Tecnologie di Sviluppo per il Web

Applicazioni Web J2EE: Struttura dell'Applicazione

versione 3.1

G. Mecca – mecca@unibas.it – Università della Basilicata

Sommario

- Struttura di un'Applicazione J2EE
  - ⇨Organizzazione Standard dei File
  - ⇨Appendice: Deployment Descriptor
  - ⇨URI delle Risorse
- Installazione ("Deployment")
  - ⇨Il manager di Tomcat
- Utilizzo di Ant
- Appendice

Struttura di un'Applicazione J2EE

- Applicazione Web J2EE
  - una cartella montata sul file system virtuale del contenitore
  - deve avere una struttura definita
  - ed essere visibile sul file system virtuale del contenitore
- Il file system virtuale
  - corrisponde al contenuto di una cartella radice chiamata comunemente "webapps"

Struttura di un'Applicazione J2EE

- Esempio: Tomcat
  - radice delle applicazioni predefinita: %TOMCAT_HOME%/webapps
- Applicazioni Web predefinite
  - ROOT (/) (pagina di benvenuto)
  - examples (esempi)
  - tomcat-docs (documentazione)
  - manager (gestore delle applicazioni)
  - admin (amministrazione)

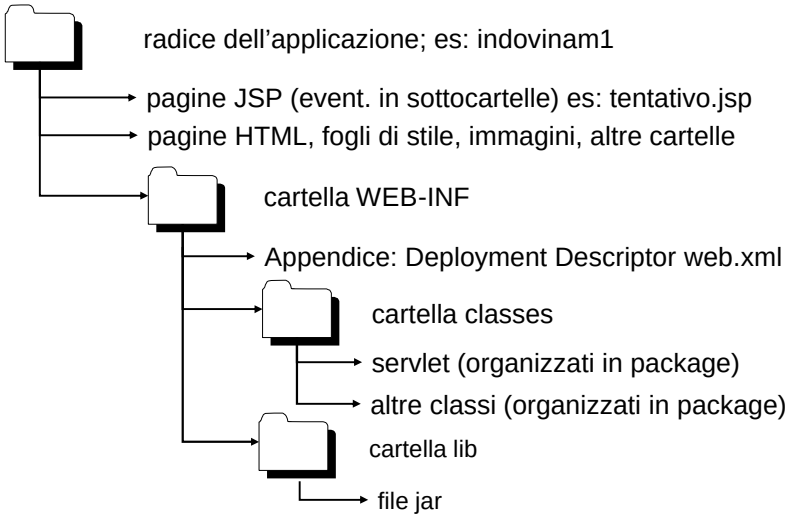
Organizzazione Standard dei File

- Struttura dell'applicazione
  - ⇨fissata dallo standard
  - ⇨è necessario che tutte le applicazioni seguano la struttura standard
  - ⇨garantisce la portabilità tra contenitori
  - ⇨se un'applicazione nella cartella webapps non rispetta la struttura il server non la rende visibile

Organizzazione Standard dei File

- Organizzazione dei file nella cartella
  - ⇨radice : pagine JSP, pagine HTML, fogli di stile CSS, immagini (tipicamente organizzati in sottocartelle)
  - ⇨cartella WEB-INF: "deployment descriptor" web.xml
  - ⇨cartella WEB-INF/classes: servlet e componenti (tipicamente organizzati in package)
  - ⇨cartella WEB-INF/lib: file jar che è necessario rendere visibili nell'applicazione (es: driver JDBC)

Organizzazione Standard dei File



Organizzazione Standard dei File

- Attenzione
  - ⇨l'applicazione Web viene eseguita dal contenitore e non dall'utente
  - ⇨non è possibile controllarne direttamente il classpath, ma bisogna rispettare delle regole
  - ⇨in particolare, il classpath per l'applicazione Web è formato solo dalle classi e dai jar contenuti in particolari cartelle

Organizzazione Standard dei File

- Il classpath per l'applicazione Web
  - ⇨tutte le classi in WEB-INF/classes
  - ⇨tutti i jar in /WEB-INF/lib
- Inoltre
  - ⇨è possibile configurare il contenitore in modo che ci siano classi e/o jar visibili a tutte le applicazioni Web

