1 História Web aplikácií

Vývoj www – World Wide Web prebiehal v dvoch líniách - dokumentový formát HTML a prenosový protokol HTTP. Základné míľniky boli :

- 60. roky 20. stor. Douglas Engelbert

vznik myšlienky previazaných dokumentov

- 1980 - Ted Nelson - Xanada

definované sieťové prostredie podobné súčasnému webu

prvýkrát použitý výraz hypertext

- 1980 - Charles Goldfarb – SGML

jazyk, ktorý bol priamy predchodca dnešného HTML

- 1989 - Tim Berners-Lee – prvý intranet

vznikol vo Švajčiarskom stredisku pre jadrový výzkum CERN

bol nazvaný World Wide Web

- 1992 – systém bol uvoľnený pre komerčné použitie

na svete existuje približne 50 serverov

prvé grafické prehliadače Midas a Viola

- 1994 - prehliadač Mosaic (neskôr Netscape Navigator) - obrázky, farby , animácie

1.1 Statický versus dymanický obsah

Pôvodné technológie boli zamerané na poskytovanie statického, hypertextovo previazaného obsahu.Dynamické zmeny obsahu boli vykonávané na pozadí.

1.2 CGI-BIN

Webový server pri obdržaní požiadavky na dynamický obsah spustí príslušný proces (program) a jeho výstup presmeruje na klienta. Podľa špecifikácie CGI-BIN nastaví pred spustením procesu premenné prostredia.

1.3 Server API

Server preposiela požiadavky na prilinkované knižnice-objekty. Rozhrania nie sú štandardizované, ale sú proprietárne, definované výrobcom webového serveru. Príklad ISAPI filter, NSAPI rozhranie.
Tieto rozhrania poskytujú aj ďalšie služby pre komponenty, napríklad udržiavanie session, zdieľanie dát medzi požiadavkami a pod.

1.4 Skriptovacie Jazyky

Pretože CGI-BIN a server API vyžadujú programovanie v „programovacích“ jazykoch, je program na generovanie HTML obsahu zbytočne zložitý. Skriptovacie jazyky rozširujú HTML o značky ktoré sú vykonávané na strane servera a generujú dynamický obsah.
Najpoužívanejšie technológie súčasnosti sú :

- PHP – Personal Home Page
- ASP (ASP.NET)
- JSP – Java Server Pages
- Python, Rubby, Perl a ďalšie

2 Http protokol

Jednoduchý bez stavový protokol. Prebieha formou otázka odpoveď (request/response) nad soketovým spojením. (HTTP 1.1)

2.1 Request

```
Request-Line
*(( general-header  | request-header  | entity-header ) CRLF)
CRLF
[ message-body ]

Request-Line   = Method SP Request-URI SP HTTP-Version CRLF
Request-URI    = "*" | absoluteURI | abs_path | authority
```

Cvičenie : Spustite si web server a pomocou telnetu skúste zaslať správny request na server. Použite TCP/IP monitor.

```
telnet localhost 8080
GET / http/1.1 [cr] [cr]
```

Metódy GET, POST …

GET – požiadavka na získanie dát (napr. HTML dokumentu)
POST – zasiela dáta na resource špecifikovaný v URI (typicky dáta z formuláru)
PUT – požiadavka na zmenu resource v URI
Ďalšie metódy ….

2.2 Response

```
Response      = Status-Line
*(( general-header | response-header  | entity-header ) CRLF)
CRLF
[ message-body ]
```

