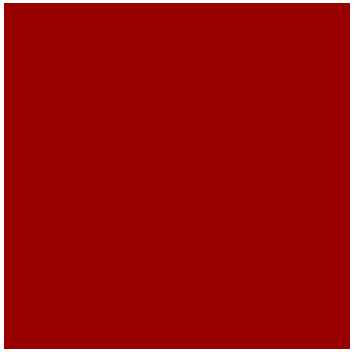
Aplicaciones Open Source

Semana02
31/03/2018

Carlos A. Quinto Cáceres
pcsicqui@upc.edu.pe

Agenda

- JSP
- JDBC



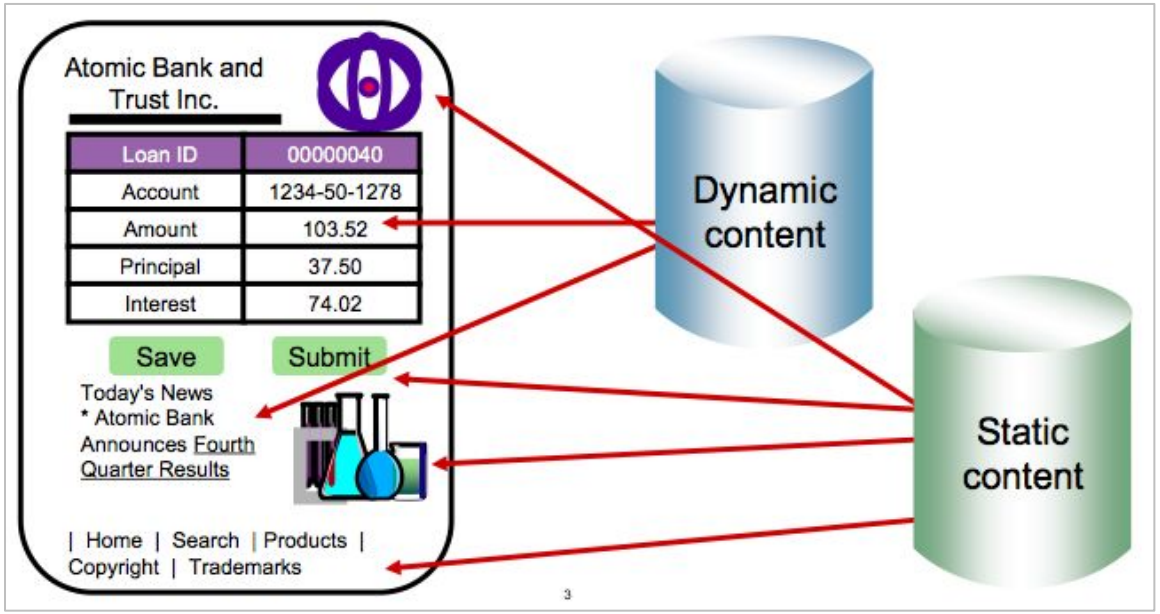
Logros del día

- Al finalizar la clase, el alumno podrá:
  - Conectarse a una base de datos desde una aplicación Web de Java.
  - Ejecutar consultas DML en la base de datos desde una aplicación Web de Java.
  - Consultar información en la base de datos desde una aplicación Web de Java.

JSP

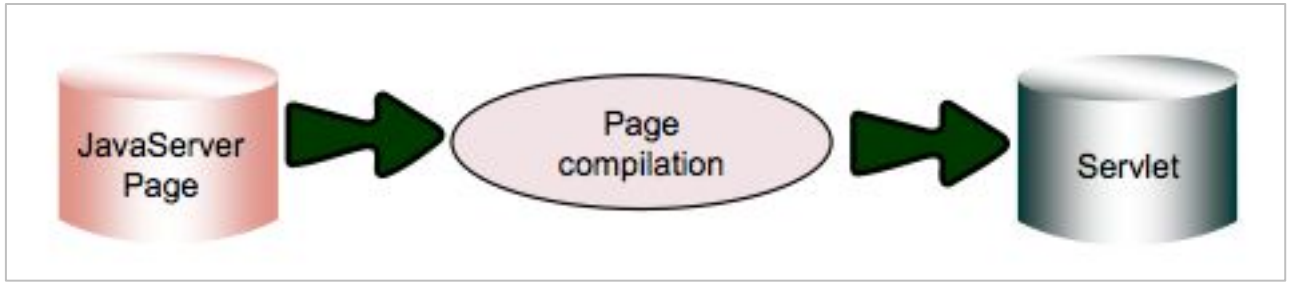
JSP

- Es la tecnología Java que permite combinar HTML estático con HTML generado dinámicamente.
- Podemos utilizar código de servidor, en este caso código Java para generar dinámicamente código HTML.



JSP

- Proceso de compilación de las páginas JSP:
  - El código del JSP es analizado.
  - El código del Servlet es generado.
  - El Servlet generado es compilado, cargado y ejecutado.



