UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
Lic. En Sistemas Computacionales	2003-1	5038	Programación Orientada a Objetos 2

PRÁCTICA No.	LABORATORIO DE		DURACIÓN (HORA)
12	NOMBRE DE LA PRÁCTICA	JSP	2

1 INTRODUCCIÓN

En esta práctica se utilizarán los elementos necesarios para el desarrollo de un sistema de de informacion WEB.

Para trabajar con Java Server Pages se requieren conocimientos previos de java y HTML, asi como de Java Script.

2 OBJETIVO (COMPETENCIA)

El alumno será capaz de desarrollar páginas dinámicas utilizando Java Server Pages.

3 FUNDAMENTO

Java Server Pages

- Ventajas:
  - Separación de datos estáticos/dinámicos.
  - Independencia de formato/plataforma.
  - Sencillez (sabiendo servlets)

Aunque el código parezca mas bien HTML, el servidor lo traduce a un servlet en la primera petición.

3 elementos en un JSP:

- ✔ Elementos script (scriptlets)
- ✔ Directivas
- ✔ Acciones

Expresión JSP

```
<%= expression %>;
```

La Expresión es evaluada y situada en la salida.
El equivalente XML es
`<jsp:expression> expression </jsp:expression>`
Las variables predefinidas son `request, response, out, session, application, config`, y `pageContext`.

Scriptlet JSP

```
<% code %>;
```

El código se inserta en el método `service`.
El equivalente XML es:
`<jsp:scriptlet>code</jsp:scriptlet>`.

Declaración JSP

`<%! code %>`

El código se inserta en el cuerpo de la clase del servlet, fuera del método `service`.
El equivalente XML es:
`<jsp:declaration> code </jsp:declaration>`.

Directiva page JSP

`<%@ page att="val" %>`

Dirige al motor servlet sobre la configuración general.

El equivalente XML es:
`<jsp:directive.page att="val"\>`.

Los atributos legales son (con los valores por defecto en negrita):

```
import="package.class"
contentType="MIME-Type"
isThreadSafe="true|false"
session="true|false"
buffer="sizekb|none"
autoflush="true|false"
extends="package.class"
info="message"
errorPage="url"
isErrorPage="true|false"
language="java"
```

Directiva include JSP

`<%@ include file="url" %>`

Un fichero del sistema local se incluirá cuando la página se traduzca a un Servlet.
El equivalente XML es: `<jsp:directive.include file="url"\>`

Comentario JSP

```
<%-- comment --%>
```

Comentario ignorado cuando se traduce la página JSP en un servlet.
Si queremos un comentario en el HTML resultante, usamos la sintaxis de comentario normal del HTML `<!-- comment -->`.

Acción `jsp:include`

```
<jsp:include page="relative URL" flush="true"/>
```

Incluye un fichero en el momento en que la página es solicitada.
Aviso: en algunos servidores, el fichero incluido debe ser un fichero HTML o JSP, según determine el servidor (normalmente basado en la extensión del fichero).

Acción jsp:useBean

```
<jsp:useBean att=val*/>
<jsp:useBean att=val*>
 ...
</jsp:useBean>
```

Encuentra o construye un Java Bean.
Los posibles atributos son:

```
id="name"
scope="page|request|session|application"
class="package.class"
type="package.class"
beanName="package.class"
```

Acción jsp:setProperty

```
<jsp:setProperty att=val*/>
```

Selecciona las propiedades del bean, bien directamente o designando el valor que viene desde un parámetro de la petición.
Los atributos legales son:

```
name="beanName"
property="propertyName|*"
param="parameterName"
value="val"
```

Acción jsp:getProperty

```
<jsp:getProperty name="propertyName" value="val"/>
```

Recupera y saca las propiedades del Bean.

Acción jsp:forward

```
<jsp:forward page="relative URL"/>
```

Reenvía la petición a otra página.

EXPRESIONES: `<%= expresión %>` Se evalúan y se insertan en la salida.
Se tiene acceso a variables:
`request`, el `HttpServletRequest`
`response`, el `HttpServletResponse`
`session`, el `HttpSession` asociado con el `request` (si existe)
`out`, el `PrintWriter` (una versión con buffer del tipo JspWriter) usada para enviar la salida al cliente.

```
Your hostname: <%= request.getRemoteHost() %>
```

SCRIPTLETS: `<% codido %>` que se insertan dentro del método service del servlet.

Tienen acceso a las mismas variables que las expresiones.
El código dentro de un scriptlet se insertará **exactamente** como está escrito, y cualquier HTML estático (plantilla de texto) anterior o posterior al scriptlet se convierte en sentencias `print`. Esto significa que los scriptlets no necesitan completar las sentencias Java, y los bloques abiertos pueden afectar al HTML estático fuera de los scriplets

DECLARACIONES: `<%! codigo %>` que se insertan en el cuerpo de la clase del servlet, fuera de cualquier método existente.

