentali.

2.2 Scelte Implementative

Durante il design dell'applicazione e lo sviluppo del codice sono emerse problematiche non strettamente vincolate dai requisiti descritti nella sezione precedente. Nella soluzione di questi dubbi si è cercato di ottenere il miglior compromesso tra usabilità dell'applicazione, performance ed eleganza architetturale. Tra i vari punti chiariti vengono qui riportati i più significativi:

- **Identificativo documenti:** la configurazione del prefisso dell'identificativo univoco di un documento avviene ad opera dell'amministratore del dipartimento. Durante il ciclo di vita dell'applicazione è possibile cambiare tale prefisso ma, per evitare conflitti, la parte numerica del codice non viene azzerata.
- **Gestione utenti e report:** la gestione degli utenti prevede sia l'inserimento che l'eliminazione. I report associati ad un utente non più presente nel sistema vengono mantenuti nel database.
- **Ricerca:** La ricerca dei report può essere fatta a livello di ateneo sui campi *titolo*, *autore* e *keyword*. Si è cercato di ridurre il carico di lavoro del Web Container facendo uso di AJAX.
- **Gestione file:** i file pdf inseriti nel sistema vengono memorizzati direttamente nel database per garantire maggiore portabilità e distribuibilità. Lato ateneo si può scaricare un documento tramite un collegamento che punta al dipartimento. I report per cui viene richiesto il download non sono memorizzati in file temporanei, ma vengono processati direttamente da una servlet.

- **Update report:** per un report preesistente i parametri che si possono aggiornare sono: *file*, *titolo* e *keyword*.
- **Catalogazione:** si è cercato di minimizzare il numero di interazioni WebService effettuando una sola richiesta dei report per sessione.

2.3 Soluzioni Software

In questa sezione vengono elencate e brevemente illustrate le soluzioni software utilizzate sia come ambiente di sviluppo sia come componenti base su cui è stata implementata la nostra applicazione.

2.3.1 JBoss 4.0.5GA

JBoss è un'implementazione opensource di un application server J2EE strutturato come un insieme i servizi middleware per la comunicazione, la persistenza le transazioni e la sicurezza. Questi servizi interoperano come una sorta di microkernel denominato *java management extension*. JMX fornisce agli sviluppatori un bus software comune che consente di integrare vari componenti quali moduli, container e plugin.

Il software scritto utilizza direttamente i servizi di JBoss che offrono un servizio di naming e di directory (JNDI), un container di enterprise java beans (EJB) ed un servlet e jsp container incluso nella distribuzione di JBoss (Tomcat).

2.3.2 Axis 1.4

Axis è un framework opensource, basato su XML, per la creazione di Web Service. La componente principale è l'implementazione di un SOAP server, corredato da varie utility che facilitano la creazione, il deploy, il testing e il debugging.

Lo sviluppo del sottosistema di dialogo remoto è stato notevolmente semplificato grazie alle parti di Axis che permettono la generazione automatica di un client java a partire da un file WSDL e la serializzazione di oggetti di tipo java bean tramite opportune dichiarazioni all'interno del file WSDD.

2.3.3 iText 2.0.6

iText è la più diffusa libreria java che permette la generazione e la manipolazione dei file in formato pdf.

Nel progetto iText è stata utilizzata durante la lettura e la manipolazione dei metadati dei pdf.

2.3.4 MyFaces 1.1.5

JavaServer Faces definisce un framework che semplifica lo sviluppo delle interfacce utente delle applicazioni Java EE. L'utilizzo di JSF è da preferirsi alle semplici pagine JSP in quanto porta il programmatore a seguire il pattern *model-view-controller*. Molti framework web adottano un'architettura MVC basata sulle richieste; JSF, a differenza di questi, utilizza un approccio basato sui componenti.

Vi sono più implementazioni di JSF, è stata scelta MyFaces poichè direttamente disponibile all'interno di JBoss.