2.3 Session handling

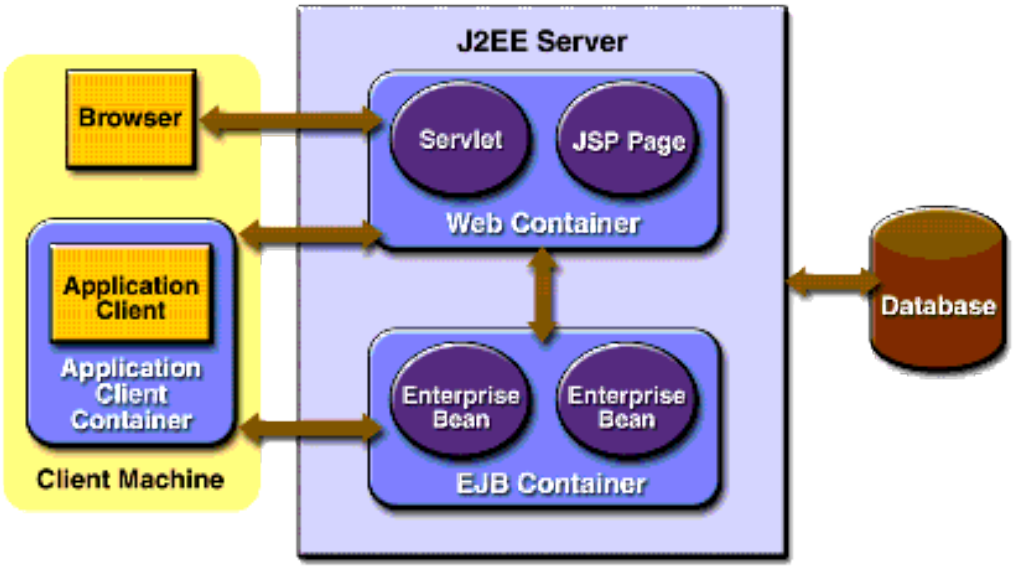
Session je sekvencia súviciach požiadaviek-odpovedí ktoré predstavujú ucelenú komunikáciu jedného klienta so serverom. Pretože http protokol je bez stavový, neobsahuje tento pojem a musí byť simulovaný inými prostriedkami.
URL rewriting - do všetkých hypertextových odkazov v rámci jedno servera-applikácie sa vloží identifikátor session., server poskytuje službu vkladania a výberu tejto informácie.
Cookies – identifikátor session je uložený v dočasných Cookies

3 Java web kontajner

3.1 Čo je to web kontajner

Architektúra J2EE – sada rozhraní (API) a knižníc na vytváranie „enterprise“ applikácií.

Jednotlivé komponenty sú spúšťané v kontajneroch, kontajner sa stará o ich životný cyklus a poskytuje im služby.



Web Services Technologies
Implementing Enterprise Web Services (JSR 109)
Java API for XML-Based Web Services (JAX-WS) 2.0 (JSR 224)
Java API for XML-Based RPC (JAX-RPC) 1.1 (JSR 101)
Java Architecture for XML Binding (JAXB) 2.0 (JSR 222)
SOAP with Attachments API for Java (SAAJ) (JSR 67)
Streaming API for XML (JSR 173)
Web Service Metadata for the Java Platform (JSR 181)

Web Application Technologies
Java Servlet 2.5 (JSR 154)
JavaServer Faces 1.2 (JSR 252)
JavaServer Pages 2.1 (JSR 245)
JavaServer Pages Standard Tag Library (JSR 52)

Enterprise Application Technologies
Enterprise JavaBeans 3.0 (JSR 220)
J2EE Connector Architecture 1.5 (JSR 112)
Common Annotations for the Java Platform (JSR 250)
Java Message Service API (JSR 914)
Java Persistence API (JSR 220)
Java Transaction API (JTA) (JSR 907)
JavaBeans Activation Framework (JAF) 1.1 (JSR 925)
JavaMail (JSR 919)

Management and Security Technologies
J2EE Application Deployment (JSR 88)
J2EE Management (JSR 77)
Java Authorization Contract for Containers (JSR 115)

Aplikačný server je applikácia, ktorá implementuje jeden alebo niekoľko kontajner a poskytuje služby definované v J2EE.

Jeden z najpoužívanješích web kontajnerov je Apache Tomcat .