Organizzazione Standard dei File

- Componenti visibili a tutte le applicazioni
  - ⇨le classi in %TOMCAT_HOME%\shared\classes
  - ⇨tutti i jar in %TOMCAT_HOME%\shared\lib
  - ⇨es: log4j.jar
- Componenti visibili a tutte le applicazioni
  - ⇨e anche al contenitore (es: driver jdbc)
  - ⇨le classi in %TOMCAT_HOME%\common\classes
  - ⇨tutti i jar in %TOMCAT_HOME%\common\lib

Organizzazione Standard dei File

- In queste cartelle
  - ⇨due jar fondamentali
- servlet-api.jar
  - ⇨package javax.servlet, javax.servlet.http, javax.servlet.resources
- jsp-api.jar
  - ⇨package javax.servlet.jsp, javax.servlet.jsp.el, javax.servlet.tagext. javax.servlet.resources

Organizzazione Standard dei File

- Web Application Archive (WAR)
  - ⇨le applicazioni Web sono spesso distribuite sotto forma di file “war”
  - ⇨archivio jar con estensione “.war”
  - ⇨si creano e si gestiscono con jar
  - ⇨l’archivio deve rispettare l’organizzazione delle cartelle dell’applicazione Web
  - ⇨i contenitori sono in grado di decompattare ed installare automaticamente i file war

Appendice: Deployment Descriptor

- Appendice: Deployment Descriptor web.xml
  - ⇨“deployment descriptor”
  - ⇨serve a specificare parametri specifici per l’applicazione
  - ⇨es: nomi per i servlet; URI per i servlet; file indice standard (es: index.jsp); “timeout” per le sessioni ecc.
  - ⇨è indispensabile, ma può essere vuoto (il contenitore assegna valori standard)

Appendice: Deployment Descriptor

- Sintassi
  - ⇨deve essere un documento XML ben formato
  - ⇨deve essere valido rispetto ad uno schema XML fissato dallo standard:
    http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd
  - ⇨fino alla versione 2.3:
    http://java.sun.com/dtd/web-app_2_3.dtd

Appendice: Deployment Descriptor

- Un descrittore “minimale”

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
                http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd„
                version="2.4">

</web-app>
```


Appendice: Deployment Descriptor

- Un descrittore completo
  - ⇨viene fornito a corredo di Tomcat
  - ⇨ed è raggiungibile a partire dalla documentazione
  - ⇨mostra molti dei possibili elementi previsti dallo schema e può essere utilizzato come base per sviluppare ulteriori descrittori

>> Application Development Guide – Deployment – “basic web-xml file”

Appendice: Deployment Descriptor

- Alcuni elementi interessanti
- Elemento session-config
  - definisce il tempo massimo di inattività di una sessione (in minuti); per Tomcat standard 30
  - sottoelemento: session-timeout
- Esempio

```
<session-config>
  <session-timeout>15</session-timeout>
</session-config>
```


Appendice: Deployment Descriptor

- Elemento error-page
  - serve a specificare l'URI di una pagina da visualizzare in caso errori, ovvero:
  - risposte HTTP di errore (codici 4xx e 5xx)
  - eccezioni nel codice Java
- Sottoelementi
  - error-code
  - exception-type
  - location

Appendice: Deployment Descriptor

- Esempio

```
<error-page>
  <error-code>404</error-code>
  <location>/errore.html</location>
</error-page>