2.3.5 Facelets 1.1.3

Facelets è un sistema di creazione dei template della parte "view" di JSF tramite la definizione di pagine *xhtml*. L'utilizzo di facelets durante la creazione delle interfacce JSF è una scelta naturale in quanto permette la suddivisione in componenti delle varie parti che occorrono in una pagina web.

2.3.6 Tomahawk 1.1.6

Tomahawk è una libreria di componenti che amplificano le potenzialità di JSF fornendo svariate funzionalità aggiuntive.

Nel progetto Tomahawk è stato utilizzato per gestire l'upload dei file.

2.3.7 RichFaces 3.2.1sp1

RichFaces è una libreria di componenti per JSF e un framework avanzato per integrare le funzionalità di AJAX all'interno di un'applicazione business. Le componenti ajax offerte da questa libreria estendono JSF permettendo di creare interfacce il cui livello di responsività si avvicina a quello delle normali interfacce desktop senza scrivere una singola riga di codice JavaScript.

Nel progetto si è fatto largo uso di RichFaces per generare pannelli, menù e tabelle al fine di facilitare il più possibile la navigazione agli utenti e delegare alcune operazioni al client web.

2.3.8 XDoclet 1.2.3

XDoclet è una libreria opensource per la generazione di codice XML che permette, tramite l'inserimento di particolari tag Javadoc, di programmare un'applicazione Java in modo Attribute-Oriented. Un programmatore, usando un paradigma di programmazione orientato agli attributi, può contrassegnare elementi di un programma (ad esempio classi o metodi) per indicare proprietà semantiche specifiche di un'applicazione o di un dominio.

Nel progetto è stata utilizzata XDoclet per generare la descrizione di proprietà relative ad EJB e Servlet. In aggiunta ai tag standard della libreria sono stati utilizzati meccanismi chiamati "merge point" che permettono l'inserimento di dichiarazioni non derivate da template chiamati. Queste personalizzazioni sono state necessarie poichè alcune librerie impiegate non sfruttano i tag XDoclet.

2.3.9 Eclipse 3.1.2

Eclipse è una piattaforma opensource estendibile che si propone di facilitare il programmatore fornendo tool di supporto a tutte le fasi del ciclo di vita del software.

Nel progetto, in aggiunta ad alcune funzionalità base quali sistemi di refactoring e di generazione e correzione automatica del codice che rendono Eclipse uno dei migliori Integrated Development Environment, ci si è avvalsi in particolare di due plugin: Subclipse e JBossIDE.

2.3.10 JBossIDE 1.6.0GA

JBoss-IDE è un plugin per l'ambiente di sviluppo Eclipse che consente di integrare all'interno del framework Eclipse la creazione e la gestione di applicazioni JBoss. In particolare JBossIDE offre un estensivo e intuitivo supporto ad XDoclet, un sistema di monitoraggio delle attività del server JBoss e di debugging e inspection delle applicazioni in fase di sviluppo, meccanismi semplici per la configurazione del layout degli archivi contenenti i pacchetti software e per il deploy degli stessi all'interno dell'application server, procedure guidate che facilitano lo sviluppo di applicazioni J2EE e un miglior supporto all'editing dei file aventi un formato utilizzato all'interno di JBoss.

2.3.11 Subclipse 1.0.6

Subclipse è un plugin per Eclipse che permette a team di sviluppatori di collaborare tra loro. Lo scopo di questa estensione è fornire all'interno dell'ambiente di sviluppo le funzionalità del sistema di controllo delle revisioni Subversion.

2.3.12 Subversion 1.4.2dfsg1-2

Subversion (svn) è un sistema centralizzato per il controllo del "versioning" di file e directory. *svn* offre molti vantaggi rispetto ad altri software del suo dominio, tra cui: atomicità delle operazioni di "commit" che, se interrotte, non causano quindi inconsistenze o corruzioni; tracciabilità di dati e metadati; supporto ai file di tipo binario; architettura client-server; supporto WebDAV.