JSP

- **Directivas**, son instrucciones para el compilador y motor JSP:
  - <%@ import="clase" %> <%@ include file="archivo.jsp" %>
  - <%@ page isErrorPage="true" %> <%@ page errorPage="file.jsp" %>
- **Scripting**:
  - **Declaraciones**, sirven para declarar métodos y variables
    - <%! codigo %>
  - **Scriplets**, son líneas de código Java
    - <% codigo %>
  - **Expresiones**, es código Java que resulta en una cadena
    - <%=variable%>

JSP

- **Comentarios**:
  - <%-- texto de comentario --%>
- **Actions**
  - <jsp:include page="StdHeader.jsp" flush="true" />
  - <jsp:forward page="ExtraInfo.jsp" />

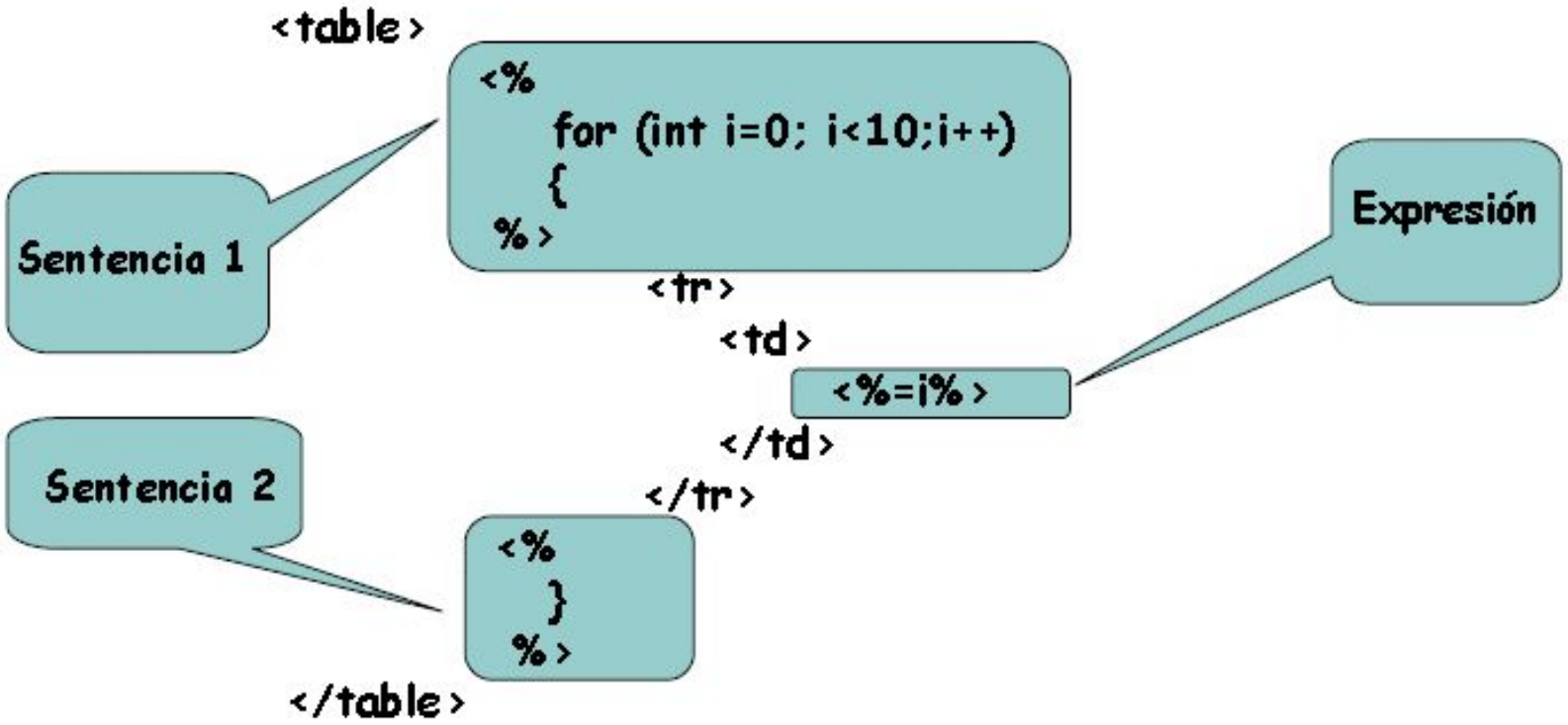
JSP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HEAD><TITLE></TITLE></HEAD>
<BODY>
<%TestAreas tests=
  (TestAreas)request.getAttribute("tests");%>

<TABLE cellspacing="10"><TBODY>
<TR>
<TD><FONT color="#0000cc" size="5" face="Comic Sans MS">
The following areas have tests available</FONT></TD>
</TR>

<%for (int i = 0; i < tests.getTestAreas().length; i++){%>
  <TR><TD align="center">

  <!-- The following two lines are really 1 line -->
  <A href="/Course/ExamCommand?cmd=displayTestsByArea&testArea=
  <%= tests.getTestAreas(i).getKey()%>">

  <IMG border="0"
   src="<%= tests.getTestAreas(i).getImageFileWithPath() %>">
  </A>
  </TD></TR>
<%}%>