<error-page>
  <exception-type>
   javax.servlet.ServletException
</exception-type>
  <location>/errore.html</location>
</error-page>
```


Appendice: Deployment Descriptor

- Attenzione
  - ⇨gli elementi devono comparire in web.xml nell'ordine previsto dallo schema relativo
- Un descrittore tipico
  - ⇨display-name
  - ⇨description
  - ⇨servlet
  - ⇨servlet-mapping
  - ⇨error-page

>> indovinam1servlet

URI delle Risorse

- Ogni risorsa dell'applicazione ha un URI
  - ⇨http://<host>:<porta>/<percorso risorsa>
  - ⇨<porta>: normalmente diversa da 80
  - ⇨esempio, per Tomcat: 8080
- Esempio:
  - ⇨http://127.0.0.1:8080/ (pagina di benvenuto)
  - ⇨http://127.0.0.1:8080/manager/html/list
  - ⇨http://127.0.0.1:8080/indovinam1/index.jsp

URI delle Risorse

- Nome dell'applicazione ("context path")
  - ⇨ogni applicazione ha un nome detto "context path" che viene utilizzato negli URI delle risorse
  - ⇨il nome corrisponde normalmente al nome della cartella in cui è contenuta l'applicazione (es: indovinam1)
  - ⇨ma può essere alterato utilizzando gli strumenti di configurazione del contenitore

URI delle Risorse

- Pagine JSP
  - ⇨percorso a partire da “webapps”
  - ⇨/<contextPath>/<percorso>/<nomeFile>
  - ⇨es: /indovinam1/fine.jsp
  - ⇨es: /studenti/inserimenti/inserisciEsame.jsp
- Pagine HTML e connessi
  - ⇨stessa convenzione delle pagine JSP

URI delle Risorse

- Servlet
  - ⇨un servlet non ha un URI definito
  - ⇨risponde a tutte le richieste corrispondenti a “URL pattern” definiti per quel servlet nel deployment descriptor
- Elemento servlet
  - ⇨consente di dare un nome ad un servlet
  - ⇨sottoelementi: servlet-name, servlet-class

URI delle Risorse

- Sintassi

```
<servlet>
    <servlet-name>Esci</servlet-name>
    <servlet-class>indovina.ServletEsci</servlet-class>
</servlet>

<servlet>
    <servlet-name>Errore</servlet-name>
    <servlet-class>indovina.ServletErrore</servlet-class>
</servlet>
```


URI delle Risorse

- Elemento servlet-mapping
  - ⇨deve comparire dopo tutti gli elementi servlet
  - ⇨associa uno o più URI alternativi a quello standard ad un servlet
  - ⇨per fare riferimento al servlet (o alla pagina Jsp) si utilizza il nome (servlet-name) assegnato con l'elemento servlet

URI delle Risorse

- Sintassi:

```
<servlet-mapping>
     <servlet-name>Esci</servlet-name>
     <url-pattern>/Esci</url-pattern>
</servlet-mapping>

<servlet-mapping>
     <servlet-name>Errore</servlet-name>
     <url-pattern>/Errore</url-pattern>
</servlet-mapping>
```


URI delle Risorse

- E' possibile specificare classi di URI
  - ⇨collezioni di URI che unificano con il pattern
  - ⇨carattere speciale *: qualsiasi stringa
- Esempio: URI che terminano con .asp

