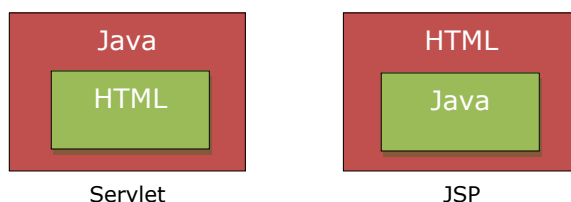
	INGENIERÍA EN INFORMÁTICA – PLAN 2003/2014 PROGRAMACIÓN DISTRIBUIDA Y COMPONENTES – 9º CUATRIMESTRE	
	APUNTE DE JSP	VERSIÓN: 1.0 VIGENCIA: 09-04-2018

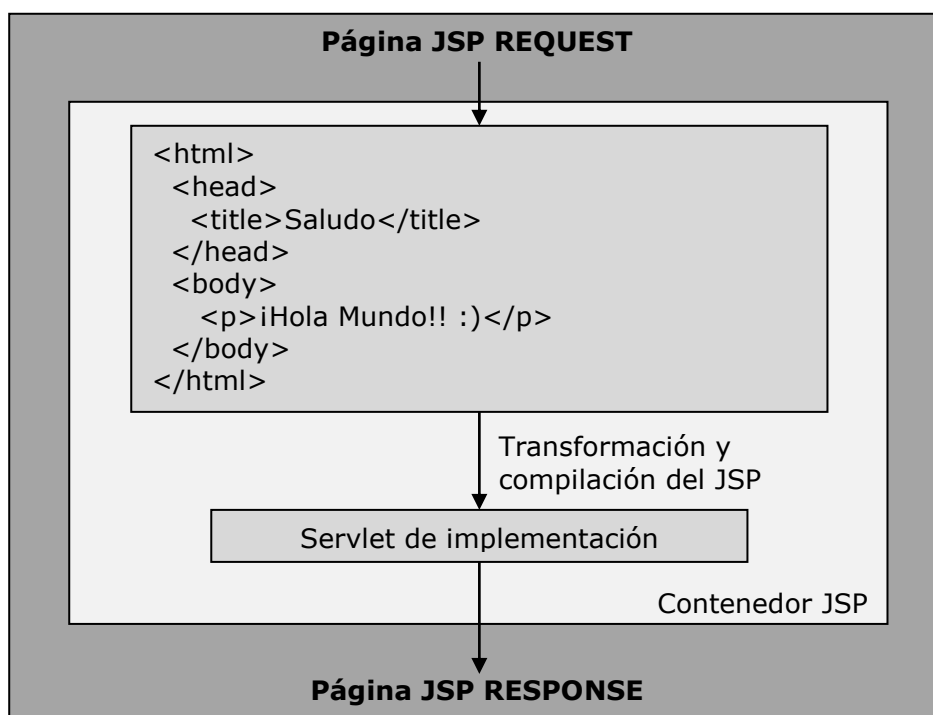
## JSP (Java Server Page)

La tecnología JSP combina HTML, XML y Java Servlet, facilitando la creación de contenido dinámico.

Si programáramos un Servlet y una página JSP que generaran el mismo código HTML, veríamos que en un Servlet el HTML es generado desde el código Java, mientras que en un JSP el código Java es embebido en el código HTML.



