# Día 1:

-Web Sockets

Los ciclos request response también pueden ser periódicos. Puede ser disparado por el cliente con AJAX o el servidor usando WebSockets.

Contenedro de servletes = java + servlets + paginas jsp -> Apache Tomcat (es un servidor Web que entiende java)

SA (servidor de aplicaciones) -> entiendo todo lo de contenedor de servlet + JEE (JMS, EJB, etc…)

Para este curso nos sirve con un contenedor de Servlets (un tomcat)

Cualquier aplicación que se ejecuten en un Contenedor de Servlets, se puede ejecutar en un SA, pero no a la inversa.

## Flujo de una aplicación web:

Compilation, packaging (WAR: Web Archive), upload (o major llamado, deploy)

Ahora, el servidor recibe el war. Desempaqueta el war y se pone a analizar buscando: web.xml (características para la app) y los ejecuta, y buscan determinadas anotaciones de java (@) y las ejecutan; todo esto dentro del tiempo de despliegue (no es accesible al usuario final). Cuando el servidor ha desplegado, entonces la hace accesible al usuario.

Finalmente, se puede parar la aplicación web (stop) el usuario no puede acceder pero sige estando en el servidor, pero si haces un undeploy, la retiras del servidor.

Si me encuentro una Java EE 4, corre!!!!!!!!!!!!!

# Día 2:

Lo que hay a la derecha de el url? En una petición GET, tiene que tener como máximo 256 caracteres.

Primer método para la response: response.setContentType("text/html;charset=UTF-8");

En el action del form, se usa el patrón URL, no el nombre del Servlet.

Cuando no se pone un action, el form se envía a la misma página donde está el form.

Para indroducir valores en la sesión y en la request: setAttribute(“clave”,”valor”)

Para obtener: getAttribute(“clave”)

# Día 3:

## JSP:

Ciclo de vida de una página JSP:

Al inicio, el Tomcat **compila** la página JSP (con un compilador de JSP) cuando se llama a ella y la transforma en una **Servlet**.

A continuación el compilador de Java compila la Servlet.

Esto solo se hace 1 vez (la próxima vez que se llama no se vuelva a compilar, ya la tiene)

Si la fecha de modificación es distinta, se vuelve a compilar (permite cambios en caliente).

Casi todos los servidores java, se pueden configurar para precompilar los JSP en tiempo de despliegue.

<%@page Directiva Page

<%= Expresion. Se convierten al final en un string.

<%! Declaracion

<% código embebeido

En una JSP los objetos Java que hemos visto en las Servlets, estan disponibles: **request**, **response**, **session**, **out** (es un PrintWriter).

## Beans

Condiciones:

1. Clase impolementa Serializszable
2. Constructor sin argumentos
3. Firmas de get y set que devuelvan y seteen los atributos

JavaBeans = POJO

Cuando se usa una etiqueta <jsp:useBean, se le está diciendo que crea una instancia de la clase o se reutilizara EN FUNCION DEL AMBITO

El id de un Bean debe ser único

A través de ese id, se identifica al Bean, y se instancia luego para obtener sus propiedades

Ambitos:

page < request < session < application (es como una variable global)

Cuando estamos con formularios, property="\*" hace que metamos todos los datos de un formulario en el bean.

# Día 4:

En la inmensa mayoría de proyectos en la web bajo java es: MVC

Los C son los responsables de mantener actualizados tanto las V como los M.

En Java, tradicionalmente los Controllers son Servlets, y las Vistas son Paginas JSP. Y Los Modelos los Beans…

En MVC puro es cuando una Vista cambia, el Controller actualiza todas las demás.

Para ello se necesita JavaScript con AJAX o similares .

Hoy en día las JSP casi no contienen código Java, sino JSPEL

Si necesitamos código Java en un sitio, una cosa que podemos usar es: taglib:

## JSTL

Java Standard Tag Libraries

Librerias de tags de java que ya están escritas y además son estándar.

Mirar el proyecto UsoJSTLMaven.

Seguridad:

JAAS