Questo capitolo descrive il packaging.

Un’applicazione Java EE è contenuta in una o più unità standard per la distribuzione in ogni sistema di compilazione della piattaforma Java EE.

Ogni unità contiene:

* **componenti funzionali** come enterpreise bean, pagine web, servlet, applet.
* **Descrittore di distribuzione** opzionale che descrive il contenuto.

## 5.1 Applicazioni di packaging

Un’applicazione Java è contenuta in un file JAR, WAR o EAR.

I file WAR e EAR sono file JAR standard con estensione *.war* o .*ear*.

Usare questi file rende possibile assemblare diverse applicazioni Java EE usando alcuni componenti comuni.

Non c’è la necessità di scrivere altro codice, basta solo assemblare diversi moduli Java EE dentro i file JAR, WAR o EAR.

Un file EAR contiene moduli Java EE e, a volte, un deployment descriptor.

Il descrittore (deployment descriptor) è un documento XML che descrive i settaggi di distribuzione di un’applicazione, un modulo o un componente.

Dato che le informazioni contenute sono dichiarative possono essere modificate senza intervenire sul codice sorgente.

Al runtime il server Java EE legge il descrittore e mette mano all’applicazione, modulo o componente in maniera coerente alle informazioni contenute nel descrittore.

Le informazioni di distribuzione sono frequentemente specificate nel codice sorgente tramite annotazioni, se presente, un descrittore effettua l’override di ciò che è specificato nel codice sorgente.



I due tipi di un deployment descriptor sono Jave EE e runtime.

Un **Java EE Deployment Descriptor** è descritto dalla specifica Java EE e può essere usato per configurare i settaggi di distribuzione su ogni implementazione conforme a Java EE.

Un **Runtime Deployment descriptor** è utilizzato per configurare parametri specifici di un’implementazione Java EE.

Ad esempio il runtime deployment descriptor del server GlassFish contiene informazioni quali la context root di un’applicazione web e i parametri specifici dell’implementazione del server GlassFish come le direttive per le cache.

I GlashFish Server runtime deployment descriptors sono chimati *glassfish-modulType.xml* e sono posizionati nella stessa directory META-INF del Java EE deployment descriptor.

Un modulo Java EE consiste di uno o più componenti Java EE per lo stesso tipo di container e, opzionalmente, un component deployment descriptor dello stesso tipo.

Un deployment descriptor di un modulo enterprise bean, ad esempio, dichiara attributi di transazione e autorizzazioni di sicurezza per un enterprise bean.

Un modulo Java EE può essere distribuito come modulo stand-alone.

I moduli EE sono dei seguenti tipi:

* Moduli EJB: contengono class files per enterprise bean e, opzionalmente, un EJB deployment descriptor. I moduli EJB sono raccolti in file .jar.
* Moduli web: contengono file di classi servlet, file web, file di classi di supporto, GIF e file HTML e, opzionalmente, un web application deployment descriptor. I moduli web sono raccolti in file .war.
* Moduli Application Client: contengono file di classi e, opzionalmente, un application client deployment descriptor. Sono raccolti in file .jar.

Moduli resource adapter: contengono tutte le interfacce, le classi e le librerie native Java e, opzionalmente, un resource adapter deployment descriptor. Insieme implementano l’architettura del Connector per un particolare EIS. Sono raccolti in file .rar.

## 5.2 Packaging Enterprise Beans

Questo capitolo spiega come si possono raccogliere gli enterprise bean i moduli EJB JAR o WAR.

## 5.2.1 Packaging Enterprise Beans in EJB JAR Modules

Un file EJB JAR può essere usato da varie applicazioni.

Per assemblare un’applicazione Java EE bisogna raccogliere uno o più moduli, come file EJB JAR, dentro un file EAR, il file di archivio che raggruppa l’applicazione.

Quando si distribuisce il file EAR che contiene il file EJB JAR dell’enterprise bean si esporta in automatico anche l’enterprise bean nel server GlassFish.

Ovviamente è possibile esportare anche un file EJB JAR non contenuto in alcun file EAR.

La figura sottostante indica i contenuti di un file EJB JAR:



## 5.2.2 Packaging Enterprise Beans in WAR Modules

Gli enterprise bean spesso forniscono la logica business di un’applicazione web. In questi casi raggruppare gli enterprise beans nel modulo WAR semplifica la distribuzione e l’organizzazione dell’applicazione.

Gli enterprise beans possono essere raggruppati in un modulo WAR come classi Java o dentro un file JAR contenuto dentro il modulo WAR.

Per includere una classe in un modulo WAR, la classe dovrebbe trovarsi nella directory *WEB-IN/classes.*

Per includere un file JAR che contiene enterprise beans all’interno di un modulo WAR bisogna aggiungere il JAR alla directory *WEB-INF/lib* del modulo WAR.