</TBODY> </TABLE> </BODY>
```

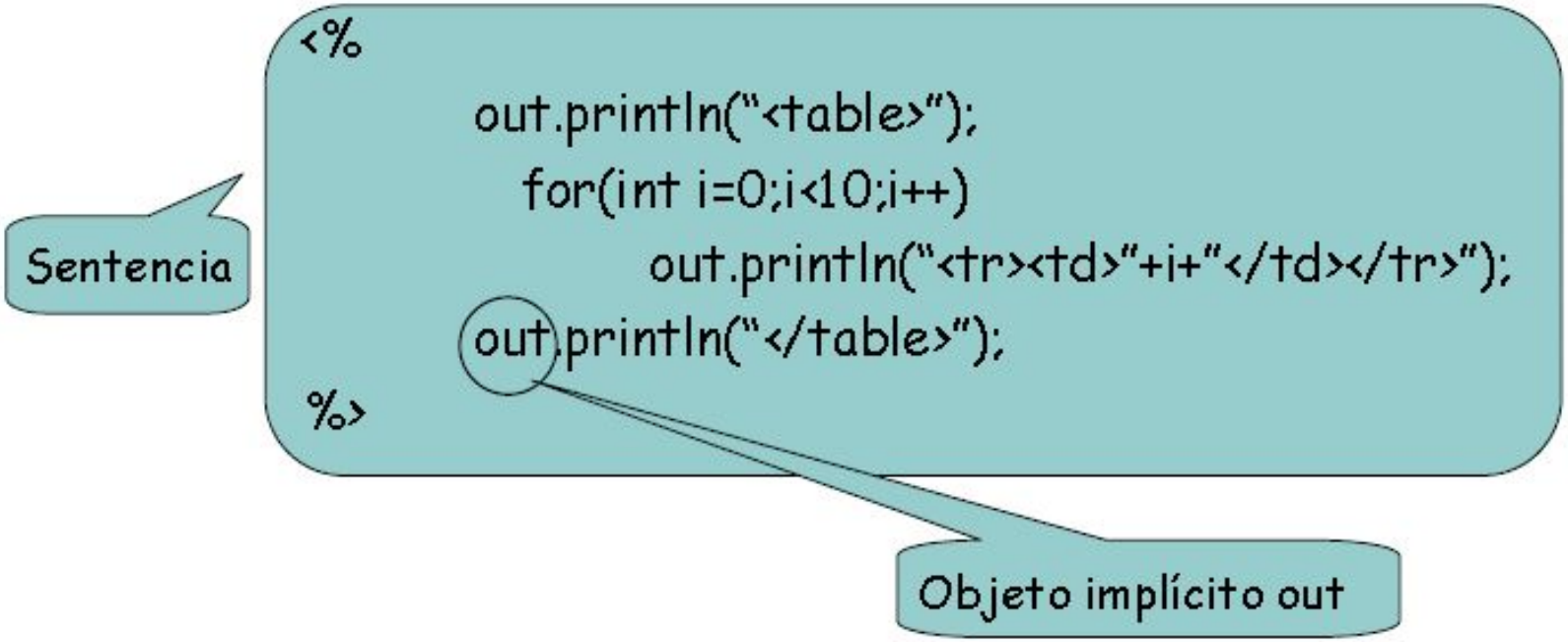
- Servlet = Código Java + HTML
- JSP = HTML + Código Java