Durante lo sviluppo del progetto è stato utilizzato un server *svn* presen-

te su un computer privato raggiungibile remotamente tramite internet. L'accesso al servizio si può ottenere sia in modalità lettura-scrittura tramite il protocollo svn+ssh (previa autenticazione) o in sola lettura tramite WebDAV su HTTPS.

2.4 Problematiche Riscontrate

L'utilizzo delle soluzioni software elencate nella sezione precedente ha portato in evidenza alcuni malfunzionamenti delle stesse. Il processo di analisi degli errori e di ricerca dei "bug" è stato talvolta lungo e difficoltoso e non ha sempre portato a una risoluzione o all'individuazione di "workaround" efficienti. Vengono qui riportate le più significative tra problematiche riscontrate:

- **PermGen error:**[2] durante lo sviluppo, dopo un numero casuale di "redeploy" effettuati per testare le componenti, il framework JBoss si bloccava con il messaggio *"java.lang.OutOfMemoryError: PermGen space"*. Questo errore sembra essere causato da un errato riuso della memoria da parte della "java virtual machine" di Sun e al momento l'unica soluzione è usare JBoss con un'altra JVM. Un workaround applicato, che però ritarda solamente il manifestarsi del "crash", è stato aumentare lo spazio disponibile per l'heap PermGen tramite le opzioni "-XX:PermSize" e "-XX:MaxPermSize".

- **Axis e Hot Deploy:**[3] La versione di Axis utilizzata non supporta correttamente l'hot-deploy di un'applicazione, ovvero non riesce a

[2]Maggiori dettagli e riferimenti alla pagina: http://blog.yannis-lionis.gr/?p=8

[3]Un esempio di discussioni su questo problema è presente nell'archivio della mailing list "jboss-user": http://www.mail-archive.com/jboss-user%40lists.sourceforge.net/msg33068.html

rigenerare la cache degli oggetti remoti fallendo quindi nel lookup degli stessi e rendendo inutilizzabili le invocazioni dei metodi remoti. Il "workaround" per questo problema è il riavvio di JBoss per ottenere un "deploy" non hot delle applicazioni.

- **Facelets "loadImplicit":**[4] introducendo Facelets nelle applicazioni abbiamo notato che, durante la prima visita delle pagine, venivano stampati nella console di JBoss numerosi errori nonostante il corretto funzionamento del software. La causa di questo è semplicemente un problema di tipologie di logging all'interno del compilatore Facelets, è stato quindi deciso di ignorare questi errori.

- **Axis log4j:**[5] se si segue la procedura di installazione di Axis consigliata nelle guide disponibili in rete si ottengono errori relativi al "logging" in fase di avvio dell'application server. La soluzione è stata rimuovere sia il ".jar" che il file di configurazione del sottosistema di logging di Axis per permettere a quest'ultimo di utilizzare quello fornito da JBoss.

[4]Maggiori dettagli e riferimenti alla pagina: http://shrubbery.mynetgear.net/wiki/Facelets_'loadImplicit'_error

[5]Maggiori dettagli e riferimenti alla pagina: http://www.jasperforge.org/index.php?option=com_joomlaboard&Itemid=215&func=view&id=22175&catid=10

Capitolo 3

Guide

Il seguente capitolo presenta il sistema sviluppato. La prima sezione è dedicata a chi deve proseguire lo sviluppo o semplicemente studiare il software: in essa vengono illustrati i dettagli implementativi. La seconda sezione è dedicata agli amministratori, si presenta infatti un piccolo ma dettagliato esempio di installazione e configurazione effettuato su un sistema operativo Unix.

3.1 Guida per lo Sviluppatore

È possibile reperire i sorgenti sia negli archivi presenti nel "tarball" consegnato che tramite *svn over WebDAV*[1].

