Föreläsning 4

DD1389
Internetprogrammering
6 hp

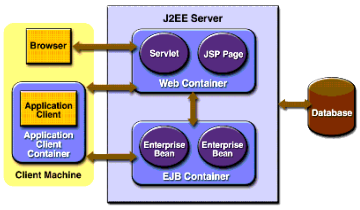
Innehåll

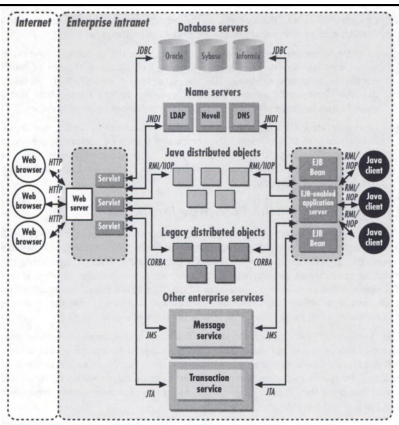
- Java EE konceptuell översikt
- Finns ett flertal implementationer
- Konfigurering av Tomcat
  - web-moduler
  - ejb-moduler
  - applikationer

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





Ett Scenario

1. kund besöker en webbsidan www.JustWhatYouNeed.com, får möjligheten att se och välja alla produkter
2. Webbservern använder kundvagns-servlet för att hålla reda de produkter som kunden väljer.
   HTTP är tillståndslös men servlets kan kvarstå mellan kundens interaktion, så servlet kan minnas kundens val.
3. Kassa-servlet mha JDBC hämtar information om valda produkter som lagrades av kundvagns-servleten i en databas
4. Kassa-servlet begär av kunden adress-uppgifter och svaren lagras i en kund-databas

Ett Scenario

5. Servletten skickar adressinformationen till en debiteringsserver. Debiteringsservern är gammal och har inte ett standard gränssnitt men lyckligtvis kan den exportera sig som ett CORBA objekt. När servletten tar emot CORBA remote objekt kan den anropa metoder avsett för debitering.
6. Företaget JustWhatYouNeed håller på att växa hela tiden med nya lager och butiker, för att företaget ska ge kunden snabbaste service använder kassa-servletten JNDI för att hitta närmaste lager/butik till kunden.
7. När kassa-sevletten har information om närmatse butik kan slutföra köpet genom att använda JMS för att skicka ett meddelande till butiken om köpet.

Java EE 7 subAPI

- Viktigaste:
  - **Servlet API 3.1**
  - **Java Server Pages (JSP 2.3) (del av Servlet API)**
  - **JDBC (JAVA DATABASE CONNECTIVITY)**
  - JNDI(JAVA NAMING AND DIRECTORY INTEFACE)
  - RMI (REMOTE METHOD INVOCATION)
  - JTA (JAVA TRANSACTION API)
  - JFS (Java Server Faces)
  - JSTL (Java Server Pages Standard Tag Library 1.2)
  - EJB(Enterprise Java Beans 3.2)
  - Java Persistence API (JPA 2.1)
  - Ett flertal till...
- Senaste version EE 8

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)

JavaBean

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

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 !!!

Apache Tomcat 8

- En applikationsserver är den middleware motor som man kör sina Java EE-applikationer på.
- Tomcat är endast en servlet-container (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.

Installation

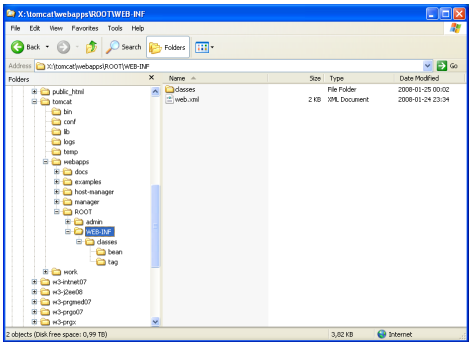
- Tomcat finns i labkatalogen på kurshemsidan som en .zip-fil, spara denna i roten på er hemkatalog, skriv sedan i en unixterminal:
  - ”unzip apache-tomcat-8.0.30.zip”
  - ”mv apache-tomcat-8.0.30 tomcat”
  - ”chmod –R 700 tomcat/”
  - ”cd tomcat/bin/”
  - ”./startup.sh”
  - http://localhost:1234
  - ”./shutdown.sh”
- Döp gärna om katalogen till endast ”tomcat”

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.

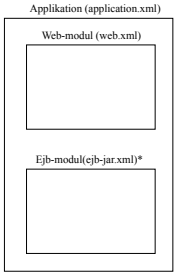
HTTP-port

- Om ni tänkt att köra mot csc:s fjärrinloggningsserver (u-shell.csc.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 något annat (max 65535).



Konfigurationsfiler

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



WEB-moduler

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

EJB-moduler

- Körs under EJB-containern
- En ejb-modul består av
  - .class-filer
    - Själva EJB:n
  - Konfigurationsfilen ejb-jar.xml
- **OBS! Tomcat saknar ejb-modul**

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”).

/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

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 7 (module add openjdk).
  2. ./startup.sh
  3. http://localhost:1234 (läs ev felmeddeande)
  4. ./shutdown.sh
- Ni behöver inte starta om servern när ni ändrat .jsp-filer, de kompileras ”live”
- OBS!!! Glöm ej punkt (4) innan ni loggar ut, annars ligger en javaprocess kvar och blockerar portar för nästa grupp.