```
<servlet-mapping>
  <servlet-name>Errore</servlet-name>
  <url-pattern>*.asp</url-pattern>
</servlet-mapping>
```

pattern; attenzione:
non comincia per "/"

Installazione (“Deployment”)

- Differenza con il server HTTP ordinario
  - per rendere eseguibile un’applicazione non basta renderla visibile sul file system virtuale del contenitore
  - è necessaria una operazione aggiuntiva
- Deployment
  - operazione di “messa in opera” di un’applicazione Web che prepara i componenti all’uso

Installazione (“Deployment”)

- Operazioni durante il deployment
  - le attività cambiano da server a server
  - normalmente vengono creati e inizializzati i servlet dell’applicazione
  - vengono compilate le pagine JSP
  - vengono caricati i componenti (Bean)
  - viene predisposto il pool di thread di servizio per l’applicazione

Installazione (“Deployment”)

- Procedura di deployment
  - ⇨lo standard prevede un’unica modalità
  - ⇨copiare il file .war dell’applicazione nella cartella webapps
  - ⇨il contenitore dovrebbe riconoscere il nuovo file, assegnargli un context path uguale al nome del file war, decomprimerlo in una cartella con lo stesso nome ed effettuare automaticamente il deployment

Installazione (“Deployment”)

- Altre operazioni importanti
  - ⇨“undeployment”: rimozione dell’applicazione dal contenitore; ATTENZIONE: equivale e rimuovere fisicamente la cartella ed il war da webapps
  - ⇨“ricaricamento” (“reload”): re-inizializzazione dei componenti dell’applicazione senza dover effettuare una rimozione ed una nuova installazione

Installazione (“Deployment”)

- In concreto
  - ⇨esistono vari modi per effettuare il deployment e l’undeployment
  - ⇨I modo: copiare fisicamente/rimuovere il file war o la cartella decompressa in webapps
  - ⇨II modo: utilizzare gli strumenti di gestione forniti dal contenitore
  - ⇨III modo: utilizzare Ant

Il Manager di Tomcat

- Applicazione Web fornita con Tomcat
  - ⇨consente di effettuare le principali operazioni sulle applicazioni Web
  - ⇨installazione, rimozione, ricaricamento
- Per utilizzare il manager
  - ⇨è necessario autenticarsi con nome utente e password come un utente di ruolo “manager” precedentemente creato

Il Manager di Tomcat

>> /manager

- Utilizzo dell'applicazione
  - ⇨attraverso il browser
  - ⇨context path: /manager
- Utilizzo dell'interfaccia HTML
  - ⇨per installare un'applicazione basta fornire il context path e l'URI della cartella relativa o del file war
  - ⇨es: file:///e:/temp/indovinam1

Utilizzo di Ant

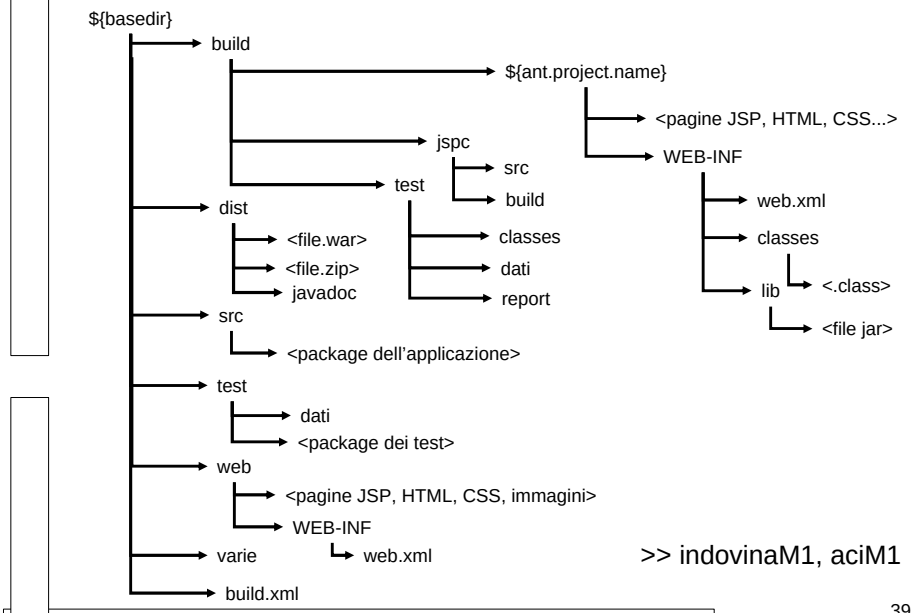
- In processi di sviluppo industriali
  - ⇨Ant rappresenta la soluzione ideale per le attività di sviluppo delle applicazioni Web
- Due aspetti nuovi
  - ⇨organizzazione della cartella di progetto
  - ⇨task e target orientati alla gestione del contenitore

Utilizzo di Ant

- Organizzazione della cartella di progetto
  - ⇨leggermente diversa da quella delle applicazioni desktop
  - ⇨ma basata sullo stesso principio di separazione dei componenti
- Una nuova cartella di sorgenti: “web”
  - ⇨in cui sono contenute pagine JSP, pagine HTML, immagini, fogli di stile
  - ⇨oltre al deployment descriptor

Utilizzo di Ant

- Una nuova cartella di build
  - ⇨/build/${ant.project.name}
  - ⇨in cui viene ricostruita tutta la struttura dell’applicazione Web rimettendo assieme i vari componenti
  - ⇨secondo la struttura dello standard J2EE
- Nota
  - ⇨serve anche una cartella per la compilazione delle pagine JSP (/build/jspc)



>> indovinaM1, aciM1

Utilizzo di Ant

- I nuovi task
  - ⇨Tomcat fornisce una serie di task di Ant pronti per l'utilizzo
  - ⇨consentono di effettuare la compilazione delle pagine JSP prima dell'installazione
  - ⇨il deployment
  - ⇨l'undeployment
  - ⇨altre operazioni basate sul manager

Utilizzo di Ant