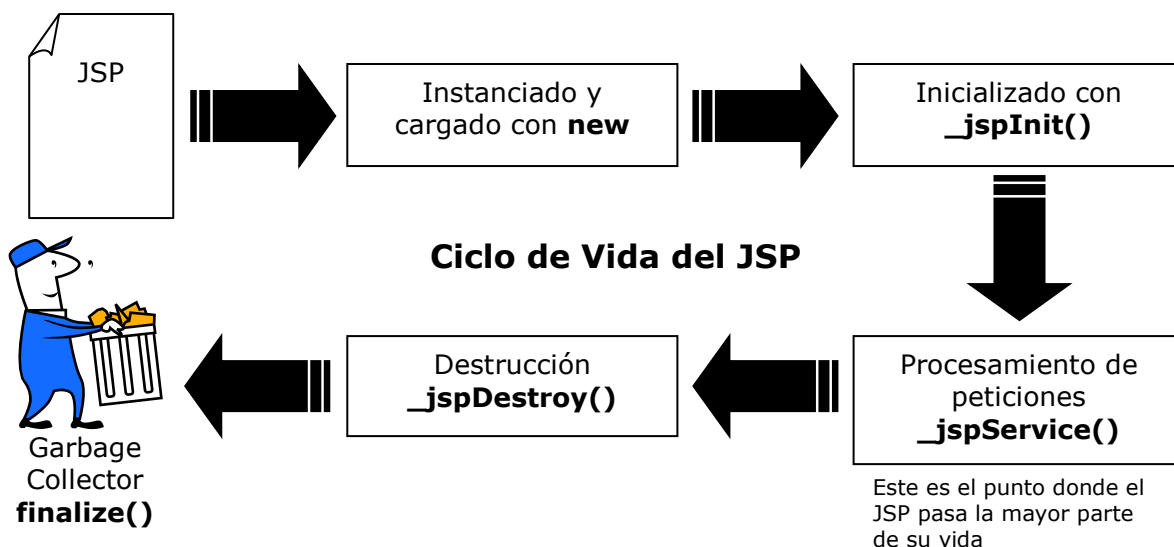
JSP es una extensión de Servlet. La página JSP es implementada a través de la clase `javax.servlet.Servlet` de acuerdo a la especificación Servlet 2.5



