J2EE

Java 2 Enterprise Edition

The Apache Jakarta Project
http://jakarta.apache.org/

DAW
06/07

Índice

- Introducción
- Servlets
- Tomcat
- Eclipse
- Filtros
- JSPs
- TagLibs
- JDBC
- ANT
- EasyEclipse

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Introducción (1/5)

- Diferentes plataformas de Java:
  - **J2SE (Java 2 Standard Edition)**
    - Desarrollo de aplicaciones y applets
  - **J2ME (Java 2 Micro Edition)**
    - Desarrollo de aplicaciones para micro-dispositivos (pantalla y memoria limitados): PDAs y teléfonos móviles
  - **J2EE (Java 2 Plataform Enterprise Edition)**
    - Aplicaciones de servidor
    - Con el tiempo, algunas APIs de J2EE se van pasando a J2SE
- J2EE es un conjunto de especificaciones orientadas al desarrollo de aplicaciones empresariales

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Introducción (2/5)

- **Características** de las aplicaciones empresariales:
  - Acceso a base de datos
  - Transaccionalidad
  - Escalabilidad
  - Disponibilidad
  - Seguridad
  - Integración
- **Arquitectura de tres capas**:
  - Máquina cliente
  - Máquina servidor J2EE
  - Base de datos

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Introducción (3/5)

- Especificaciones principales:
  - **Servlets:** API para crear aplicaciones web dinámicas
  - **JSPs (Java Server Page):** extensión de Servlets para facilitar el desarrollo de aplicaciones web
  - **JDBC (Java Dabase Connectivity)**: API para acceder a base de datos (incluida en J2SE)
  - **JNDI (Java Naming and Directory Interface)**: API para acceder a servicios de directorios. Ej. LDAP (incluida en J2SE)
  - **JMS (Java Message Service)**: API para colas de mensajes asíncronas
  - **EJBs (Enterprise JavaBeans)**: API para la realización de componentes distribuidos
  - **JTA (Java Transaction API)**: API para realizar aplicaciones transaccionales distribuidas

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Introducción (4/5)

- **JavaMail**: API para realizar aplicaciones de email independientes de plataforma y protocolo
- **JAF (JavaBeans Activation Framework)**: API necesario para JavaMail para manejar mensajes MIME (Multipurpose Internet Mail Extension)
- **JAXP (Java API for XML Processing)**: API para parsear XML y transformaciones XSLT
- **JCA (Java Connector Architecture)**: API para conectar con otros sistemas de información. Ej. centro de mensajes cortos
- **JAAS (Java Authentication and Authorization Server)**: API que proporciona mecanismos de autenticación y autorización

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

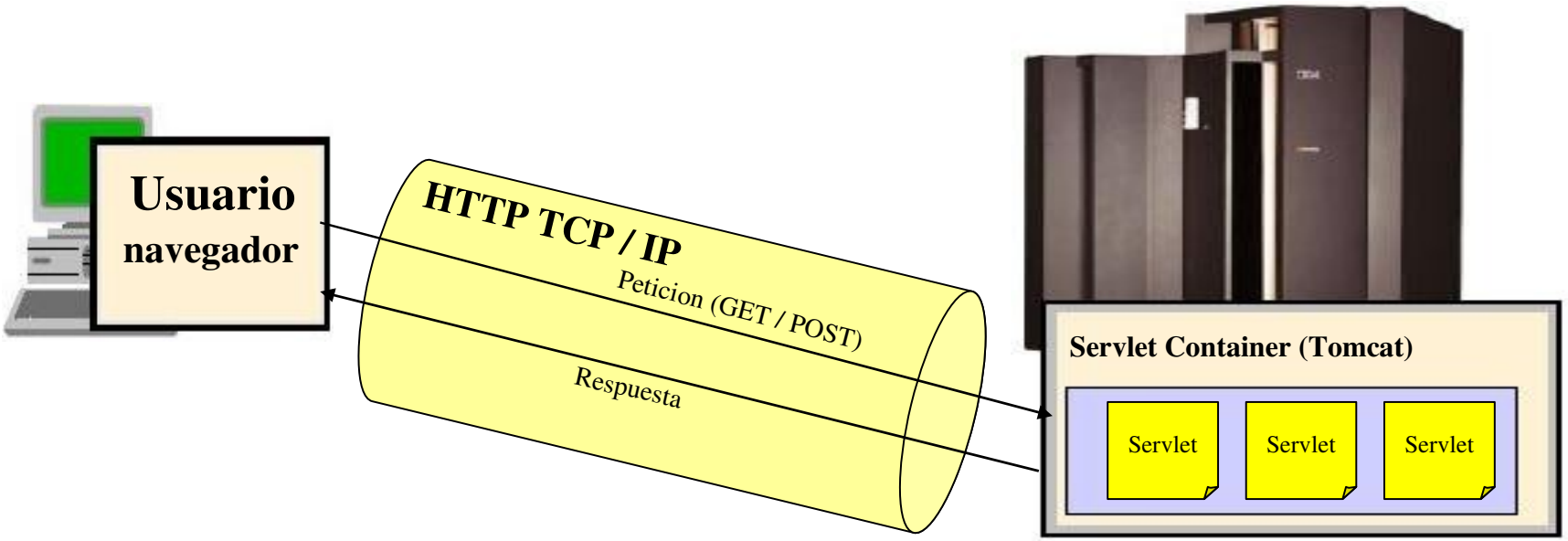
Introducción (5/5)

