



JSP



- Un script JSP termina siendo un Servlet
 - Un servlet que no creamos directamente
 - El contenedor traduce el script en código Java
 - Lo compila en una clase servlet
 - El contenedor ejecuta el mismo ciclo de vida de un servlet para los JSP
- Ejemplo contador.jsp



- Se debe importar el package o usar el nombre completo de la clase
- Directrices
 - Son instrucciones especiales para el contenedor
 - Las utiliza al momento de traducir a código Java
 - Se escriben dentro de `<%@ directriz %>`
 - Importar un package

```
<%@ import="ejemplo.*" %>
```

```
<%@ import="ejemplo.*,otro.*" %>
```



- Expresiones

- Imprime automáticamente cualquier cosa que esté entre los tags

```
<% out.println(Counter.getCount()); %>
```

```
<%=Counter.getCount() %>
```

- Las expresiones terminan siendo el argumento a out.print()
- Nunca se debe terminar la expresión con ;

```
out.print(Counter.getCount());
```



- Todas las variables declaradas dentro de tags script son **locales**
 - Ejemplo: contador2.jsp
 - La variable contador se reinicia cada vez que se ejecuta el método “service”
- Tag para declaraciones
 - Sirve para declarar miembros de una clase servlet
 - Se pueden declarar variables y métodos

```
<%! int count=0; %>
```

- Se necesita el ; ya que no es una expresión
- Agregar ! a ejemplo contador2.jsp



- Que hace el contenedor con los JSP?
 - Busca las directrices, para obtener la información que necesita en la traducción
 - Crea una subclase de HttpServlet
 - que extiende de `org.apache.jasper.runtime.HttpJspBase`
 - Escribe las sentencias “import” al inicio de la clase
 - Utiliza como package: `org.apache.jsp`
 - Si hay declaraciones las escribe antes del método `service`
 - Construye método `service` llamándolo `_jspService()`
 - Recibe un `HttpServletRequest` y `HttpServletResponse`
 - Combina HTML, scriptlets y expresiones en el método `service`



- Ejemplos: test{1,2,3}.jsp
- Se puede sobrescribir los métodos
 - Sobrescribir método `jspInit()`, el cual se invoca apenas comienza el ciclo de vida del servlet
 - Se ejecuta y a la vez queda disponible el `ServletConfig` y `ServletContext`
- Ámbito Servlet – JSP
 - Aplicación: `getServletContext()` - application
 - Requerimiento: request – request
 - Sesión: `request.getSession()` - session
 - Página: – - `pageContext`



- EL: Expression Language
 - Parte oficial de las especificaciones JSP 2.0
 - Permite separar el código Java de JSP
 - Su propósito es ofrecer un camino simple de invocación de código Java
 - EL:
Please contact: `${applicationScope.mail}`
 - JSP Expresión:
Please contact:
`<%=application.getAttribute("mail")%>`



- ¿Como se puede obligar a no dejar código Java en los JSP? (solo usar EL)
 - Se puede usar el TAG <scripting-invalid> en el DD
 - Deshabilita todos los scripting de los JSP indicados
- Se puede deshabilitar EL
 - Se utiliza el TAG <el-ignored> en DD
- Aparte de scriptlets, directrices, declaraciones, expresiones Java y EL, pueden haber **acciones**
 - Existen acciones estándares y no-estándares

```
<jsp:include page="ejemplo.jsp" />
```