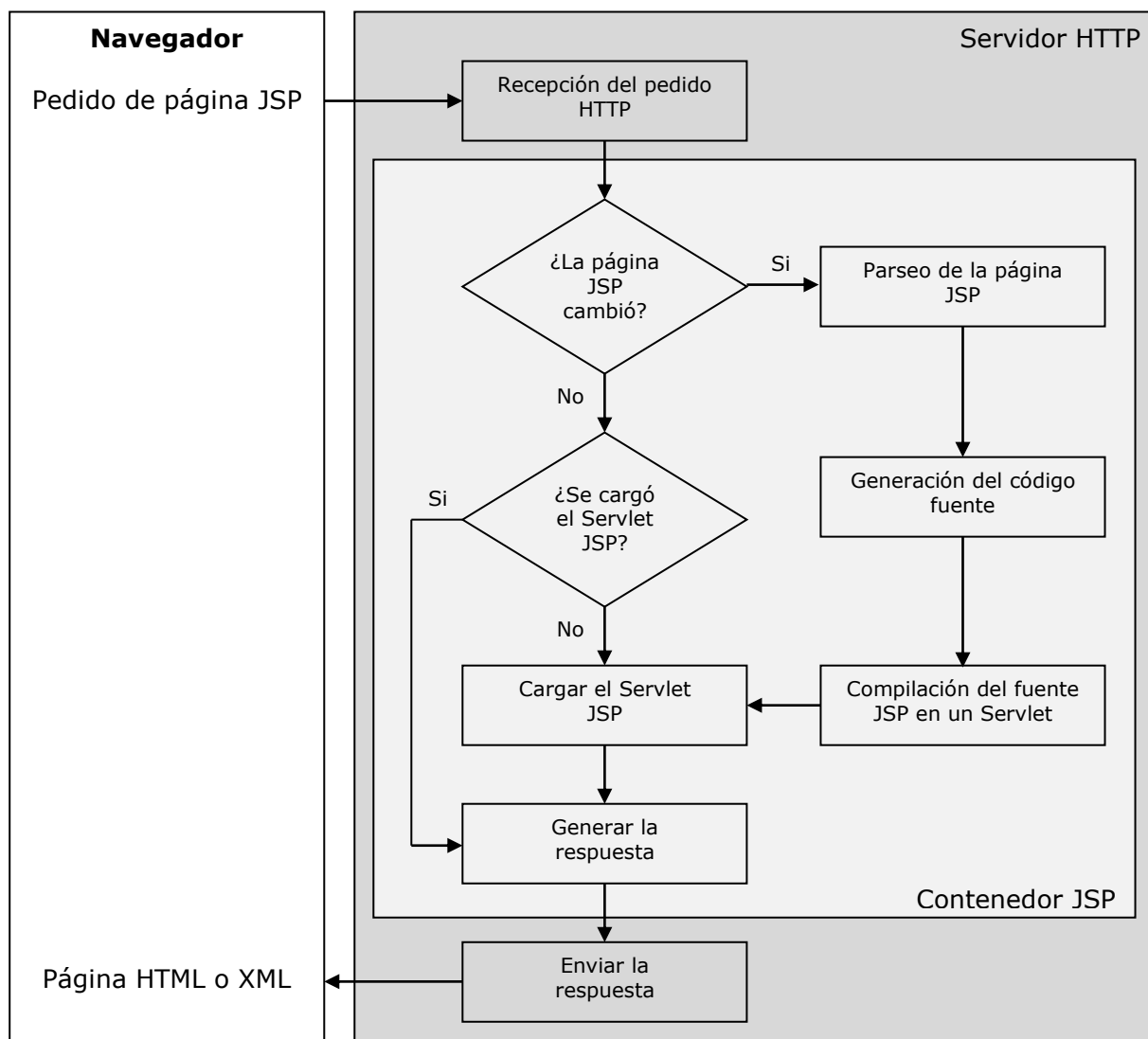
## Ciclo de Vida del JSP

El contenedor JSP es el encargado de regular y controlar el ciclo de vida de un JSP. El ciclo de vida consiste de cuatro fases:

- *Traslación*: Si la sintaxis es correcta se traslada al Servlet de implementación.
- *Inicialización*: El contenedor JSP carga el Servlet de implementación y crea una instancia del Servlet para procesar el pedido. Se invoca al método `_jspInit()`
- *Ejecución*: Después de que el contenedor cargó e inicializó el Servlet de implementación, el pedido es procesado. Para resolver el pedido se llama al método `_jspService()`
- *Finalización*: El contenedor invoca el método `_jspDestroy()`. Después de que este método es invocado el Servlet de implementación no puede responder a ningún otro pedido.



### ¿Cómo trabaja una página JSP?



	INGENIERÍA EN INFORMÁTICA – PLAN 2003/2014 PROGRAMACIÓN DISTRIBUIDA Y COMPONENTES – 9º CUATRIMESTRE	
	APUNTE DE JSP	VERSIÓN: 1.0 VIGENCIA: 09-04-2018

Es común que un proceso JSP se designe como scriptlet, lo cual ayuda a distinguir del concepto de script común.

Habitualmente, un programa JPS se compone de una combinación de etiquetas HTML y de etiquetas JSP. Estas últimas definen el código Java que se ejecutará antes de enviar la salida al navegador. Las etiquetas JSP comienzan con la secuencia **<%** y finalizan con la secuencia **%>**. Hay también una versión con formato XML para las etiquetas JSP: estas etiquetas comienzan con **<jsp:etiqueta>** y finalizan con **</jsp:etiqueta>** y donde la palabra etiqueta se reemplaza con alguna palabra específica. Ambas formas de etiquetas se incluyen o entremezclan dentro del HTML, en forma similar a como lo hacen los códigos de los lenguajes de script.

El código Java contenido dentro de una etiqueta JSP debe ser compilado en algún momento. La compilación se realiza por una aplicación generalmente llamada "motor de JSP" o "intérprete de JSP". Un motor de JSP muy conocido es Apache Tomcat.

Técnicamente, la página JSP es compilada por el motor de JSP sólo la primera vez que esa página es invocada. En ese momento, el motor crea un Servlet a partir de la página, y lo ejecuta. El resultado de la ejecución es enviado al navegador cliente que solicitó la página. De allí en más, cada vez que la página se invoca se ejecuta el Servlet directamente. La página sólo necesita ser recompilada cuando sufra alguna modificación en su contenido.

Las etiquetas JSP pueden ser de cinco tipos diferentes:

- **Comentarios:** Comienzan con **<%--** y finalizan con **--%>**. Contienen comentarios o aclaraciones de texto, que son ignorados por el motor al compilar:

```
<%-- Ejemplo de comentario JSP --%>
```

- **Declaraciones:** Comienzan con **<%!** y finalizan con **%>**. Contienen instrucciones Java que declaran variables, objetos o métodos que luego pueden ser usados en cualquier otro lugar de la página JSP. Notar que las variables declaradas de esta forma, son variables de instancia que se mantendrán cuando la página vuelva a llamarse. Recordar que cuando la JPS se compila por primera vez, el motor crea un objeto Servlet, y toda variable o método declarado en la página mediante la etiqueta **<%!** será un atributo o un método de ese Servlet:

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


- **Directivas (o acciones):** Comienzan con **<%@** y finalizan con **%>**. Le indican al motor de JSP que realice alguna tarea o acción determinada, como puede ser importar un paquete de Java (entre otras acciones). Dentro de esta forma de etiqueta, aparecen parámetros diversos, dependiendo de la acción que se quiere llevar a cabo. Los tres tipos de directivas son las siguientes:

- o **page:** permite configurar la página con valores y elementos que serán útiles al proceso. Por ejemplo,

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

- o **include:** permite insertar otros recursos dentro de la página (tal como código fuente incluido en otro archivo). Por ejemplo,

```
<%@ include file="./docs/ejemplos.html" %>
```

	INGENIERÍA EN INFORMÁTICA – PLAN 2003/2014 PROGRAMACIÓN DISTRIBUIDA Y COMPONENTES – 9º CUATRIMESTRE	
	APUNTE DE JSP	VERSIÓN: 1.0 VIGENCIA: 09-04-2018

- **taglib**: permite que un programador incluya sus propias librerías de tags, para simplificar el código JSP. Por ejemplo,

```
<%@ taglib uri="./WEB-INF/tld/etiquetas.tld" %>
```

- **Expresiones**: Comienzan con **<%=** y finalizan con **%>**. Contienen una expresión, cuyo resultado reemplazará a la etiqueta al procesar la JSP. El valor de la expresión será mostrado en la página HTML que se presente al cliente en el navegador. Sólo las etiquetas que contengan expresiones de este tipo serán transformadas y mostradas al cliente al devolverle la página. Estos elementos son los que permiten que finalmente la página tenga contenidos no fijos, actualizables según los datos de cada cliente:

```
<% int contador = 0; %>
<p>Contador: <%= contador %></p>
```

- **Scriptlets (o procesos)**: Comienzan con **<%** y finalizan con **%>**. Contienen bloques de instrucciones Java que conforman un proceso a realizar en la página:

```
<%
contador++;
out.println("<p>Contador: " + contador + "</p>");
%>
```


Como se dijo, una página JSP se usa normalmente para permitir interacción entre los usuarios y el sistema Web. Los usuarios cargan páginas en las que se les solicita enviar datos, mediante algún formulario HTML consistente en campos de edición, casillas de selección, botones, y otros elementos. Cuando el usuario carga todos los datos pedidos, generalmente se dispone de un botón para enviar esos datos al servidor de la aplicación, y si se presiona ese botón se recogen todos los datos del formulario y se envían al servidor en alguna forma que depende del valor del atributo **method** usado al definir el formulario. Ese valor puede ser "get" o "post" (si no se especifica, el valor default es get):

```
<form method="post" action="response.jsp">
</form>
```

El otro atributo del formulario es **action**, y se usa para indicar cuál es el programa del servidor que recibirá y procesará el envío al presionar el botón de enviar. Normalmente se tratará de una página JSP o de un Servlet. En el código anterior, se supone que ese programa será una página JSP llamada response.jsp.

El programa de respuesta (indicado en el atributo action) debe ser capaz de tomar los valores enviados, procesarlos y generar la página de respuesta. Además de la programación en Java propia de la JSP, toda JSP dispone de una serie de objetos predefinidos y ya creados, listos para usar en diversas tareas específicas. Uno de esos objetos es el objeto **request** (de la clase `HttpServletRequest`), que se usa para acceder a los valores de los campos enviados en la "cadena de petición" o "query string". Dispone de un método `getParameter()` que toma como parámetro una cadena con el nombre del campo que se quiere acceder, y retorna el valor de ese campo en forma de String. Por ejemplo,

```
<%
String nombre = request.getParameter("nombre");
String apellido = request.getParameter("apellido");
%>
```

	INGENIERÍA EN INFORMÁTICA – PLAN 2003/2014 PROGRAMACIÓN DISTRIBUIDA Y COMPONENTES – 9º CUATRIMESTRE	
	APUNTE DE JSP	VERSIÓN: 1.0 VIGENCIA: 09-04-2018