- J2EE no deja de ser un **CONJUNTO DE ESPECIFICACIONES.** Existen diferentes implementaciones según el fabricante:
  - **Libres**
    - **Tomcat**: Contenedor de Servlets y motor de JSPs
    - **JBoss**: Servidor de aplicaciones J2EE
    - **Geronimo**: Servidor de aplicaciones J2EE
  - **Comerciales**
    - **BEA WebLogic Server**
    - **IBM WebSphere**
    - **Oracle Application Server**

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Servlets. Introducción (1/4)

- API para la generación de contenido dinámico
- Servlet: clase Java que recibe peticiones (normalmente HTTP) y genera una respuesta (normalmente HTML o XML)
- Una aplicación web compuesta de Servlets se ejecuta en un servidor de aplicaciones web (contenedor)



Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Servlets. Ejemplo: “Hola Mundo Servlet” (2/4)

EJEMPLO

- Imprime la cadena “Hola Mundo desde un Servlet” en una página web
- Ficheros:
  - HolaMundoServlet.java
  - web.xml

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Servlets. Ejemplo: "Hola Mundo Servlet" (3/4)

HolaMundoServlet.java

```
package servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.*;

public class HolaMundoServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<body><h1>Hola Mundo desde un Servlet</h1></body>");
        out.println("</html>");
    }
}
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Servlets. Ejemplo: "Hola Mundo Servlet" (4/4)

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

  <display-name>Ejemplos</display-name>