```
<%@ page language="java" %>

<%@ include file="companyBanner.html"%>
```

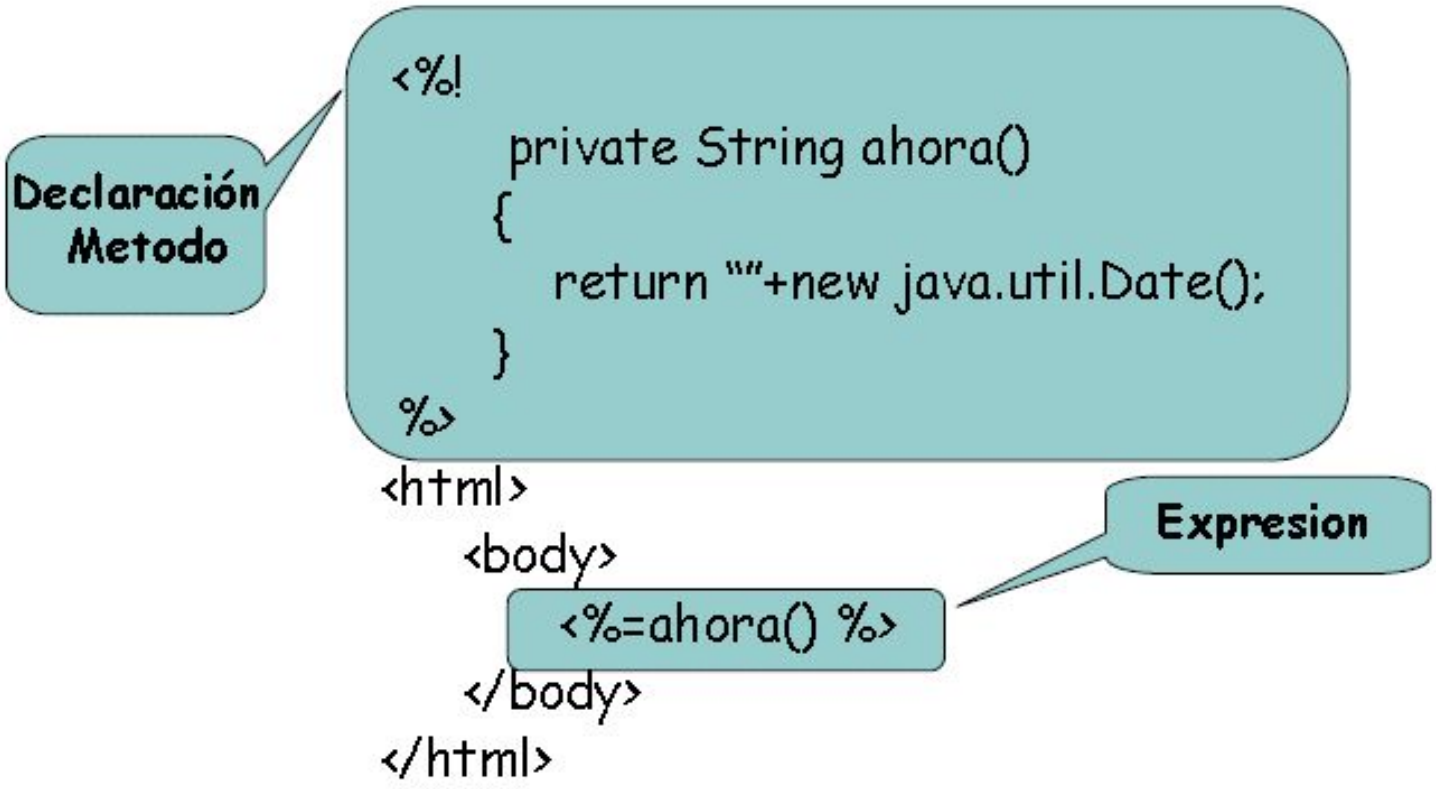
Ejemplos de JSP



Ejemplos de JSP

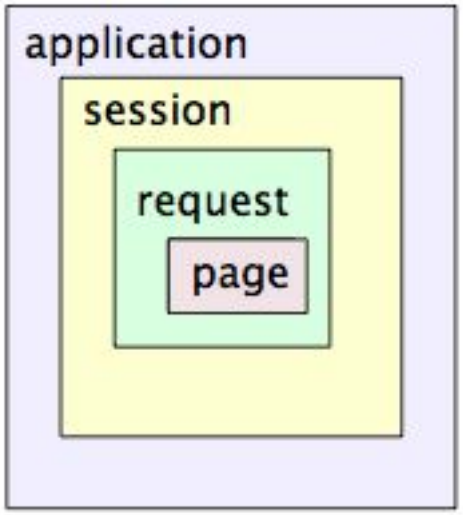


Ejemplo de JSP



Ámbitos

Servlet	JSP
PageContext	Page
HttpServletRequest	request
HttpSession	session
ServletContext	Application



JSP

- **Page,** el alcance es corto, es parecido a tener una variable local.
- **Request**, el alcance se mantiene a través del request hasta que la respuesta es enviada al cliente (HttpServletRequest).
  - request.getAttribute()
  - request.setAttribute()
- **Session**, se tiene el alcance mientras se mantenga la sesión del usuario (HttpSession).
  - session.getAttribute()
  - session.setAttribute()
- **Application**, permite compartir información para toda la aplicación (ServletContext).
  - application.getAttribute()
  - application.setAttribute()

JSP

- Variables predefinidas:
  - **request**, objeto HttpServletRequest
  - **response**, objeto HttpServletResponse
  - **session**, objeto HttpSession