Permite insertar métodos, variables...
No generan salida alguna. Se usan combinadas con scriptlets.

```
<%! private int accessCount = 0; %>
Accesses to page since server reboot:
<%= ++accessCount %>
```

Directivas

Afecta a la estructura general de la clase servlet. Normalmente tienen la siguiente forma:

```
<%@ directive attribute="value" %>
```

También podemos combinar múltiples selecciones de atributos para una sola directiva:

```
<%@ directive attribute1="value1"
attribute2="value2"
...
attributeN="valueN" %>
```

`include`: Permite incluir ficheros en el momento en que la página JSP es traducida a un servlet.

```
<%@ include file="url relativa" %>
```

Los contenidos del fichero incluido son analizados como texto normal JSP y así pueden incluir HTML estático, elementos de script, directivas y acciones.

Uso: Barras de navegación.

Variables predefinidas

`request:` Este es el `HttpServletRequest` asociado con la petición, y nos permite mirar los parámetros de la petición (mediante `getParameter`), el tipo de petición (GET, POST, HEAD, etc.), y las cabeceras HTTP entrantes (cookies, Referer, etc.). Estrictamente hablando, se permite que la petición sea una subclase de `ServletRequest` distinta de `HttpServletRequest`, si el protocolo de la petición es distinto del HTTP. Esto casi nunca se lleva a la práctica.

`response`**:** Este es el `HttpServletResponse` asociado con la respuesta al cliente. Como el stream de salida tiene un buffer, es legal seleccionar los códigos de estado y cabeceras de respuesta, aunque no está permitido en los servlets normales una vez que la salida ha sido enviada al cliente.

`out`: Este es el `PrintWriter` usado para enviar la salida al cliente. Sin embargo, para poder hacer útil el objeto response esta es una versión con buffer de `PrintWriter` llamada `JspWriter`. Podemos ajustar el tamaño del buffer, o incluso desactivar el buffer, usando el atributo buffer de la directiva `page`. Se usa casi exclusivamente en scriptlets ya que las expresiones JSP obtienen un lugar en el stream de salida, y por eso raramente se refieren explícitamente a `out`.

`session`: Este es el objeto `HttpSession` asociado con la petición. Las sesiones se crean automáticamente, por esto esta variable se une incluso si no hubiera una sesión de referencia entrante. La única excepción es usar el atributo `session` de la directiva `page` para desactivar las sesiones, en cuyo caso los intentos de referenciar la variable `session` causarán un error en el momento de traducir la página JSP a un servlet.

`page`: Esto es sólo un sinónimo de `this`, y no es muy útil en Java. Fue creado como situación para el día que el los lenguajes de script puedan incluir otros lenguajes distintos de Java.

Formato para prácticas de laboratorio

4 PROCEDIMIENTO (DESCRIPCIÓN)

A	EQUIPO NECESARIO	MATERIAL DE APOYO
	Computadora con Tomcat y maquina virtual de java	Practica impresa Tutoriales en internet sobre Java server Pages

B DESARROLLO DE LA PRÁCTICA

Ejercicios:

1. Elabore una pagina web que represente un negocio en linea Su negocio vendera cualquier articulo que usted desee al precio que usted defina. Se tendran tres requerimientos basicos para si sitio web.
El primero es que se incluyan por lo menos tres paginas (con extension html), una de las cuales sera la pagina de inicio y por lo menos dos paginas accesadas mediante la pagina de inicio.
El segundo requerimiento sera que se muestre una imagen por cada articulo que venda.
Tercero, las paginas deben tener un formato uniforme (look and feel), desde el inicio hasta el final todas las paginas deben tener las mismas caracteristicas en el sitio web.

2.Continuue con el ejercicio del negocio en linea. Modifique sus archivos para que en lugar de tener extension html tengan extension jsp. Despues agregue el codigo necesario para que cada pagina muestre la fecha actual.
(nota: puede utilizar java.text.DateFormat).
El ultimo requerimiento es que utilice por lo menos una expresion JSP, una declaracion JSP, un scriplet JSP, y una directiva import page de JSP en su sitio.

3. Ahora agregue los mecanismos necesarios para que los visitantes puedan hacer compras de los productos que se ofertan en su sitio. El proceso de compra se iniciara al dar click en un articulo. Esto llevera al cliente a una pantalla de confirmacion en la que ingresará su nombre y dirección de envío. Para este caso no se procesara la forma de pago.

C CÁLCULOS Y REPORTE

5 RESULTADOS Y CONCLUSIONES

El alumno deberá desarrollar un sitio WEB utilizando Java Server Pages.

6 ANEXOS

http://java.sun.com
http://www.programacionfacil.com/javajsp/indice.htm
http://www.programacion.com/java/tutorial/servlets_jsp/13/
http://java.sun.com/developer/technicalArticles/javaserverpages/code_convention/
http://archive.coreservlets.com/