  <description>
    Ejemplos de J2EE (DAW0607)
  </description>

  <servlet>
    <servlet-name>HolaMundoServlet</servlet-name>
    <servlet-class>servlets.HolaMundoServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HolaMundoServlet</servlet-name>
    <url-pattern>/HolaMundoServlet</url-pattern>
  </servlet-mapping>

</web-app>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Tomcat (1/3)

- Servidor de aplicaciones web o Contenedor de Servlets
- Tomcat: http://tomcat.apache.org
- Descargar: Tomcat 5.0.28 (jakarta-tomcat-5.0.28.tar.gz)
- Requisito: JDK (Java Development Kit)
- Descomprimir: tar zxvf jakarta-tomcat-5.0.28.tar.gz
- export JAVA_HOME="DIR_INSTALACION_JDK"
- Scripts de arranque y parada del servidor, dentro del directorio /bin de instalación:
  - Arrancar Tomcat:

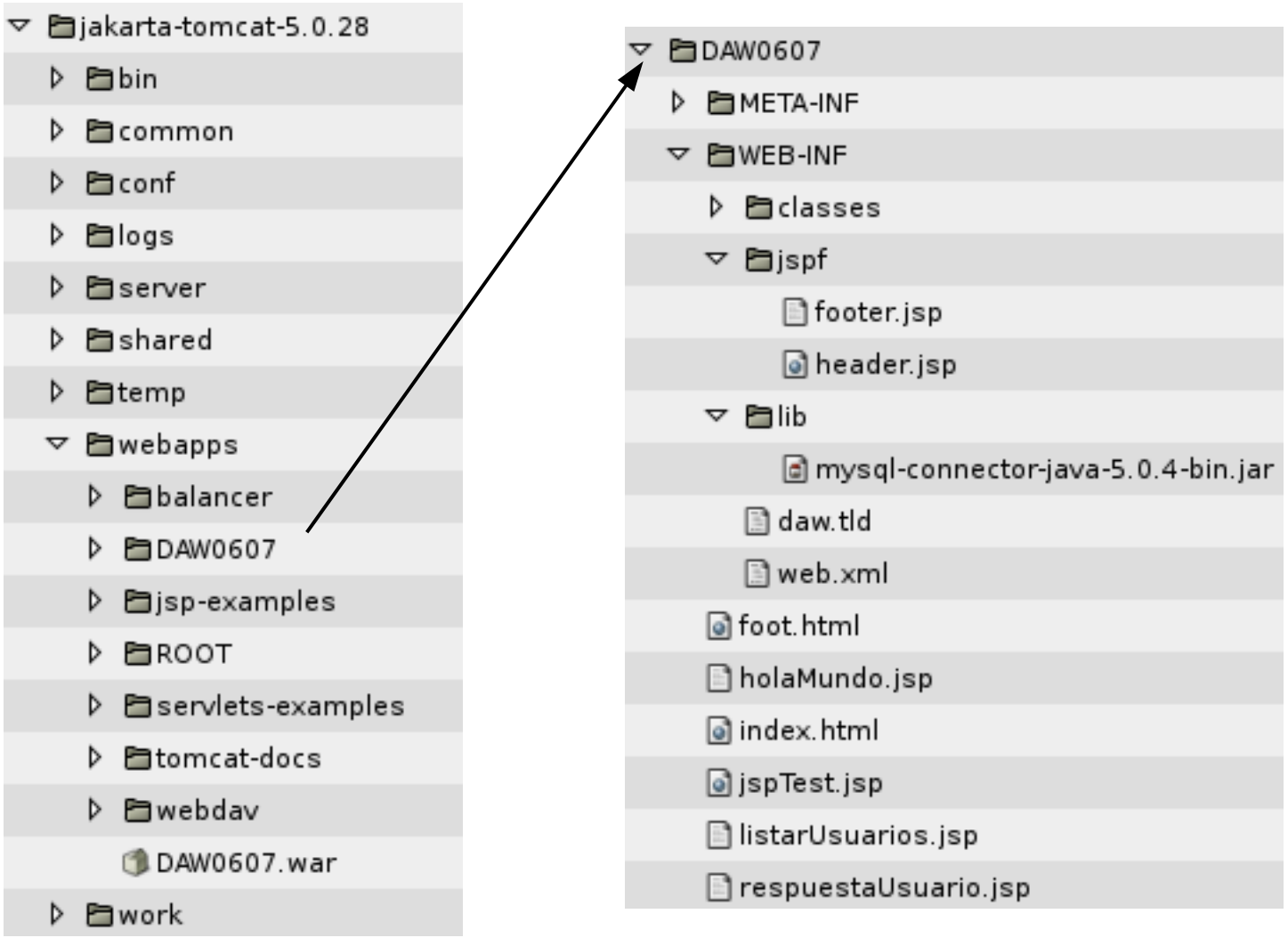
    ./startup.sh
  - Parar Tomcat:

    ./shutdown.sh

Servlet/JSP Spec	Apache Tomcat version
2.5/2.1	6.0.x
2.4/2.0	5.5.x
2.3/1.2	4.1.x
2.2/1.1	3.3.x

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Tomcat (2/3)



Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Tomcat (3/3)

- Pasos para **desplegar una aplicación web**:
  - Crear un archivo WAR (Ejemplo: DAW0607.war):
    - WEB_INF
    - WEB_INF/classes
    - WEB_INF/lib
    - WEB_INF/web.xml
  - Copiar dentro del directorio webapps
  - Reiniciar el Tomcat
  - Abrir un navegador: http://localhost:8080/DAW0607/MI_APLICACION
- **server.xml** (directorio conf): permite definir nuevos contextos