  - **application**, objeto HttpServletContext
  - **config**, objeto ServletConfig
  - **out**, JspWriter

JSP

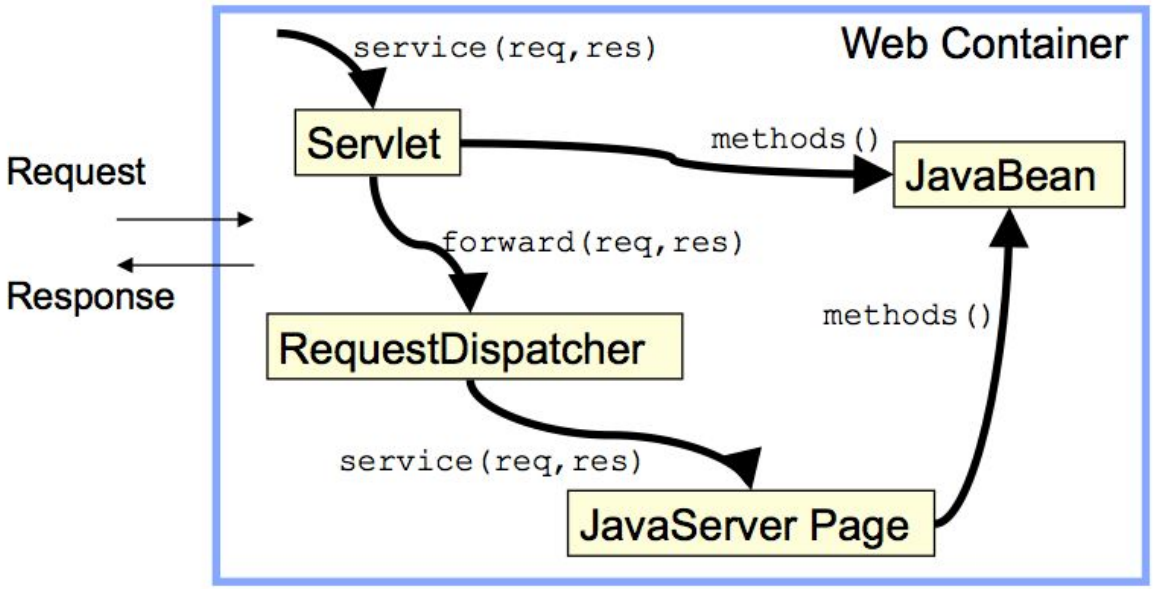
- La página JSP puede ser invocada:
  - Por URL
  - Por un Servlet
  - Por otra página JSP
- Una página JSP puede invocar:
  - Un Servlet
  - Otra página JSP

¿JSP o Servlet?

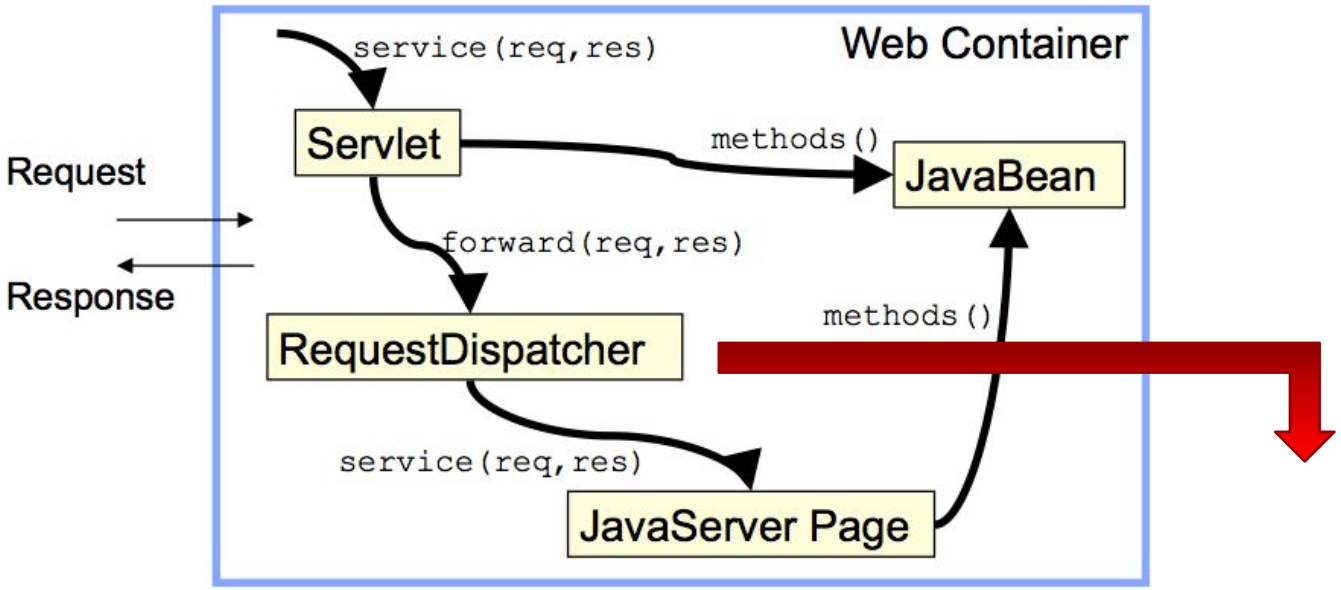
- Servlets
  - Determinar los procesos requeridos para cumplir con la petición
  - Validar la información enviada en la petición
  - Acceder a los datos y realizar el proceso requerido por la petición
  - Controlar el flujo de la aplicación Web
- JSP
  - Mostrar el contenido generado por la aplicación Web

Aplicaciones con JSP

1. El Servlet recibe las peticiones del cliente.
2. El Servlet ejecuta la lógica de negocio que tiene desarrollado.
3. Realiza una redirección a la página JSP utilizando RequestDispatcher.
4. El JSP contiene el HTML de respuesta.



Aplicaciones con JSP



Recurso, puede ser:

- Alias de Servlet
- Página JSP