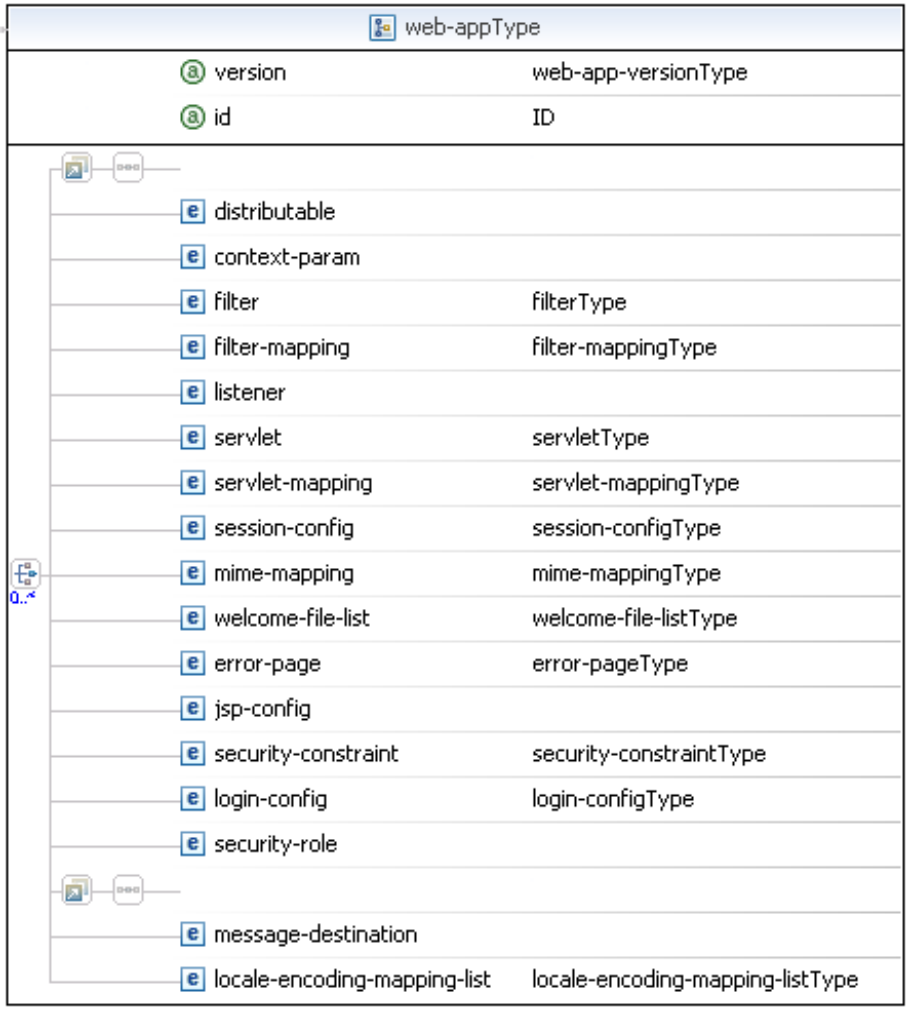
3.2 Web aplikácia

Web aplikácia je sada súviciacich komponentov distribuovaná ako jeden celok. Web applikácia (tiež content) sa nasadzuje ako jeden adresár alebo obsah adresára zabalený (zip) do súboru .war . Štruktúra tohto adresára je daná:

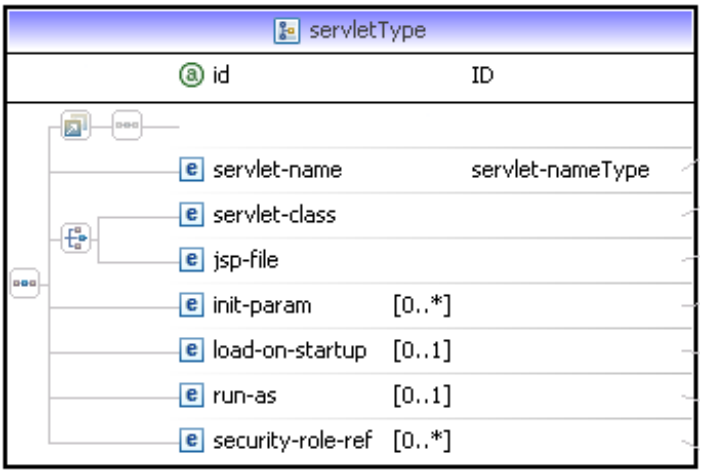
```
<content>
     [WEB-INF]
          [classes]
          [lib]
               *.jar
          [tags]
               *.tag
          web.xml
          [*]
          *.*
     [*]
     *.*
```