```
 <Context path="/daw0607" docBase="/home/rcedazo/workspace/DAW0607/web/"
reloadable="true">
 </Context>
```

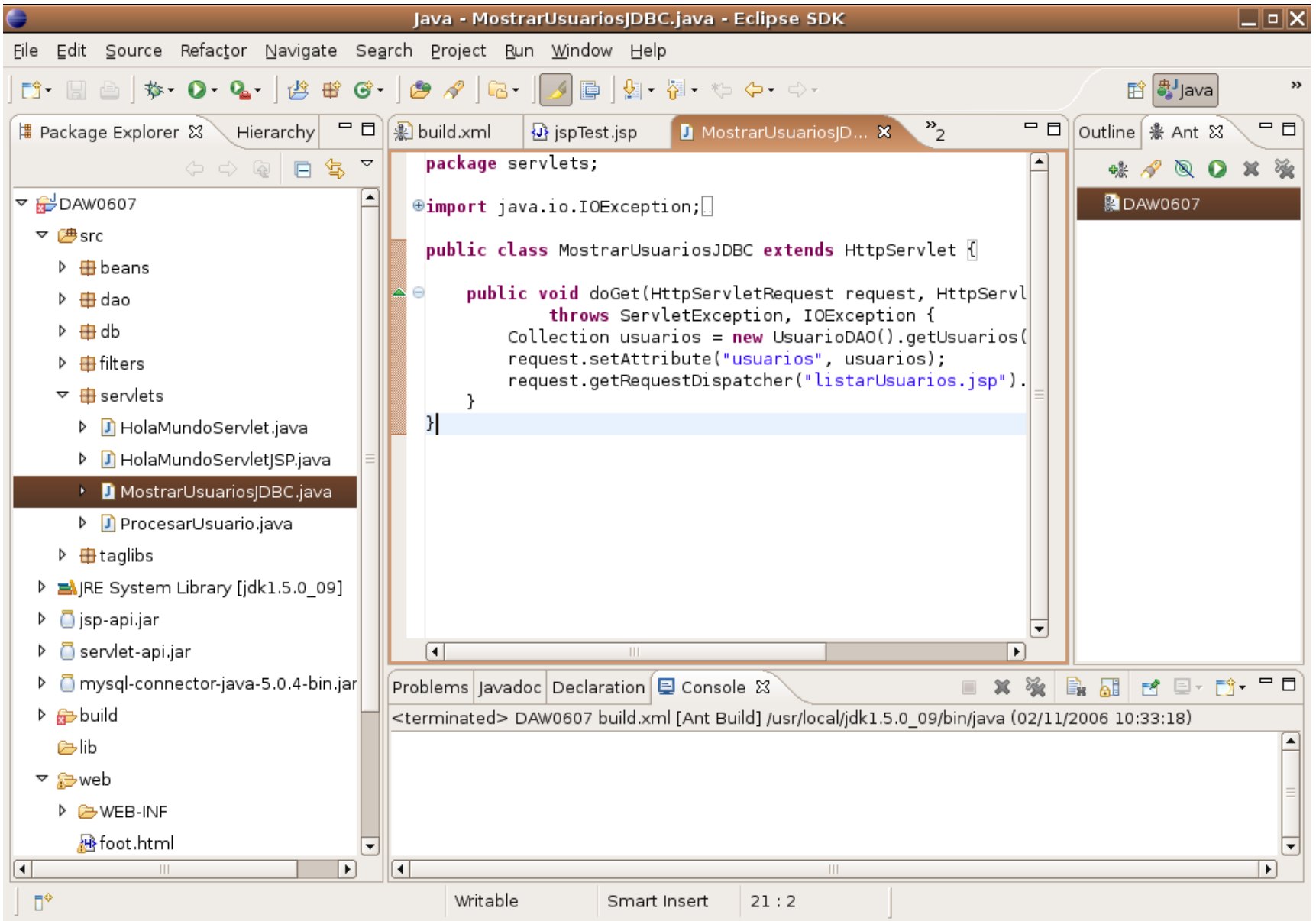

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Eclipse (1/2)

- IDE (Integrated Development Environment)
- Descargar: Eclipse SDK 3.2.1 http://www.eclipse.org
- Requisito: JRE (Java Runtime Environment)
- Eclipse plugins: http://eclipse-plugins.2y.net/

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Eclipse (2/2)



Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Filtros. Introducción (1/5)

- Acciones repetitivas: control de seguridad, herramienta de logs, sistema compresión de datos
- Solución tradicional: “include” de ficheros
- Filtro: mecanismo gestionado por el contenedor de Servlets que intercepta las peticiones a Servlets o JSPs y que actúa antes y después de pasar el control al Servlet
- Pre-procesadores y post-procesadores de Servlets
- Transparencia y reutilización de código
- Contenido dinámico y estático (HTML, imágenes)
- Métodos que impone la interfaz:
  - init(): inicialización del filtro
  - destroy(): destrucción del filtro
  - doFilter(request, response, filterChain)

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Filtros. Ejemplo: “Velocidad Filter” (2/5)

EJEMPLO

- Mide el tiempo de ejecución de los Servlets y lo imprime por la salida estándar
- Recibe un parámetro que indica la unidad en la que se quiere medir el tiempo. En este caso pueden ser “segundos” o “milisegundos”
- Ficheros:
  - VelocidadFilter.java
  - web.xml

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Filtros. Ejemplo: "Velocidad Filter" (3/5)

VelocidadFilter.java

```
package filters;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public final class VelocidadFilter implements Filter {

    private FilterConfig filterConfig = null;