>> tomcat-template-build.xml

- Per utilizzarli
  - ⇨è necessario utilizzare il task tasdkef per introdurre i nuovi task
- I target principali
  - ⇨compile-jsps
  - ⇨install, uninstall, reinstall (lavora con la cartella)
  - ⇨deploy, undeploy, redeploy (lavorano con il war)

Utilizzo di Ant

>> web-template-build.xml
>> web-dbms-template-build.xml

- Nota
  - ⇨per poterli utilizzare è necessario fornire i parametri per l'accesso al manager
  - ⇨host, porta, nome utente, password
  - ⇨in dependent.properties
- Modelli di file di build
  - ⇨uno per le applicazioni a due livelli
  - ⇨uno per le applicazioni a tre livelli

Utilizzo di Ant

- NetBeans
  - ⇨al solito, la gestione dei progetti Web di NetBeans è basata su Ant
  - ⇨è opportuno definire i progetti come progetti “freeform” utilizzando i propri file di build
- In alternativa
  - ⇨è possibile utilizzare i progetti Web predefiniti di NetBeans, che ubbidiscono alla stessa struttura con piccole varianti

Utilizzo di Ant

- Le varianti principali
  - ⇨NetBeans ricostruisce l’applicazione Web in una cartella /build/web
  - ⇨inoltre NetBeans installa l’applicazione Web nel proprio Tomcat (fornito a corredo), che ascolta tipicamente sulla porta 8084
- Vantaggio principale
  - ⇨è possibile utilizzare il debugger anche per le applicazioni Web

Riassumendo

- Struttura di un'Applicazione J2EE
  - ⇨Organizzazione Standard dei File
  - ⇨Appendice: Deployment Descriptor
  - ⇨URI delle Risorse
- Installazione ("Deployment")
  - ⇨Il manager di Tomcat
- Utilizzo di Ant
- Appendice

Appendice: Deployment Descriptor

- Ordine degli elementi in web.xml
  - ⇨description: serve a dare una breve descrizione dell'applicazione Web
  - ⇨context-param: serve a definire parametri di inizializzazione validi per tutti i servlet/jsp
  - ⇨filter >>
  - ⇨filter-mapping >>
  - ⇨servlet
  - ⇨servlet-mapping

Appendice: Deployment Descriptor

- Ordine degli elementi in web.xml (cont.)
  - ⇨session-config
  - ⇨mime-mapping: definisce il tipo MIME per estensioni di file inusuali
  - ⇨welcome-file-list: definisce i file standard da servire in caso di URI che fanno riferimento a cartelle (es: index.jsp, index.html)
  - ⇨error-page

Appendice: Deployment Descriptor

- Ordine degli elementi in web.xml (cont.)
  - ⇨taglib >>
  - ⇨security-constraint: vincola l'accesso ad opportuni URI dell'applicazione
  - ⇨login-config: configura il tipo di autenticazione da usare per le risorse protette (es: basic o digest)

Appendice: Deployment Descriptor

- Sottoelemento <init-param> di <servlet>
  - definisce parametri di inizializzazione per il servlet o la pagina Jsp
  - sono accessibili nel codice Java
- Sottoelementi
  - param-name: nome del parametro
  - param-value: valore del parametro

Appendice: Deployment Descriptor

- Sintassi

```
<servlet>
 <servlet-name>Esci</servlet-name>
 <servlet-class>indovina.ServletEsci</servlet-class>
 <init-param>
   <param-name>nomeFileLog</param-name>
   <param-value>c:\tmp\logIndovina.txt</param-value>
 </init-param>
</servlet>
```

Appendice: Deployment Descriptor

- Accesso ai parametri di inizializzazione
  - si utilizza un oggetto ServletConfig
  - metodo ServletConfig getServletConfig()
  - sull'oggetto è possibile utilizzare il metodo String getInitParameter(Strin nomeParam)
- Esempio:

```
ServletConfig config=getServletConfig();
String nomeFileLog =
        config.getInitParameter("nomeFileLog");
```


Appendice: Deployment Descriptor

- Utilizzo
  - fornire parametri di inizializzazione consente di modificare l'applicazione senza ricompilare il codice
  - ma modificare web.xml richiede tipicamente di re-installare l'applicazione Web (non basta ricaricarla)
  - in alcuni casi può essere più conveniente utilizzare file di configurazione esterni

Termini della Licenza