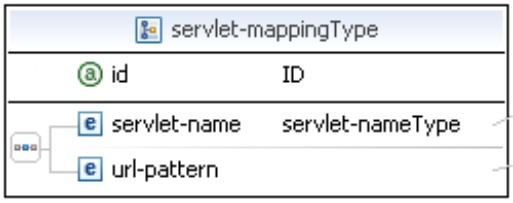
3.3 Deployment deskriptor

Deployment descriptor web.xml je konfiguračný súbor, ktorý popisuje nasadzované komponenty a web aplikáciu. Root element deployment descriptora je web-app:



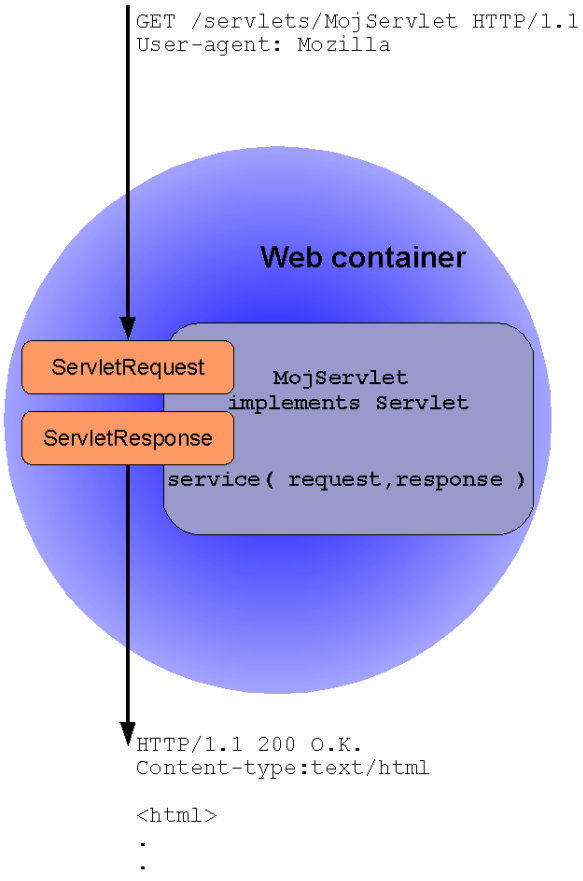
Najdôležitejším komponentom je servlet a jeho mapovanie na URI – servlet-mapping :





4 Rozhranie javax.servlet.Servlet

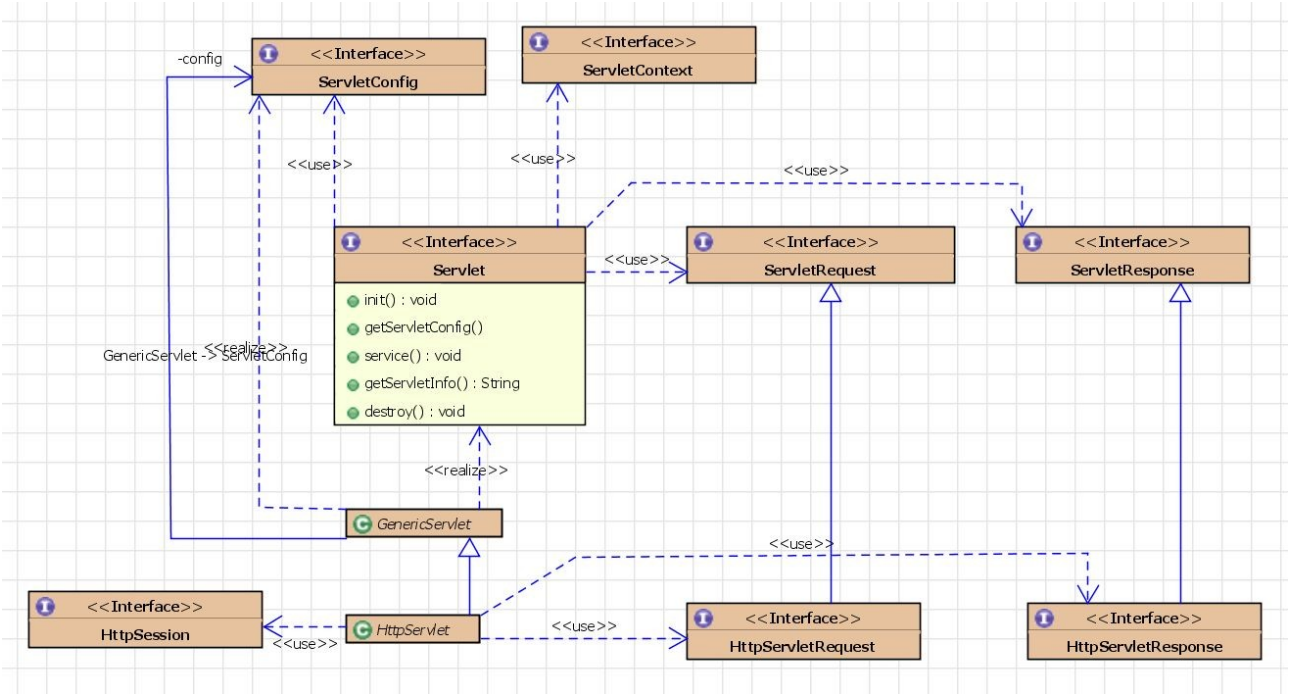
- Kontainer pripraví objekty request a response
- Nájde zodpovedajúci servlet
- Zavolá metódu service()
- odošle response