    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;
    }

    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Filtros. Ejemplo: "Velocidad Filter" (4/5)

```
        long startTime = System.currentTimeMillis();

        chain.doFilter(request, response);

        long stopTime = System.currentTimeMillis();

        String unidad = filterConfig.getInitParameter("unidad");
        Double tiempo;
        if (unidad.equalsIgnoreCase("segundos")) {
            tiempo = new Double (((double)(stopTime - startTime))/1000);
        } else {
            tiempo = new Double (stopTime - startTime);
        }

        System.out.println("Tiempo de ejecución del servlet
            ["+((HttpServletRequest)request).getServletPath()+"]:
            "+tiempo+" "+unidad);
    }

    public void destroy() {
        this.filterConfig = null;
    }

}
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Filtros. Ejemplo: "Velocidad Filter" (5/5)

web.xml

```
<filter>
  <filter-name>VelocidadFilter</filter-name>
  <filter-class>filters.VelocidadFilter</filter-class>
  <init-param>
    <param-name>unidad</param-name>
    <param-value>segundos</param-value>
    <description>Unidad en la que se mide el tiempo (milisegundos o
      segundos)</description>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>VelocidadFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JSPs. Introducción (1/7)

- Java Server Page: Servlet orientado a generar el texto de la interfaz gráfica
- Ventajas:
  - Uso de herramientas de diseño de páginas web directamente
  - Actualizaciones del aspecto gráfico no provocan el re-arranque del servidor
- Ciclo de vida de un JSP:
  - Primera vez: a partir del JSP el servidor de aplicaciones crea un Servlet, lo compila y lo carga en memoria.
  - En caso contrario: le pasa la petición al Servlet ya compilado.
  - Si se modifica el JSP desde la última compilación: se genera un nuevo Servlet, lo compila y lo vuelve a cargar en memoria

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JSPs. Sintaxis (2/7)

- **Variables implícitas:**
  - **request** (HttpServletRequest)
  - **response** (HttpServletResponse)
  - **out** (PrintWriter)
  - **session** (HttpSession)
  - **application** (ServletContext)
  - **config** (ServletConfig)
  - **pageContext** (PageContext)
  - **page** (this)

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JSPs. Sintaxis (3/7)

- Archivo compuesto por:
  - **Contenido cliente** (HTML, XML y Javascript)
  - **Comentario <%--** comentario **--%>**
  - **Scriptlet**: código servidor Java, escrito entre **<%** codigo Java **%>**
  - **Expresión**: inserta valores Java directamente en la salida
    <%= expresión Java **%>**
  - **Declaración**: define métodos o campos que son insertados dentro de la clase Servlet **<%!** codigo Java **%>**
  - **Directiva**: dos tipos principales “page” e “include”
    <%@ directiva atributo=”valor” **%>**

    <%@ page import=”clase” %> <%@ include file=”fichero” %>
  - **Tag**: instrucción en formato XML asociada a clases Java

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JSPs. Sintaxis (4/7)

- **Acciones**: construcciones XML que controlan el comportamiento del motor de Servlets
  - <**jsp:attribute name**=”nombre”>
  - <**jsp:body**>
  - <**jsp:element name**=”nombre”>
  - <**jsp:include page**=”url”>
  - <**jsp:forward page**=”url”>
  - <**jsp:param name**=”nombre” **value**=”valor”>
  - <**jsp:text**>
  - <**jsp:useBean id**=”nombre” **class**=”clase” **/**>
  - <**jsp:setProperty name**=”idBean” **property**=”prop” **value**=”valor”>
  - <**jsp:getProperty name**=”idBean” **property**=”prop”>

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JSP. Ejemplo: “JSP TEST” (5/7)

EJEMPLO

- JSP que genera un HTML utilizando:
  - Expresión: devuelve el nombre del equipo
  - Scriptlet: obtiene los datos enviados por el método GET
  - Declaración: variable que mide el número de accesos a esa página web
  - Directiva page: importa la clase “java.util.*”
  - Directiva include: incluye el fichero “foot.html”
- Fichero:
  - jspTest.jsp

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JSPs. Ejemplo: "JSP TEST" (6/7)

jspTest.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <title>Ejemplo JSP</title>
  <meta name="GENERATOR" content="Quanta Plus">
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<ul>
  <li><b>Expresi&oacute;n:</b><br />
      Tu equipo: <%= request.getRemoteHost() %>.
  </li>
  <li><b>Scriptlet:</b><br />
      <%  String peticion = request.getQueryString();
      if (peticion != null)
        out.println ("Datos GET: " + peticion);
          else
            out.println ("No hay datos GET");
      %>
  </li>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JSPs. Ejemplo: "JSP TEST" (7/7)

```
  <li><b>Declaraci&oacute;n:</b><br />
    <%! private int accessCount = 0; %>
    N&#186; de accesos a la p&aacute;gina desde que arranc&oacute; el
servidor: <%= ++accessCount %>
  </li>
  <li><b>Directiva "page":</b><br />
    <%@ page import = "java.util.*" %>
    Fecha actual: <%= new Date() %>
  </li>
  <li><b>Directiva "include" fichero:</b><br />
    <%@ include file="foot.html" %>
  </li>
</ul>

</body>
</html>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Combinación de Servlets y JSPs. Ejemplo: “Hola Mundo Servlet JSP” (1/3)

EJEMPLO

- Imprime la cadena “Hola Mundo desde un JSP” en una página web
- Ficheros:
  - HolaMundoServletJSP.java
  - holaMundo.jsp
  - header.jsp
  - footer.jsp
  - web.xml

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Combinación de Servlets y JSPs. Ejemplo: "Hola Mundo Servlet JSP" (2/3)

HolaMundoServletJSP.java