Una volta importati i progetti all'interno di Eclipse-JBossIDE è necessario effettuare alcune configurazioni per ottenere un ambiente di lavoro correttamente compilabile:

[1]Importando i progetti *Atenarium* e *Departmentarium* dall'indirizzo https://shammash.homelinux.org/svn/middleware/trunk/

- in “Properties” → “XDoclet” selezionare *Apply* per adattare la configurazione di XDoclet al nuovo workspace;
- ricreare il codice generato automaticamente da XDoclet tramite “Run XDoclet”;
- compilare tramite “Run Packaging”.

3.1.1 Osservazioni generali

Prima di descrivere in modo dettagliato la struttura delle applicazioni esaminandole package per package è opportuno menzionare alcuni assunti fatti durante il design dell’applicazione che possono facilitare l’approccio alla lettura del codice:

- nessuna eccezione viene gestita nel container ejb per evitare il sovraccarico di questa parte dell’application server, l’handling viene quindi demandato al client;
- sempre per snellire il numero di informazioni presenti nel container ejb i session bean sono stateless, per questo motivo si rimanda al client la gestione della sessione.

3.1.2 Departmentarium

departmentarium.ejb

Questo package contiene gli Enterprise Java Bean. Sono stati utilizzati due tipi di ejb: gli entity e i session (stateless).

- **Departmentarium:** ejb di tipo session (stateless). Questo ejb ricopre il ruolo di *façade* tra gli ejb e i client. In questo modo è stato

possibile creare una struttura più semplice da comprendere e da mantenere poichè i client utilizzano metodi generici e non operano direttamente sugli ejb di tipo entity. Siccome l'ejb è stateless ogni metodo richiede come parametri *username* e *password* per verificare che l'utente sia autorizzato ad eseguire l'operazione che richiede.

- **Configuration:** ejb di tipo entity in cui vengono memorizzate le informazioni relative alla configurazione di Departmentarium. La sua unicità è garantita da una sorta di *singleton* realizzato definendo la chiave primaria con una costante in modo da ottenere un'eccezione nel caso si provi a creare più di un ejb di questo tipo. Ha un metodo business che si occupa di fornire su richiesta un *ReportId*; per evitare possibili problemi di concorrenza questo metodo è stato definito come *synchronized*.
- **User:** ejb di tipo entity che rappresenta un utente del sistema.
- **Report:** ejb di tipo entity che contiene sia la pubblicazione che i dati ad essa relativi.

Tra "Report" e "User" è stata definita una *Container Managed Relationship* (1 - N) che ci permette di risalire direttamente dall'utente ai report pubblicati e viceversa tramite la semplice invocazione di un metodo. Grazie a questa relazione è stato dimezzato il numero di "query" ottenendo quindi sia un incremento prestazionale che una semplificazione dell'applicazione.

departmentarium.interfaces

Questo package, generato tramite XDoclet, contiene le interfacce locali e remote degli ejb descritti precedentemente.

departmentarium.utils

Questo package e i suoi sottopackage contengono classi di utilità generale usate in altre parti dell'applicazione.

- **departmentarium.utils.pdf:** sottopackage contenente classi improntate all'handling dei file pdf e dei metadati.
- **departmentarium.utils.wrappers:** sottopackage contenente i *java bean* utilizzati nella comunicazione tra l'ejb-container e i client.
- **departmentarium.utils.exceptions** sottopackage contenente le definizioni delle eccezioni.
- **Constants** classe che colleziona le costanti utilizzate in vari punti dell'applicazione rendendone più facile e più organica sia l'amministrazione che la modifica.
- **EJBGetter** classe che esporta metodi comuni per il lookup degli ejb evitando quindi la duplicazione di codice.
- **WebServiceManager** classe che esporta i metodi per eseguire le chiamate a procedure remote.

departmentarium.web

Questo package e i suoi sottopackage contengono le classi i cui oggetti istanziati risiederanno nel web-container.

