Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

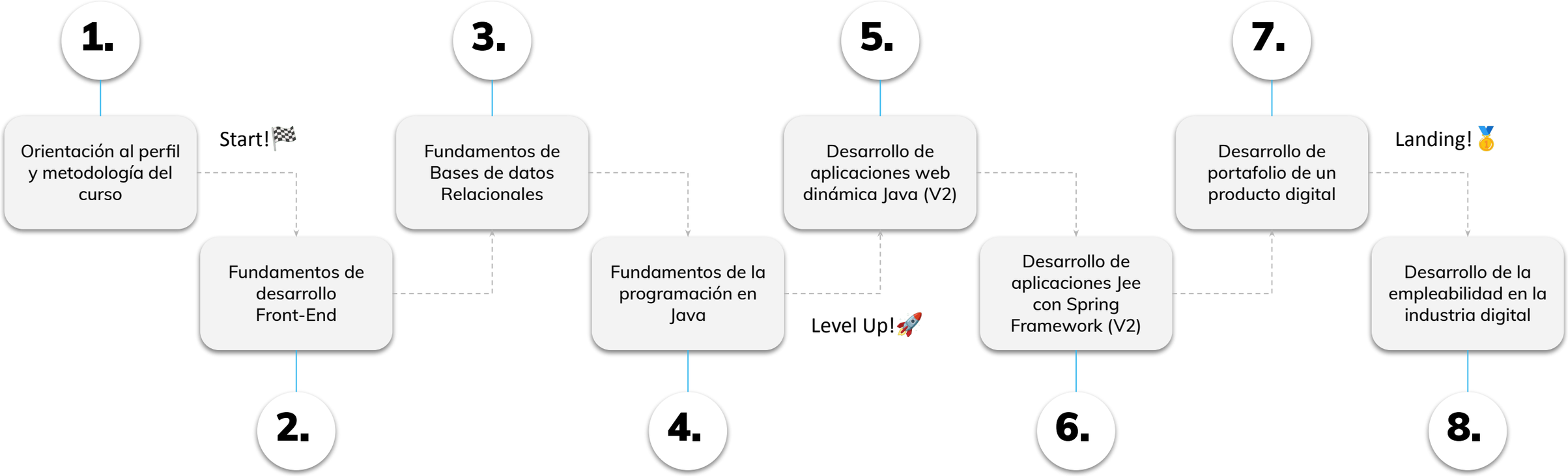
Conociendo los Servlets

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✔ *Creación de formularios con entrada de datos para procesamiento y manipulación.*

English Always | alkemy

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.3 Start!🏁

Conociendo los Servlets

Introducción a los Servlets y
Contenedores Web
Sesiones y cookies

Servlets

Servlets

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Comprender el concepto de Servlets y Contenedores Web*
- *Aprender el manejo de sesiones y cookies*

Rompehielo🥶

Pensemos juntos y respondan en el chat: 🙌

1. ¿Qué tienen en común las siguientes imágenes?
2. ¿Qué tarea cumple cada oficio?

cajero de banco

operador de call center

cocinero

English Always | alkemy

Rompehielo🥶

Pensemos juntos y respondan en el chat: 🙌

1. ¿Dónde se envían los datos que enviamos en las páginas JSP?
2. ¿Quién procesa y devuelve la respuesta?

Haciendo una analogía de lo visto, en esta clase veremos cómo trabajar con Servlets.

Servlets

Servlets

¿Qué es?:

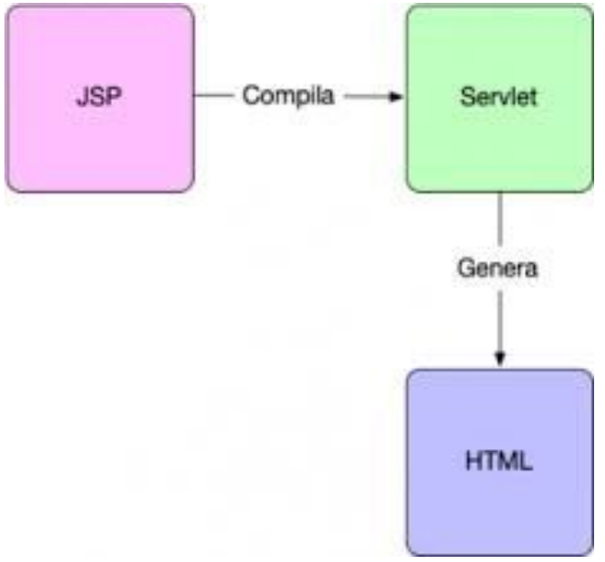
Es un programa Java que **se ejecuta en un servidor Web y construye o sirve páginas web**. De esta forma se pueden construir páginas dinámicas, basadas en diferentes fuentes variables: datos proporcionados por el usuario, fuentes de información variable (páginas de noticias, por ejemplo), o programas que extraigan información de bases de datos.