```
package servlets;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.*;

public class HolaMundoServletJSP extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String message = "Hola Mundo desde un JSP";
        request.setAttribute("message", message);
        request.getRequestDispatcher("holaMundo.jsp").forward(request,
response);
    }
}
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Combinación de Servlets y JSPs. Ejemplo: "Hola Mundo Servlet JSP" (3/3)

holaMundo.jsp

```
<jsp:include page="/WEB-INF/jspf/header.jsp">
   <jsp:param name="title" value="Hola mundo"/>
</jsp:include>

  <strong><%=request.getAttribute("message")%></strong>

<jsp:include page="/WEB-INF/jspf/footer.jsp"/>
```

header.jsp

```
<html>
  <head>
    <title><%=request.getParameter("title")%></title>
  </head>
  <body>
```

footer.jsp

```
  </body>
</html>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

TagLib. Introducción (1/7)

- Son librerías de etiquetas que permiten crear etiquetas personalizadas para incluir en los JSPs
- Permiten minimizar código Java en JSPs
- JSTL (Java Standard Template Library): Biblioteca que implementa funciones de uso frecuente en aplicaciones JSP. Incluye:
  - Funciones comunes de iteración sobre datos, operaciones condicionales, e importación de otras páginas
  - Internacionalización y formateo de texto
  - Funciones de manipulación de cadenas
  - Procesamiento de XML
  - Acceso a bases de datos (NO RECOMENDADO)
  - Un lenguaje de expresión para referenciar objetos y sus propiedades sin necesidad de código Java

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

TagLib. Ejemplo: “Mostrar Email y Poner Mayusculas” (2/7)

EJEMPLO

- Servlet que recoge los datos de un usuario introducidos en un formulario y redirige a un JSP que utiliza tres TagLibs:
  - muestraEmail: sustituye el carácter @ de un email por la cadena “ at “
  - ponMayusculas: pone en mayúsculas el atributo de un usuario
  - horaSistema: imprime la hora del sistema
- Ficheros:
  - index.html
  - ProcesarUsuario.java
  - respuestaUsuario.jsp
  - MostrarEmail.java
  - daw.tld
  - web.xml

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

TagLib. Ejemplo: “Mostrar Email y Poner Mayusculas” (3/7)

ProcesarUsuario.java

```
package servlets;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.*;

import beans.Usuario;

public class ProcesarUsuario extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
        Usuario usuario = new Usuario();
        usuario.setEmail(request.getParameter("email"));
        usuario.setNombre(request.getParameter("nombre"));
        usuario.setApellidos(request.getParameter("apellidos"));
        request.setAttribute("usuario", usuario);
        request.getRequestDispatcher("respuestaUsuario.jsp").
            forward(request, response);
    }
}
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

TagLibs. Ejemplo: "Mostrar Email y Poner Mayusculas" (4/7)

respuestaUsuario.jsp

```
<%@ taglib prefix="daw" uri="daw" %>
<%@ page import="beans.Usuario"%>

<html>
<head><title>Ejemplo de uso de beans</title></head>
<body>
<jsp:useBean id="usuario" class="beans.Usuario"
   scope="request"></jsp:useBean>
Ha introducido los siguientes datos:
<br>
Email:
<strong><daw:muestraEmail valor="<%=usuario.getEmail()%>"/></strong>
<br>
Nombre:
<strong><daw:ponMayusculas nombre="usuario" propiedad="nombre"/></strong>
<br>
Apellidos:
<strong><daw:ponMayusculas nombre="usuario"
   propiedad="apellidos"/></strong>
<br>
Hora del sistema: <strong><daw:horaSistema/></strong>
</body>
</html>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

TagLibs. Ejemplo: "Mostrar Email y Poner Mayusculas" (5/7)

MostrarEmail.java