- **RegistrationServlet:** servlet di tipo *ContextListener* dedita alla registrazione (in fase di avvio) e rimozione (in fase di spegnimento) su Atenarium delle informazioni relative al dipartimento.

- **ReportReturnServlet:** servlet che gestisce le richieste di download dei file pdf dei report.

departmentarium.web.utils.beans

Questo package contiene i *java bean* utilizzati dalle *jsf*.

- **Time:** bean che gestisce le date e il cui scope è a livello *applicazione*.
- **User:** bean avente scope a livello *sessione* che gestisce il login e il logout di un utente. Al suo interno viene mantenuto un riferimento all'ejb stateless Departmentarium per velocizzare l'applicazione evitando di effettuare il lookup ad ogni operazione. In fase di login viene stabilito quali siano le azioni che l'utente può compiere tramite un controllo sul tipo di utente restituito da Departmentarium.

departmentarium.web.utils.beans.actions

Questo package contiene l'interfaccia *ActionBean* e le varie azioni implementate. Siccome è molto probabile che in futuro l'applicazione venga espansa con nuovi tipi di azioni si è cercato di rendere più semplice possibile il processo di estensione offrendo un'interfaccia comune. Una volta definito un nuovo tipo di azione è sufficiente inserire la sua istanziazione in "UserJavaBean.setInfos" per rendere disponibile all'utente la nuova funzionalità.

departmentarium.webservice

Questo package contiene i metodi dell'applicazione esposti tramite WebService. Per coerenza descrittiva nella directory del progetto che rap-

presenta questo package sono stati posti anche i file xml che definiscono i parametri di "deploy" del WebService.

web

Questa directory contiene i file dell'interfaccia web. "welcome.xhtml" è la descrizione della pagina principale del sito e include le varie parti che occorrono nella generazione del *template* di una pagina ("templates/template.xhtml").

La componente più importante di "welcome.xhtml" è l'inclusione dell'*actionsViewer* ("templates/actionsViewer.xhtml"): a seconda dell'azione selezionata nel menu "Departmentarium" dell'interfaccia web, *actionsViewer* contatta il java bean opportuno e include nella pagina l'xhtml appropriato.

3.1.3 Atenarium

atenarium.ejb

Questo package contiene gli Enterprise Java Bean. Sono stati utilizzati due tipi di ejb: gli entity e i session (stateless).

- **Atenarium:** ejb di tipo session (stateless). Questo ejb ricopre il ruolo di *façade* tra gli ejb e i client. In questo modo è stato possibile creare una struttura più semplice da comprendere e da mantenere poichè i client utilizzano metodi generici e non operano direttamente sugli ejb di tipo entity.
- **Department:** ejb di tipo entity che rappresenta un dipartimento registrato tramite WebService.

atenarium.interfaces

Questo package, generato tramite XDoclet, contiene le interfacce locali e remote degli ejb descritti precedentemente.

atenarium.utils

Questo package e i suoi sottopackage contengono classi di utilità generale usate in altre parti dell'applicazione.

- **Constants** classe che colleziona le costanti utilizzate in vari punti dell'applicazione rendendone più facile e più organica sia l'amministrazione che la modifica.
- **EJBGetter** classe che esporta metodi comuni per il lookup degli ejb evitando quindi la duplicazione di codice.

atenarium.utils.report

- **ReportDetails:** classe che rappresenta gli attibuti di un report presenti in un dipartimento (e quindi scambiati tramite WebService).
- **DepartmentReportDetails:** *decorator* di ReportDetails che aggiunge il nome del dipartimento e il link per effettuare il download del pdf, informazioni a corredo utili durante la creazione della lista di report da presentare all'utente di Atenarium.

atenarium.web

Questo package e i suoi sottopackage contengono le classi i cui oggetti istanziati risiederanno nel web-container.