I moduli WAR contenenti enterprise beans non richiedono un *ejb-jar.xml* deployment descriptor ma se l’appliczione lo utilizza va collocato nella directory *WEB-INF* del modulo WAR.

I file JAR che contengo classi enterprise bean raggruppate in un modulo WAR non sono considerati file EJB JAR anche se i file JAR contenutivi si attengono al formato di un file EJB JAR.

Gli enterprise beane contenuti in un file JAR sono semanticamente uguali agli enterprise beans contenuti nella directory *WEB-INF/classes* del modulo WAR e il namespace dell’ambiente di tutti gli enterprise beans ha come ambito di validità il modulo WAR.

Ad esempio supponiamo che un’applicazione sia costituita da un enterprise bean che fa da carrello della spesa, un enterprise bean che processa la carta di credito e una Java servlet che fa da front end.

Il carrello della spesa offrirebbe una vista locale (senza interfacce) e sarebbe scritto così:

*package com.example.cart;*

*@Stateless*

*public class CartBean { ... }*

Il bean che processa la carta di credito sarebbe raggruppato nel proprio file JAR , *cc.jar*, e offrirebbe anch’esso una vista locale e senza interfacce tipo la seguente:

*package com.example.cc;*

*@Stateless*

*public class CreditCardBean { ... }*

La servlet, *com.example.web.StoreServlet,* prenderebbe in carico la parte front end e utilizzerebbe entrambi i bean precedenti. Il layout del modulo WAR sarebbe il seguente:

*WEB-INF/classes/com/example/cart/CartBean.class*

*WEB-INF/classes/com/example/web/StoreServlet*

*WEB-INF/lib/cc.jar*

*WEB-INF/ejb-jar.xml*

*WEB-INF/web.xml*

## 5.3 Packaging Web Archives

Un modulo web è l’unità più piccola delle risorse web che sia esportabile e utilizzabile in un’architettura Java EE.

Un modulo web contiene i componenti web e files statici di contenuto web, come le immagini, che soono definiti risorse web (web resources).

Un modulo web Java EE corrisponde a un’applicazione Java EE come viene definita nella specifica di Java Servlet.

In aggiunta ai componenti e alle risorse web un modulo web può contenere altri file quali:

* Classi di utilità lato server, ad esempio carrelli della spesa
* Classi lato client, come le classi di utilità.

Un modulo web ha una struttura specifica: le directory di alto livello del modulo web costituiscono la **document root** dell’applicazione.

La document root è dove pagine XHTML, le classi e gli archivi lato-client e le risorse web statiche, come le immagini, sono immagazzinate.

La document root contiene una sotto-directory chiamata WEB-INF che può contenere i seguenti files e directory:

* classes: una directory che contiene classi lato-server, ovvero servlets, file di classi enterprise beans, classi utility e componenti JavaBeans.
* lib: una directory che contienei file JAR che a loro volta contengono enterprise beans, archivi JAR e librerie chiamate dalle classi lato-server.
* Deployment descriptors: ad esempio web.xml e ejb-jar.xml.

Un modulo web necessita del file web.xml se utilizza la tecnologia dell JavaServer Faces, se deve specificare certi tipi di informazioni di sicurezza o se vuole effettuare l’override di informazioni specificate in annotazioni sui componenti web.

Si possono inoltre creare sotto-directories specifiche per l’applicazione sia nel document root che nel *WEB-INF/classes.*

Si può fornire un **runtime deployment descriptor (DD)** quando si esporta un WAR su GlassFish Server, ma non è quasi mai richiesto.

Il runtime DD è un file XML che può contenere informazioni quali la context root di un’applicazione web, il mapping dei nomi delle risorse di un’applicazione per le risorse di un server GlassFish e il mapping delle regole di sicurezza dell’applicazione definite nel Server GlassFish.

Il GlassFish Server web application runtime DD, se utilizzato, è chiamato *glassfish-web.xml* e è posto nella directory *WEB-INF.*

La figura sottostante indica la struttura di un modulo web che può essere importato su Server GlassFish:



## 5.4 Packaging Resource Adapter Archives

Un file Resource Adapter Archives (RAR) raggruppa file XML, classi Java e altri oggetti per applicazioni con architettura Java EE Connector (JCA).

Un resource adapter può essere importato su qualunque server EE come un’applicazione Java EE.

Un file RAR può essere inserito in un file EAR o essere in un file separato.

Il file RAR contiene:

* Un file JAR con le classi implementate del resource adapter
* Una directory META-INF/ facoltativa che può conservare un file *ra.xml* e/o un deployment descriptor specifico per il server dell’applicazione usato per scopi di configurazione.

Un file RAR può essere importato sul server dell’applicazione come un componente standalone o come parte di un’applicazione più grossa. In entrambi i casi l’adattatore è disponibile per tutte le applicazioni usando una procedura di ricerca.