```
request.getRequestDispatcher("/recurso")
       .forward(request, response);
```

Ejercicio

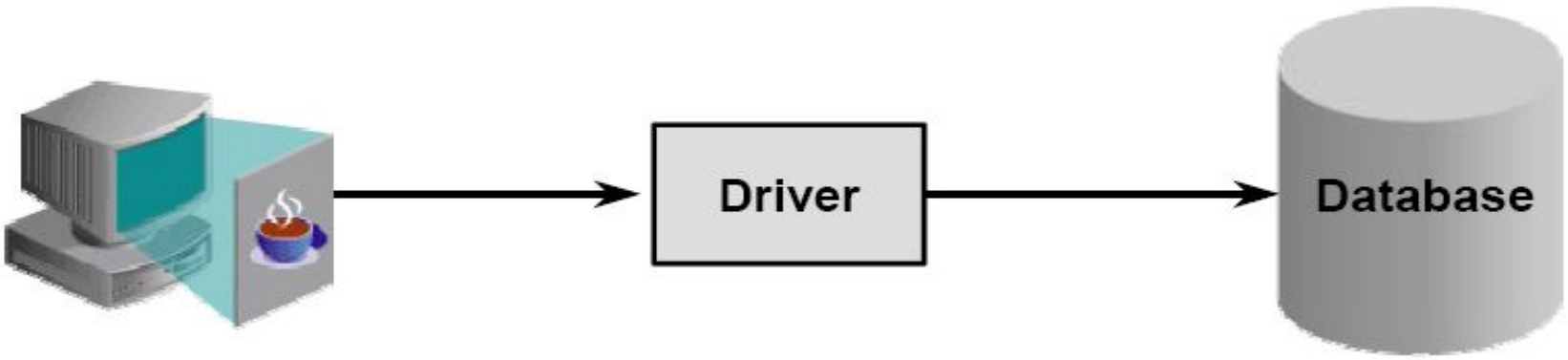
1. Crear la página JSP index.jsp
   - Hacer uso de scriplets
   - Crear una variable y mostrar su valor, hacer uso de declaraciones y expresiones
   - Incluir comentarios en el JSP

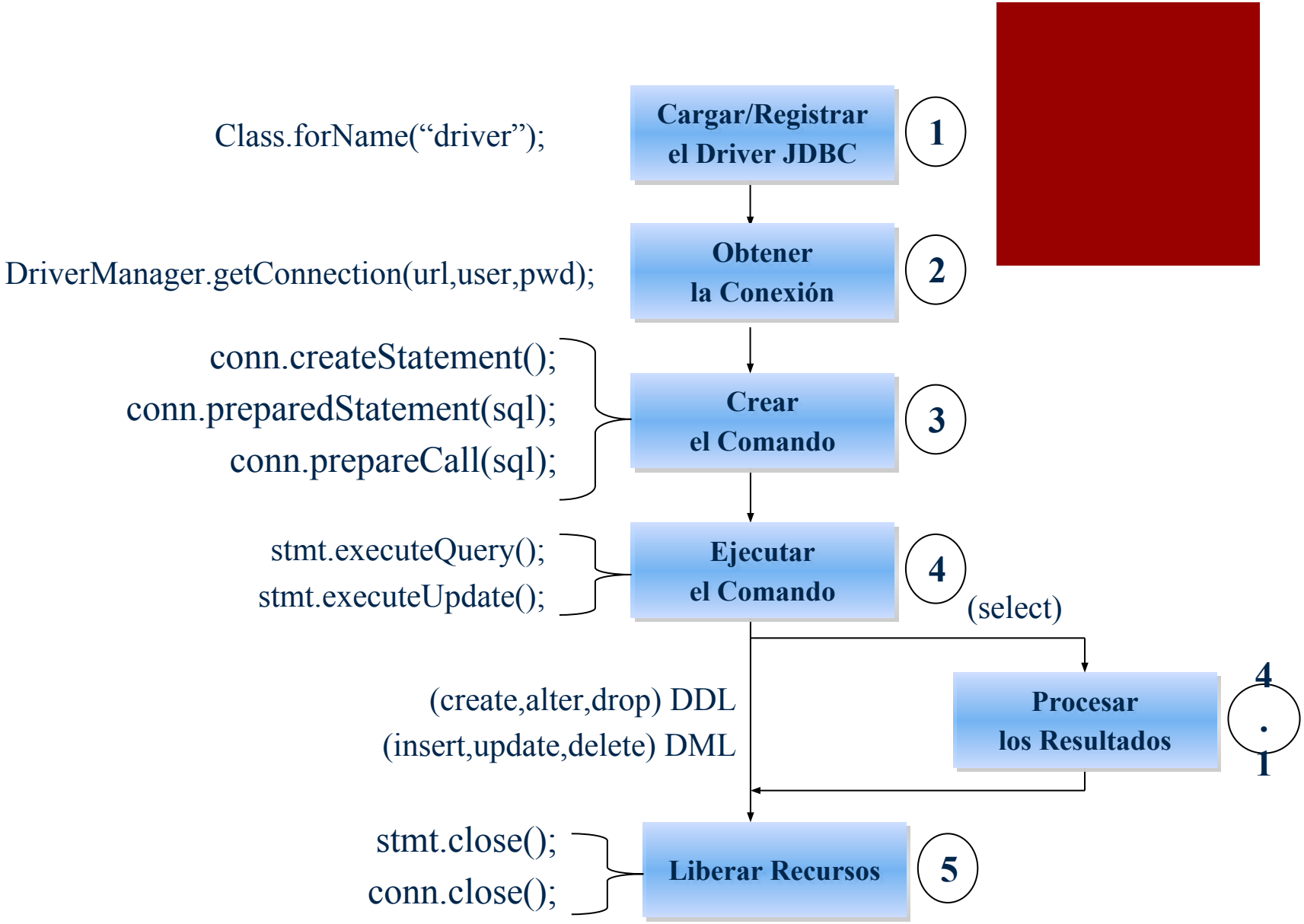
JDBC

JDBC

- Java Database Connectivity.
- Es una API (Application Programming Interface) para Java que nos permite realizar operaciones sobre un origen de datos, sea una base de datos o una fuente de datos tabular.
- Nos proporciona métodos para consultar y actualizar información de una base de datos.

JDBC




Class.forName("driver");
Cargar/Registrar el Driver JDBC
1
DriverManager.getConnection(url,user,pwd);
Obtener la Conexión
2
conn.createStatement();
conn.preparedStatement(sql);
conn.prepareCall(sql);
Crear el Comando
3
stmt.executeQuery();
stmt.executeUpdate();
Ejecutar el Comando
4
(select)
(create,alter,drop) DDL
(insert,update,delete) DML
Procesar los Resultados
4.1
stmt.close();
conn.close();
Liberar Recursos
5

Cargar el Driver JDBC

- Instrucciones para la carga del driver
  - Class.forName(“NOMBRE_DRIVER”);
- El driver es cargado a memoria y la clase DriverManager lo gestiona

Obtener la Conexión

- Un objeto java.sql.Connection representa una conexión con una Base de Datos. Sobre una conexión pueden ser ejecutadas sentencias SQL y obtenerse resultados.
- La clase DriverManager provee el método getConnection, que requiere la URL JDBC para especificar los detalles de la conexión.
- Una base de datos en JDBC es identificada por una URL (Uniform Resource Locator). Especifica el nombre y la ubicación de la BD. La sintaxis recomendada para la URL JDBC es la siguiente:

```
Class.forName("com.sun.sql.jdbc.sqlserver.SQLServerDriver") ;
String url = "jdbc:sun:sqlserver://localhost:1433;DatabaseName=Pubs" ;
Connection miConexion = DriverManager.getConnection(url, "sa", "");
```

Crear y ejecutar el Comando

- java.sql.Statement
  - Se utiliza el método createStatement de una instancia Connection para su creación.
  - Permite la ejecución de una sentencia SQL, sus métodos principales son:
    1. executeQuery
       - Sentencias SELECT que devuelve un java.sql.ResultSet
    2. executeUpdate
       - Sentencias INSERT, DELETE, UPDATE, CREATE, que devuelve un entero
    3. execute
       - Sentencias desconocidas en tiempo de compilación o sentencias que devuelven resultados complejos, devuelve true/false.

Crear y Ejecutar el Comando

- java.sql.PreparedStatement
  - Se utiliza el método preparedStatement de una instancia Connection para su creación.
  - Extiende Statement para añadir sentencias pre compiladas SQL, que compila la sentencia SQL la primera vez. Éstas son llamadas más de una vez en el programa.
  - Soporta parámetros de entrada: setInt, setFloat, setLong, setString

- java.sql.CallableStatement
  - Se utiliza el método prepareCall de una instancia Connection para su creación.
  - Extiende la funcionalidad de PreparedStatement, permite la invocación de procedimientos almacenados, si el manejador los soporta.

Procesar los Resultados

- java.sql.ResultSet
  - Contiene los datos resultado de una sentencia SQL que se recuperan secuencialmente en filas
  - El método next() sirve para avanzar una fila
  - Se puede acceder a los datos de las columnas en cualquier orden por índice de posición o nombre del campo
  - Provee métodos para recuperar los datos de un campo: getString, getFloat, getInt, etc.
  - El método wasNull() indica si el campo contiene valores nulos.

5. Liberar Recursos

- Las clases:
  - Connection
  - Statement, PreparedStatement, CallableStatement
  - ResultSet
- Proveen el método close(), que activa la realización de la instancia específica, que liberan todos los recursos JDBC que se ejecutan sobre el servidor de datos.
- Cuando se invoca el método close() de un Statement, los ResultSet asociados son cerrados automáticamente.
- La invocación al método close() de Connection, puede provocar un SQLException si esta ya está cerrada.

