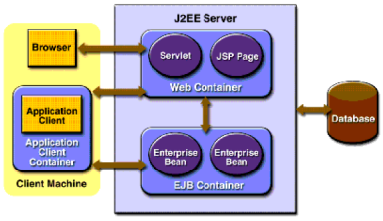
Lektion 2

DD2483
Enterprise Java
6 hp

Java EE



Innehåll

- Java EE konceptuell översikt
- Tomcat / OpenEJB(*)
- Konfigurering av Tomcat / OpenEJB(*)
  - web-moduler
  - ejb-moduler(*)
  - applikationer

Java EE 5 subAPI

- Viktigaste:
  - Servlet API 2.5
  - Java Server Pages (JSP 2.1) (del av Servlet API)
  - Enterprise Java Beans (EJB 3.0)
  - Java Server Faces (JSF 1.2)
  - Java Server Pages Standard Tag Library (JSTL 1.2)
  - JavaMail 1.4 (ingår ej i kursen)
  - Ett flertal till...
- Senaste version EE 6.

Java EE

- Ett samlingsnamn för ett flertal olika subAPI:n med gemensam nämnare att de lämpar sig för **Enterprise applikation**er där en sådan kännetecknas av:
  - hög belastning (många samtidiga klienter)
  - "komponentbaserat" vilket innebär att en applikation är uppdelad i flera fristående moduler som kommunicerar med varandra. Dessa logiska skikt underlättar underhåll.
  - klustringsmöjligheter

Java EE

- Enterprise applikationer använder sig också frekvent av följande från Java SE:
  - Remote Method Invokation (RMI)
  - Java Naming and Directory Interface (JNDI)
  - Java Database Connectivity (JDBC)

Servlet / JSP

- En (HTTP)Servlet är en klass dedikerad åt att hantera (HTTP)Request:s, behandla dessa och sedan generera (HTTP)response:s
- JSP-kod anges i filer med ändelsen .jsp
- .jsp-filer är .(x)html-filer med inbäddad javakod
- .jsp-filer översätts först till Servlet:ar (som sedan kopileras och exekveras)

Apache Tomcat 6.0.26

- En applikationsserver är den middleware motor som man kör sina Java EE-applikationer på.
- Tomcat är endast en servletcontainer (med webserver) och inte en full Java EE-server.
- En webserver ingår alltid i produkten men kan även agera middleware mot klienter genom andra protokoll än http. Detta tas dock inte upp i denna kurs.

JavaBean

- En vanlig Javaklass med följande egenskaper
  - publik konstruktor utan argument
  - set:er och get:er metoder för samtliga instansvariabler
  - POJO (Plain Old Java Object)

Installation

- Tomcat finns i labkatalogen på kurshemsidan som en .zip-fil, spara denna i roten på er hemkatalog, skriv sedan i en unixterminal:
  - ”unzip apache-tomcat-6.0.26.zip”
  - ”mv apache-tomcat-6.0.26 tomcat”
  - ”chmod –R 700 tomcat/”
  - ”cd tomcat/bin/”
  - ”sh startup.sh”
  - http://localhost:8080
  - ”sh shutdown.sh”
- Döp gärna om katalogen till endast ”tomcat”

Enterprise JavaBeans (EJB)

- Erbjuder databeständighet (persistens) och distribution av objekt.
- Persistensen uppnås genom att EJB:n kan speglas i en databas, d v s:
  - skapa objekt => SQL-INSERT
  - ändraobjekt => SQL-UPDATE
  - radera objekt => SQL-DELETE
- Distributionen uppnås genom att EJB:n kan anropas via RMI i t e x ett serverkluster.
- EJB != JavaBean !!!

Konfigurering

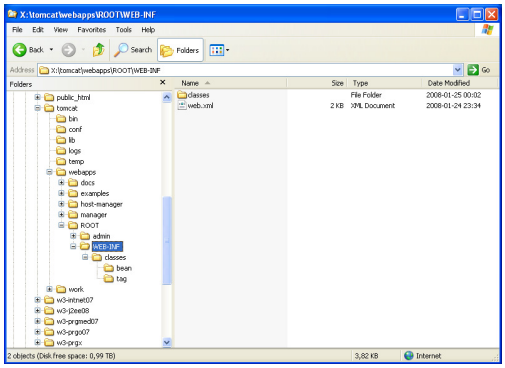
- I appservrar används xml för konfigurering där en del xml-filer är Java EE-specifika och andra är applikationsserverspecifika. Vi ska främst beröra de EE-specifika.

HTTP-port

- Om ni tänkt att köra mot nada:s fjärrinloggningsserver (my.nada.kth.se) måste ni tänka på att köra mot en annan port än 8080, annars får ni portkonflikt med andra studenter.
- Editera tomcat/conf/server.xml och byt ut de två förekomsterna av 8080 mot ett femsiffrigt slumptal, dock max 65535. Ersätt 8080 i denna guide med detta tal genomgående.