Cuando en el formulario de carga se incluya un campo que permita selección de valores múltiples, el objeto request cuenta con un método alternativo `getParameterValues()`, el cual toma como parámetro el nombre del campo de múltiples valores que se quiere acceder, y retorna un arreglo de cadenas con los valores de ese campo. Por ejemplo,

```
<%
String hobbies[] = request.getParameterValues("hobbies");
%>
```

## Objetos implícitos

Toda página JSP contiene un conjunto de objetos implícitamente creados, y listos para usar en distintas tareas. El objeto **request** es uno de ellos, pero hay algunos más:

Objeto	Significado y uso
application	Este objeto de la clase <code>javax.servlet.ServletContext</code> representa el contenedor en el cual se ejecuta la JSP.
config	Este objeto de la clase <code>javax.servlet.ServletConfig</code> representa la opciones de configuración de la JSP.
exception	Este objeto de la clase <code>java.lang.Throwable</code> representa la excepción que es pasada a la página de error JSP. Sólo está disponible en una página JSP de error.
out	Este objeto de la clase <code>javax.servlet.jsp.JspWriter</code> escribe texto como parte de la respuesta a un requerimiento. Se usa implícitamente con expresiones y acciones JSP que insertan cadenas en una respuesta.
Page	Este objeto de la clase <code>java.lang.Object</code> representa la referencia <code>this</code> para la instancia actual de la JSP.
pageContext	Este objeto de la clase <code>javax.servlet.jsp.PageContext</code> oculta los detalles de implementación del Servlet generado y del contenedor JSP, y provee a los programadores acceso a los objetos implícitos enunciados en esta tabla.
response	Este objeto representa la respuesta al cliente. Normalmente es instancia de una clase que implementa <code>javax.servlet.http.HttpServletResponse</code> . Si se usa un protocolo diferente de HTTP, el objeto es instancia de una clase que implementa <code>javax.servlet.ServletResponse</code> .
request	Este objeto representa la petición del cliente. Normalmente es instancia de una clase que implementa <code>javax.servlet.http.HttpServletRequest</code> . Si se usa un protocolo diferente de HTTP, el objeto es instancia de una subclase de <code>javax.servlet.ServletRequest</code> .
session	Este objeto de la clase <code>javax.servlet.http.HttpSession</code> representa la información de la sesión del cliente si tal sesión ha sido creada.

## Acciones estándar

Como se mencionó anteriormente, además de la etiquetas `<%` y `%>` existe un formato XML de etiquetas JSP, de la forma `<jsp:etiqueta>` y `</jsp:etiqueta>`. Un uso común de estas formas de etiquetas es la introducción de acciones estándar.

Estas acciones proveen a los desarrolladores de una forma flexible de acceder a las tareas más comunes que pueden ser necesarias en un programa JSP, como podría ser incluir recursos, reenviar peticiones a otros recursos o interactuar con JavaBeans. Estas acciones son procesadas en tiempo de petición (cuando la petición se realiza). Las acciones estándar son las siguientes:

Acción	Descripción
<code>&lt;jsp:include&gt;</code>	Incluye dinámicamente otros recursos en una JSP. A medida que la página

	INGENIERÍA EN INFORMÁTICA – PLAN 2003/2014 PROGRAMACIÓN DISTRIBUIDA Y COMPONENTES – 9º CUATRIMESTRE	
	APUNTE DE JSP	VERSIÓN: 1.0 VIGENCIA: 09-04-2018

	se ejecuta, los recursos referidos se incluyen y se procesan.
<jsp:forward>	Redirecciona una petición hacia otra JSP, servlet o página estática. Esta acción finaliza la ejecución de la JSP actual.
<jsp:plugin>	Permite que un componente plugin se añada a la página en forma de objeto específico o elemento HTML embebido.
<jsp:param>	Se usa junto con las acciones include, forward y plugin para especificar pares adicionales de la forma nombre/valor para ser usados por estas acciones.
<jsp:useBean>	Indica que la JSP usa una instancia JavaBean. Especifica el ámbito del bean y le asigna un ID que puede ser usado por componentes que manipulen al bean.
<jsp:setProperty>	Configura una propiedad dentro de la instancia JavaBean especificada.
<jsp:getProperty>	Obtiene una propiedad desde la instancia JavaBean especificada y convierte el resultado a una cadena para ser desplegada en la respuesta al cliente.