```
package taglibs;

import java.io.IOException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;

public class MostrarEmail extends TagSupport {

    private String valor;

    public int doStartTag() throws JspException {
        try {
            pageContext.getOut().println(valor.replace("@", " at "));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return SKIP_BODY;
    }
    public void setValor(String valor) {
        this.valor = valor;
    }
}
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

TagLibs. Ejemplo: “Mostrar Email y Poner Mayusculas” (6/7)

daw.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library
1.1//EN"
 "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
 <taglib>
    <tlibversion>1.0</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>daw</shortname>
    <uri>/WEB-INF/tlds/daw.tld</uri>
    <info>My first Tag library</info>
    <tag>
        <name>muestraEmail</name>
        <tagclass>taglibs.MostrarEmail</tagclass>
        <info>Muestra el email sustituyendo @ por at</info>
        <attribute>
            <name>valor</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>
    </tag>
</taglib>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

TagLibs. Ejemplo: “Mostrar Email y Poner Mayusculas” (7/7)

web.xml

```
<taglib>
   <taglib-uri>daw</taglib-uri>
   <taglib-location>/WEB-INF/daw.tld</taglib-location>
</taglib>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Introducción (1/8)

- API que permite hacer consultas a una base de datos relacional
- Es una **ESPECIFICACIÓN,** los drivers los proporcionan los diferentes fabricantes de bases de datos
- Un driver es un JAR (Java Archive) con la implementación de los interfaces del API de JDBC
- Lo ideal es que si se cambia de BBDD, no se necesite cambiar el código.
  - Inconveniente: Se usa diferente sintaxis SQL
  - Existen patrones para resolver este problema

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Ejemplo “Mostrar usuarios de una BBDD” (2/8)

EJEMPLO

- Obtiene los datos de los usuarios que hay en la tabla “usuarios” de la base de datos “daw0607” y se imprimen en un JSP.
- Ficheros:
  - DatabaseManager.java
  - UsuarioDAO.java
  - MostrarUsuariosJDBC.java
  - listarUsuarios.jsp
- Conector MySQL:
  - mysql-connector-java-5.0.4-bin.jar

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Ejemplo "Mostrar usuarios de una BBDD" (3/8)

DatabaseManager.java

```
package db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseManager {

    private static String CONEXION = "jdbc:mysql://localhost/daw0607";
    private static String USER = "rcedazo";
    private static String PASSWORD = "mypassword";
    private static DatabaseManager instance;

    private DatabaseManager() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (java.lang.ClassNotFoundException e1) {
            System.err.println(e1.getMessage());
        }
    }
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Ejemplo "Mostrar usuarios de una BBDD" (4/8)

```
    public static DatabaseManager getInstance() {
        if (instance == null) {
            instance = new DatabaseManager();
        }
        return instance;
    }

    public Connection getConnection() throws SQLException {
        return DriverManager.getConnection(CONEXION, USER, PASSWORD);
    }
}
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Ejemplo "Mostrar usuarios de una BBDD" (5/8)

UsuarioDAO.java

```
package dao;

import java.sql.*;
import java.util.*;
import beans.Usuario;
import db.DatabaseManager;

public class UsuarioDAO {

    public UsuarioDAO() {
    }

    public Collection getUsuarios() {
        Statement st = null;
        ResultSet rs = null;
        Connection connection = null;
        try {
            connection = DatabaseManager.getInstance().getConnection();
            st = connection.createStatement();
            rs = st.executeQuery("SELECT * FROM usuarios");
            Collection resultado = new ArrayList();
            Usuario usuario;
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Ejemplo "Mostrar usuarios de una BBDD" (6/8)

```
            while (rs.next()) {
                usuario = new Usuario();
                usuario.setEmail(rs.getString("EMAIL"));
                usuario.setNombre(rs.getString("NOMBRE"));
                usuario.setApellidos(rs.getString("APELLIDOS"));
                resultado.add(usuario);
            }
            return resultado;
        } catch (SQLException e) {
            System.err.println(e.getMessage());
            return null;
        } finally {
            try {
                if (rs != null)
                    rs.close();
                if (st != null)
                    st.close();
                if (connection != null)
                    connection.close();
            } catch (Exception e) {
            }
        }
    }
}
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Ejemplo "Mostrar usuarios de una BBDD" (7/8)

MostrarUsuariosJDBC.java

```
public class MostrarUsuariosJDBC extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        Collection usuarios = new UsuarioDAO().getUsuarios();
        request.setAttribute("usuarios", usuarios);
        request.getRequestDispatcher("listarUsuarios.jsp").forward(request,
response);
    }
}
```

listarUsuarios.jsp