WEB-moduler

- Körs under servletcontainern (=tomcat)
- En webmodul består av
  - .jsp-filer,.(x)html-filer,.css-filer
  - .class-filer
    - Servlets
    - JavaBeans
    - Tagklasser
  - .jar (importerade klasser under /lib/)
  - Konfigurationsfilen web.xml

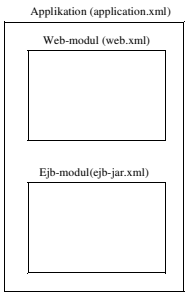


EJB-moduler

- Körs under EJB-containern (=openEJB)
- En ejb-modul består av
  - .class-filer
    - Själva EJB:n
  - Konfigurationsfilen ejb-jar.xml

Konfigurationsfiler

- En applikation kan bestå av web-moduler och ejb-moduler och dessa har varsin Java EE-specifika konfigurationsfil.



Enterprise applikationer

- Ingen, en eller flera web-moduler samt ingen, en eller flera ejb-moduler definierar en enterprise applikation.
- Något av ovanstående måste givetvis ingå!!!

Filtyper för arkiv

- .jar (Java ARchive) = zip med .class-filer
- .war (Web ARchive) = zip av en web-modul
- .ear (Enterprise ARchive) = zip av .war + ejb-modul
- Fördelen med denna hantering är att man får en paketering av applikationen som man kan ”droppa” på en annan applikationsserver. Detta kallas att driftsätta applikationen (”deploy”).
- Det finns ett populärt verktyg för att hantera denna paketering som heter ”Ant”

JDBC

- Applikationsservern kan ansluta mot samtliga databaser som det finns en JDBC-drivrutin till
- Denna måste laddas ned, följer ej med Tomcat
  - http://www.mysql.com/
  - http://www.mysql.com/downloads/connector/j/5.1.html
  - Plocka ur zip-filen ut
    - mysql-connector-java-5.1.12-bin.jar
  - och placera den sedan under tomcat/lib/

Kompileringsfil

- UNIX: Skapa en fil ”compile.sh” under ”tomcat/bin” som innehåller en enda lång javac-rad som samkompilerar alla .java-filer i hela er applikation.
  - #!/bin/sh
  - set tomcatpath=$HOME/tomcat/lib
  - set webapppath=$HOME/tomcat/webapps/ROOT/WEB-INF/classes
  - javac -cp $tomcatpath/servlet-api.jar $webapppath/*.java $webapppath/bean/*.java
- Windows: Skapa en fil ”compile.bat” under ”tomcat/bin” som innehåller en enda lång javac-rad som samkompilerar alla .java-filer i hela er applikation.
  - set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_04
  - set tomcatpath=X:\tomcat\lib
  - set webapppath=X:\tomcat\webapps\ROOT\WEB-INF\classes
  - javac -cp %tomcatpath%\servlet-api.jar %webapppath%*.java %webapppath%\bean*.java
- Observera att ni måste byta ut sökvägarna ovan till motsvarande på er egen dator

JDBC (context.xml)

- Tomcat-specifik konfigurationsfil.
- <Resource
- name="jdbc/db"
- auth="Container"
- type="javax.sql.DataSource"
- username=“root"
- password=“*****"
- driverClassName="com.mysql.jdbc.Driver"
- url="jdbc:mysql://localhost:3306/test "
- maxActive="8"
- maxIdle="4"/>

/lib/-kataloger

- Används för att importera .jar-filer, ofta jdbc-drivrutiner eller t e x JFreeChart för att få tillgång till ett grafritningsAPI.
- Dessa finns på ett flertal platser i filträder och var de placeras är viktigt. De vanligaste är tillhörande:
  - Servern som helhet, d v s gäller samtliga applikationer som körs på servern.
  - en applikation => tillgänglig i hela applikationen
  - en web-modul => tillgänglig i i web-modulen
  - ejb-modulen => tillgänglig i ejb-modulen

JDBC (web.xml)

- <resource-ref>
- <res-ref-name>
- jdbc/db
- </res-ref-name>
- <res-type>
- javax.sql.DataSource
- </res-type>
- <res-auth>
- Container
- </res-auth>
- </resource-ref>

Första exemplet

```
import java.io.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request,HttpServletResponse
     response)
     throws IOException{
     PrintWriter out = response.getWriter();
     out.println("Hello, world!");
     out.close();
    }
}
```

web.xml

- <servlet>
- <servlet-name>smurf</servlet-name>
- <servlet-class>HelloWorld</servlet-class>
- </servlet>
- <servlet-mapping>
- <servlet-name>smurf</servlet-name>
- <url-pattern>/HelloWorld</url-pattern>
- </servlet-mapping>

Sammanfattning

- Under laborerandets gång:
  1. Se till att du använder Java SE 6.
  2. csh compile.sh (ersätter javac)
  3. sh startup.sh (ersätter java)
  4. http://localhost:8080 (läs felmeddeande)
  5. sh shutdown.sh
  6. => 2
- Ni behöver inte starta om servern när ni ändrat .jsp-filer, de kompileras ”live”
- OBS!!! Glöm ej punkt (5) innan ni loggar ut, annars ligger en javaprocess kvar och blockerar port 8080 för nästa grupp.