4.1 Životný cyklus servletu

- Vytvorenie inštancie
- init(ServletConfig)
- (0-*) service()
- destroy()
- finalize()

4.2 Ďalšie triedy servletov



Trieda HttpServlet robí:

- spracovanie inicializačných parametrov
- vetvenie z metódy service na metódy doGet, doPost, doPut …
- obaľuje ServletRequest do HttpServletRequest čím získa prístup na HttpSession

Trieda ServletContext

- predstavuje našu web aplikáciu
- umožňuje prístup ku zdrojom aplikácie
- umožňuje zdielanie objektov v rámci aplikácie

5 Rozhranie javax.servlet.Filter

Definuje sa podobne ako servlet v deployment deskriptore.
Zadáva sa meno filtra, trieda filtra a mapovanie filtrov.
Kontajner nájde všetky filtre zodpovedajúce URI adrese a vytvorí z nich reťaz (FilterChain) na konci ktorej servlet.

```
public void doFilter(
     ServletRequest request,
     ServletResponse response,
     FilterChain chain) throws IOException, ServletException {
// Urob daco predtym
     chain.doFilter(request,response);
// aj potom
}
```

6 Listenery

Komponenty, ktoré môžu reagovať na udalosti vo web aplikácii. Až na dve výnimky sa definujú v deployment deskriptory.

ServletContextListener

void contextDestroyed(ServletContextEvent sce)

void contextInitialized(ServletContextEvent sce)

ServletRequestListener

void requestDestroyed(ServletRequestEvent rre)

void requestInitialized(ServletRequestEvent rre)

HttpSessionListener

void sessionCreated(HttpSessionEvent se)

void sessionDestroyed(HttpSessionEvent se)

ServletContextAttributeListener

void attributeAdded(ServletContextAttributeEvent scab)

void attributeRemoved(ServletContextAttributeEvent scab)

void attributeReplaced(ServletContextAttributeEvent scab)

ServletRequestAttributeListener

void attributeAdded(ServletRequestAttributeEvent srae)

void attributeRemoved(ServletRequestAttributeEvent srae)

void attributeReplaced(ServletRequestAttributeEvent srae)

HttpSessionAttributeListener

void attributeAdded(HttpSessionBindingEvent se)

void attributeRemoved(HttpSessionBindingEvent se)

void attributeReplaced(HttpSessionBindingEvent se)

Nasledujúce listenery sa nedeklarujú v deployment deskriptore, viažu sa automaticky.

HttpSessionBindingListener

HttpSessionActivationListener

7 Distribuovateľná session

Web aplikácia môže byť vytvorená ako distribuovateľná. Web kontajner tak môže bežať na viacererých strojoch a session objekty môžu medzi nimi migrovať. Kontajner notifikuje objekty o ich presune cez interfejs

HttpSessionActivationListener

void sessionDidActivate(HttpSessionEvent se)

void sessionWillPassivate(HttpSessionEvent se)