- **ActionServlet:** servlet “dumb”, definita esclusivamente per poter sfruttare le funzionalitŝ di XDoclet. In questo modo vengono automaticamente generati i riferimenti all’ejb Atenarium.

atenarium.web.utils.beans

Questo package contiene i *java bean* utilizzati dalle *jsf*.

- **Time:** bean che gestisce le date e il cui scope è a livello *applicazione*.
- **Visit:** bean che gestisce le visite degli utenti e il cui scope è a livello *sessione*. Al suo interno viene mantenuto un riferimento all’ejb stateless Atenarium per velocizzare l’applicazione evitando di effettuare il lookup ad ogni operazione. All’inizio della sessione vengono caricati i filtri che l’utente può applicare alla lista visualizzata. Il metodo “getDepartmentsReportDetails” applica i filtri alla lista in modo da minimizzare ogni volta il numero di elementi da filtrare.

atenarium.web.utils.beans.filters

Questo package contiene la classe astratta *FilterBean* e i vari filtri implementati. Contrariamente a quanto fatto per Departmentarium in questo caso si è preferito ricorrere ad una classe astratta anzichè ad un’interfaccia data la maggior percentuale di codice condiviso tra i vari filtri.

atenarium.webservice

Questo package contiene i metodi dell’applicazione esposti tramite WebService. Per coerenza descrittiva nella directory del progetto che rap-

presenta questo package sono stati posti anche i file xml che definiscono i parametri di “deploy” del WebService.

web

Questa directory contiene i file dell’interfaccia web. “welcome.xhtml” è la descrizione della pagina principale del sito e include le varie parti che occorrono nella generazione del *template* di una pagina (“templates/template.xhtml”).

La componente più importante di “welcome.xhtml” è l’inclusione di *menu* (“templates/menu.xhtml”) che cura i parametri inseriti dall’utente nei vari filtri sia invocando la visualizzazione di elenchi che richiedendo l’aggiornamento della lista dei report visualizzati.

3.2 Guida per l’Amministratore

Questo guida illustra i passi essenziali per installare e configurare correttamente il sistema software precedentemente descritto. La procedura viene descritta usando la sintassi tipica di una “command-line” UNIX per facilitare la comprensione da parte degli amminstratori di sistema. La sequenza di azioni da svolgere si può suddividere in due fasi: la prima in cui si installa e configura il framework destinato ad ospitare le applicazioni, la seconda in cui si installano e configurano le applicazioni vere e proprie.

3.2.1 Installazione e configurazione Framework

Lo scopo di questa fase è ottenere un application server j2ee che offra tutte le funzionalità richieste dal software da noi sviluppato: un appli-

cation server JBoss con le ultime librerie MyFaces disponibili corredato con il WebService framework Axis.

```
$ wget http://kent.dl.sourceforge.net/sourceforge/jboss/
jboss-4.0.5.GA.zip
$ unzip jboss-4.0.5.GA.zip


$ rm jboss-4.0.5.GA/server/default/deploy/
jbossweb-tomcat55.sar/jsf-libs/*.jar
$ wget http://apache.fagioli.biz/myfaces/binaries/
myfaces-core-1.1.5-bin.tar.gz
$ tar xfz myfaces-core-1.1.5-bin.tar.gz
$ cp myfaces-core-1.1.5/lib/*.jar jboss-4.0.5.GA/server/
default/deploy/jbossweb-tomcat55.sar/jsf-libs/


$ wget http://apache.fagioli.biz/ws/axis/1_4/
axis-bin-1_4.tar.gz
$ tar xfz axis-bin-1_4.tar.gz
$ mkdir jboss-4.0.5.GA/server/default/deploy/webapps/
$ cp -r axis-1\_4/webapps/axis jboss-4.0.5.GA/server/
default/deploy/webapps/axis.war
$ rm jboss-4.0.5.GA/server/default/deploy/webapps/
axis.war/WEB-INF/lib/log4j-1.2.8.jar
```

3.2.2 Installazione e configurazione Applicazioni

In questa fase utilizzeremo l'application server configurato nella sezione precedente per installare e configurare le applicazioni che soddisfano i requisiti del progetto. Si assume che nella directory corrente siano

presenti gli archivi contenenti le due parti di progetto già compilate, reperibili dal "tarball" consegnato. Alternativamente all'installazione dei package binari si può procedere con l'installazione tramite subversion, discussa nel Guida per lo Sviluppatore.

