




Tehnologii Java

Curs -

Cristian Frăsinaru

`acf@infoiasi.ro`

Facultatea de Informatică
Universitatea "Al. I. Cuza" Iași





Structura aplicațiilor Web



Cuprins

- Structura unei aplicații Web
- Tipuri de componente
- Fișierul *web.xml*
- Ciclul de viață al aplicațiilor Web
- Tratarea evenimentelor

Structura unei aplicații Web

A "web application" is a collection of servlets and content installed under a specific subset of the server's URL namespace such as */aplicatie* and possibly installed via a .war file.

- Pagini web statice, dinamice
- Resurse (imagini, css, etc.)
- Fisiere de configurare
- Clase
- Librării

Componente Web



- HTML, XHTML, CSS, JavaScript, JSON, etc.
- Servlet
- JSP
- Tag Handler, Tag Library Descriptor, Tag File
- Web Application Listener
- Service Locator



Organizarea componentelor

- La nivel de surse: *Java EE blueprints*
- La nivel executabil → *war* (Web Archive)

```
\aplicatie
```

```
    Pagini Web si resurse: .html, .jsp, imagini, etc.
```

```
    \WEB-INF
```

```
        web.xml
```

```
        \classes
```

```
            .class, .properties
```

```
        \lib
```

```
            .jar
```

```
        \tags
```

```
            .tag
```

web.xml



Este fișierul principal de configurare al oricărei aplicații Web. Conține următoarele secțiuni:

- General
- Servlets
- Filters
- Pages
- References
- Security



Exemplu de fișier web.xml

```
<web-app>
  <display-name>aplicatie</display-name>
  <description>Prima mea aplicatie</description>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>

  <error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/error.jsp</location>
  </error-page>

  <servlet>
    <servlet-name>FirstServlet</servlet-name>
    <servlet-class>com.samples.FirstServlet</servlet-class>
    <load-on-startup> 1 </load-on-startup>
  </servlet>
</web-app>
```

Contextul unei aplicații

ServletContext Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.

There is one context per "web application" per Java Virtual Machine.

The ServletContext object is contained within the ServletConfig object, which the Web server provides the servlet when the servlet is initialized.

```
<web-app>
  ...
  <context-param>
    <param-name>    someParam </param-name>
    <param-value>   someValue </param-value>
  </context-param>
  ...
</web-app>
```

Ciclul de viață al unei aplicații Web

Avem nevoie de o modalitate de a fi informați de evenimente ce țin de ciclul de viață al unei aplicații Web, evenimente generate de containerul aplicației.

- Când o aplicație este inițializată
- Când o aplicație este eliminată din container
- Când este primită o cerere
- Când este creată o sesiune de lucru
- etc.

Interfețe de tip *Listener*

- ServletContextListener
- ServletContextAttributeListener
- ServletRequestListener
- ServletRequestAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener

ServletContextListener

```
public class ApplicationWatch implements ServletContextListener {

    private static long startupTime = 0L;

    /* Application Startup Event */
    public void contextInitialized(ServletContextEvent ce) {
        startupTime = System.currentTimeMillis();
    }

    /* Application Shutdown Event */
    public void contextDestroyed(ServletContextEvent ce) {}

    public static Date getStartupTime() {
        return startupTime;
    }
}
```

HttpSessionListener

```
public class SessionCounter implements HttpSessionListener {

    private static int users = 0;

    /* Session Creation Event */
    public void sessionCreated(HttpSessionEvent httpSessionEvent) {
        users ++;
    }

    /* Session Invalidation Event */
    public void sessionDestroyed(HttpSessionEvent httpSessionEvent) {
        users --;
    }

    public static int getConcurrentUsers() {
        return users;
    }
}
```

Inregistrarea în web.xml

```
<web-app>
  ...
  <listener>
    <listener-class>
      util.listeners.ApplicationWatch
    </listener-class>

    <listener-class>
      util.listeners.SessionCounter
    </listener-class>
  </listener>
  ...
</web-app>
```

sau adnotarea clasei cu: @WebListener()

Exemplu de utilizare

```
public class InfoServlet extends HttpServlet {  
  
    protected void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
  
        HttpSession session = request.getSession();  
  
        response.setContentType("text/html");  
        PrintWriter out = new PrintWriter(response.getWriter());  
        out.println("<html>");  
        out.println("<h1>Aplicatia initializata la data: " +  
            new Date(ApplicationWatch.getStartupTime()));  
        out.println("<h1>Sesiuni active: " +  
            SessionCounter.getConcurrentUsers());  
        out.println("</html>");  
        out.close();  
    }  
}
```