```
<%@ page import="java.util.*"%>
<%@ page import="beans.Usuario"%>

<html>
<head>
<title>Ejemplo de uso de beans</title>
</head>
<body>
  <table border="1" cellpadding="5" cellspacing="0">
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

JDBC. Ejemplo "Mostrar usuarios de una BBDD" (8/8)

```
  <thead>
    <th>Email</td>
    <th>Nombre</td>
    <th>Apellidos</td>
  </thead>
  <tbody>
<%
  Collection usuarios = (Collection)request.getAttribute("usuarios");
  Iterator it = usuarios.iterator();
  while (it.hasNext()) {
    Usuario usuario = (Usuario) it.next();
%>
    <tr>
      <td><%=usuario.getEmail()%></td>
      <td><%=usuario.getNombre()%></td>
      <td><%=usuario.getApellidos()%></td>
    </tr>
<%
  }
%>
  </tbody>
</table>
</body>
</html>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

ANT. Introducción (1/4)

- Proyecto de Apache: http://ant.apache.org
- Herramienta usada para la realización de tareas automáticas y repetitivas (compilación, construcción)
- Del tipo *Make,* pero sin dependencias del sistema operativo
- Escrita en JAVA
- Muy portable: utiliza ficheros XML
  - Cada fichero contiene un proyecto
  - Cada proyecto contiene uno o más targets (tareas que se ejecutan secuencialmente)
  - Se pueden establecer dependencias entre targets
  - Se pueden definir propiedades

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

ANT. Ejemplo: build.xml (2/4)

EJEMPLO

- Archivo build.xml para generar automáticamente el WAR (Web Archive) de la aplicación de ejemplos de J2EE
- Tareas:
  - clean
  - compile
  - generateWar

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

ANT. Ejemplo: build.xml (3/4)

build.xml

```
<?xml version="1.0"?>

<project name="DAW0607" default="generateWar" basedir="..">
    <description>
            Generación automática del WAR
    </description>

    <property name="srcDir" value="${basedir}/src"/>
    <property name="libDir" value="${basedir}/lib"/>
    <property name="buildDir" value="${basedir}/build"/>
    <property name="webDir" value="${basedir}/web"/>
    <property name="warFile" value="${buildDir}/DAW0607.war"/>

    <property name="binDir" value="${buildDir}/bin"/>

    <path id="compileLibs">
        <fileset dir="${libDir}">
            <include name="**/*.jar"/>
        </fileset>
    </path>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

ANT. Ejemplo: build.xml (4/4)

```
    <target name="clean">
        <delete dir="${binDir}" includeemptydirs="true"/>
        <delete file="${warFile}"/>
    </target>

    <target name="compile" depends="clean">
        <mkdir dir="${binDir}"/>
        <javac srcdir="${srcDir}" destdir="${binDir}"
            classpathref="compileLibs">
        </javac>
    </target>

    <target name="generateWar" depends="compile">
        <war destfile="${warFile}" webxml="${webDir}/WEB-INF/web.xml"
            basedir="${webDir}">
            <exclude name="WEB-INF/web.xml"/>
            <exclude name="WEB-INF/classes/"/>
            <exclude name="WEB-INF/lib/"/>
            <classes dir="${binDir}"></classes>
            <lib dir="${webDir}/WEB-INF/lib"></lib>
        </war>
    </target>

</project>
```


Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

EasyEclipse

- Distribuciones de Eclipse con la funcionalidad necesaria para tener un entorno de desarrollo específico, sin necesidad de descargar software adicional (plugins, librerías...)
- **EasyEclipse Server Java**: aplicaciones Java del lado del servidor, tales como JSPs, EJBs y Web Services.
- **EasyEclipse Mobile Java:** aplicaciones Java para dispositivos con J2ME.
- **EasyEclipse for PHP**: desarrollar PHP con una base de datos.
- **EasyEclipse Desktop Java**: para desarrollar interfaces gráficas con Swing o SWT.
- Y más en: http://www.easyeclipse.org

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Referencias

- Tutoriales Java EE:
  http://java.sun.com/javaee/reference/tutorials
- Eclipse plugins:
  http://eclipse-plugins.2y.net/
- ANT Apache:
  http://ant.apache.org [Manual: http://ant.apache.org/manual/]
- EasyEclipse:
  http://www.easyeclipse.org
- The Essentials of Filters:
  http://java.sun.com/products/servlet/Filters.html
- Manual de JTSL:
  http://www.1x4x9.info/files/jstl/html/online-chunked/

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es

Java 2 Enterprise Edition

The Apache Jakarta Project
http://jakarta.apache.org/

Raquel CEDAZO LEÓN
rcedazo@fi.upm.es