"Deploy" ear

"Deploy" applicazioni su JBoss:

```
$ tar xfz smw_project/Departmentarium.tar.gz
$ cp Departmentarium/packages/DepartmentariumApp.ear
jboss-4.0.5.GA/server/default/deploy/
$ tar xfz smw_project/Atenarium.tar.gz
$ cp Atenarium/packages/AtenariumApp.ear
jboss-4.0.5.GA/server/default/deploy/
```

Avvio di JBoss:

```
$ cd jboss-4.0.5.GA
$ sh bin/run.sh
```

"Deploy" WebService

"Deploy" dei WebService su Axis:

```
$ export AXIS_CP=`for jar in axis-1_4/lib/*.jar ;
do echo -n $jar':' ; done | sed 's/:$/\n/'`
$ java -cp $AXIS_CP org.apache.axis.client.AdminClient
-p8080 Departmentarium/src/departmentarium/
webservice/deployDepartmentariumWS.wsdd
$ java -cp $AXIS_CP org.apache.axis.client.AdminClient
-p8080 Atenarium/src/atenarium/webservice/
deployAtenariumWS.wsdd
```

Configurazione Amministratore Departmentarium

Definizione dell'utente amministratore di Departmentarium tramite il pannello di amministrazione di Hypersonic:

- aprire il browser all'indirizzo "http://<hostname>:<port>/jmx-console";
- selezionare il collegamento "database=localDB,service=Hypersonic";
- invocare il metodo "void startDatabaseManager".

Nella nuova finestra inserire ed eseguire il comando SQL:

```
insert into public.xuser values('<username>','<password>',
null,true);
```

Configurazione Departmentarium

Configurazione di Departmentarium con l'utente amministratore:

- aprire il browser all'indirizzo "http://<hostname>:<port>/departmentarium/";
- effettuare login con username e password precedentemente inseriti;
- nel menu "Departmentarium" selezionare "Setup Departmentarium";

Compilare i campi nel seguente modo:

- *Departmentarium Name:* Computer Science;
- *Departmentarium Host:* http://<hostname>:<port>;

- *Departmentarium WebSite Name:* departmentarium;
- *Departmentarium WebService Name:* axis/services/DepartmentariumWS;
- *Id Prefix:* CS;
- *Atenarium WebService Url:* http://<hostname>:<port>/axis/services/AtenariumWS.

Bibliografia

[Arm05] Armstrong, E. ; et. al.; *The J2EE 1.4 Tutorial*, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/, 2005

[BBS04] Bates, B. ; Basham, B. ; Sierra, K. ; *Head first Servlets and JSP*, USA, O'Reilly, 2004

[Blo01] Bloch, J. ; *Effective Java Programming Language Guide*, California USA, Addison Wesley, 2001

[Bur87] Burbeck, S. ; *Applications Programing in Smalltalk-80: How to use Model-View-Controller (MVC)*, 1987

[GHJV95] Gamma, E. ; Helm, R. ; Johnson, R. ; Vlissides, J. ; *Design Patterns: Elements of Reusable Object-Oriented Software*, USA, Addison Wesley Professional, 1995

[JBo06] JBoss Inc. ; *Getting Started with JBoss 4.0*, http://docs.jboss.org/jbossas/getting_started/v5/html/, 2006

[Man05] Mann, K. D. ; *JavaServer Faces in Action*, USA, Manning Publications, 2005

[Shi06] Shirazi, J. ; *Tuning EJBs*, in *Java Performance Timing*, USA, O'Reilly, 2006