JDBC

- Clases disponibles, estas clases se importan desde **java.sql**:
  - **DriverManager**

    Sirve para cargar un driver específico
  - **Connection**

    Permite establecer una conexión a un origen de datos
  - **Statement**

    Para crear consultas SQL y enviarlas al origen de datos
  - **ResultSet**

    Permite almacenar el resultado de una consulta

Código

1. Cargar driver

   Class.forName("com.mysql.jdbc.Driver");

2. Obtener conexión

   String url = “jdbc:mysql://localhost:3306/mibd”;

   Connection con =

   DriverManager.getConnection(url, “user”, “clave”);

3. Crear la sentencia

   Statement stmt = con.createStatement();

Código

El cuarto paso, dependerá del tipo de comando SQL:

- **Modificar información**

  int filas_afectadas = stmt.executeUpdate(String sql);

  Ejemplo:

  String query= “insert into cursos(codigo, nombre) values(‘30F’, ‘prog1’)”;

  int filas_afectadas = stmt.executeUpdate(query);

- **Consultar información**

  ResultSet rs = stmt.executeQuery(String sql);

  Ejemplo:

  String query= “select * from cursos”;

  ResultSet rs = stmt. executeQuery(query);

Laboratorio

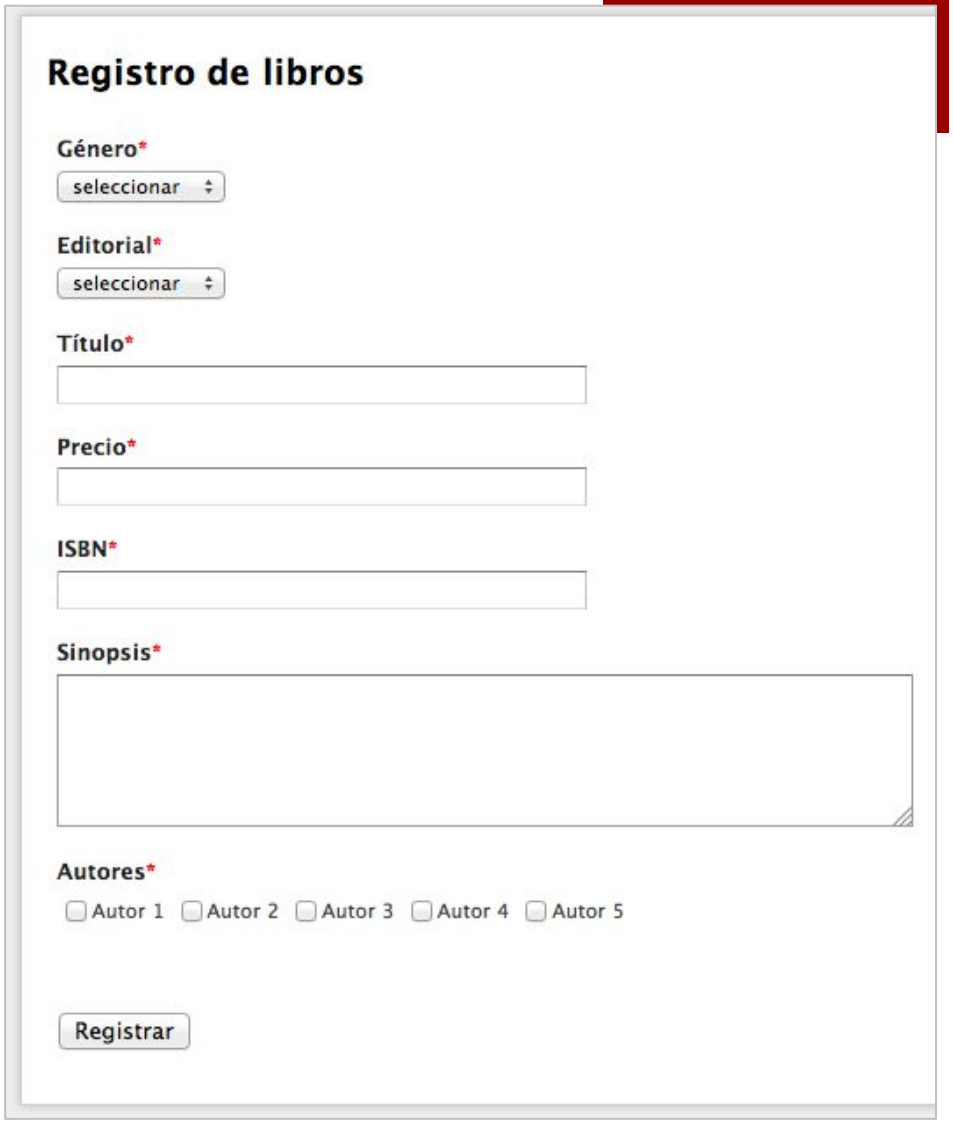
- Conocer las herramientas MySQL
- Crear una base de datos y tabla de ejemplo
- Realizar una copia de la base de datos
- Restaurar una base de datos
- Realizar mantenimiento de la tabla autores.

Laboratorio

- Crear la siguiente estructura en la aplicación:
  - WebContent/admin/
    - Esta carpeta contendrá los archivos, componentes y páginas disponibles para el administrador.
    - Crear la página principal.jsp, que tenga las opciones:
      - **Ver listado de autores y Agregar nuevo autor**
  - WebContent/
    - Los archivos, componentes y páginas en la raíz de la aplicación serán de acceso publico a los visitantes.
    - Crear index.jsp, que será la home de la aplicación

Ejercicio

- Dentro del módulo de administración crear la pagina libro_agregar.jsp que va a permitir agregar nuevos libros a la base de datos.
- La información de autores, géneros y editoriales debe ser cargada de base de datos.



Enlaces de interés

- JDBC

  http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

- JSP
  - http://www.oracle.com/technetwork/java/javaee/jsp/index.html