Los servlets son sencillos de utilizar, más eficientes (se arranca un hilo por cada petición y no un proceso entero), más potente y portable. Con los servlets podremos, entre otras cosas, **procesar, sincronizar y coordinar múltiples peticiones de clientes**, reenviar peticiones a otros servlets u otros servidores, etc.

Servlets

¿Cuál es la función de un Servlet?

1. Tomar la solicitud del cliente y devolver una respuesta.
2. La solicitud adjunta datos, y el código del Servlet sabe como recuperarlos y como usarlos.
3. La respuesta adjunta la información que el navegador necesita para mostrar la página, y el Servlet es quien envía esta información.
4. El Servlet puede decidir enviar la solicitud a otra página, Servlet o JSP.

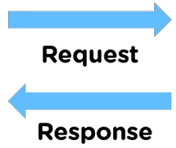


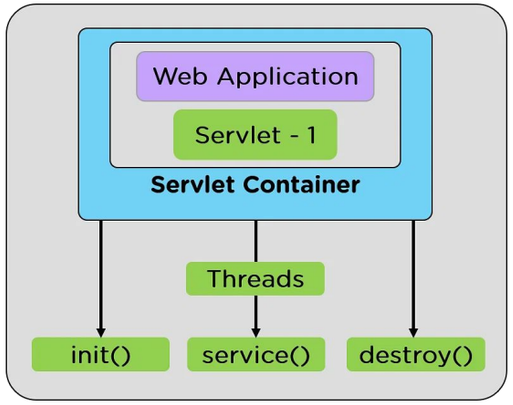
Servlets

Ciclo de vida de un servlet

Todos los servlets tienen el mismo ciclo de vida:

1. **Inicialización**: un servidor carga e inicializa el servlet.
2. **Procesamiento de peticiones**: el servlet procesa cero o más peticiones de clientes (por cada petición se lanza un hilo).
3. **Destrucción**: el servidor destruye el servlet (en un momento dado o cuando se apaga).





Vamos a ver estos pasos en profundidad.

Servlets

1- Inicialización:

Los servlet tienen un inicializador por defecto en el método init().

- public void init() throws ServletException{ ... }
- public void init(ServletConfig conf) throws ServletException{

  super.init(conf);...}

El primer método se utiliza si el servlet no necesita parámetros de configuración externos. El segundo se emplea para tomar dichos parámetros.

Si queremos definir nuestra propia inicialización, deberemos **sobreescribir** alguno de estos métodos. Podemos utilizar la inicialización para establecer una conexión con una base de datos, abrir ficheros, o cualquier tarea que se necesite hacer una sola vez antes de que el servlet comience a funcionar.

Servlets

2. Procesamiento de peticiones

Una vez inicializado, cada petición de usuario lanza un hilo que llama al método service() del servlet.

public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

Este método obtiene el tipo de petición que se ha realizado (GET, POST, PUT, DELETE). Dependiendo del tipo de petición que se tenga, se llama luego a uno de los métodos:

- **doGet():** Para peticiones de tipo GET (aquellas realizadas al escribir una dirección en un navegador, pinchar un enlace o rellenar un formulario que no tenga METHOD=POST)
- **doPost()**: Para peticiones POST (aquellas realizadas al rellenar un formulario que tenga METHOD=POST)

Servlets

3. Destrucción

El método destroy() de los servlets se emplea para eliminar un servlet y sus recursos asociados.

public void destroy() throws ServletException

Aquí debe deshacerse cualquier elemento que se construyó en la inicialización (cerrar conexiones con bases de datos, cerrar ficheros, etc). El servidor llama a destroy() cuando todas las llamadas de servicios del servlet han concluido, o cuando haya pasado un determinado número de segundos (lo que ocurra primero).

Servlets

Para convertir una clase de Java en un Servlet y que funcione como tal, basta con agregar la anotación @WebServlet y luego entre paréntesis la URL a la que responderá este servlet.

Además, la clase debe **extender de HttpServlet**.

```
13 @WebServlet("/productServlet")
14
15 public class ProductServlet extends HttpServlet {
16
```

Servlets

Estructura básica de un servlet:

```java
@WebServlet("/productServlet")

public class ProductServlet extends HttpServlet {


    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

    //... codigo para peticion Get

    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

    //... codigo para peticion Post

    }
}
```

LIVE CODING

Ejemplo en vivo

Creando Servlets:

Vamos a crear un Servlet básico con los métodos vacíos para comenzar a practicar:

1- Crear un servlet WalletServlet que tenga declarados métodos para procesar solicitudes GET y POST.

Tiempo: 15 minutos

English Always | alkemy

Manejando las Sesiones

Manejando las sesiones

Las sesiones contienen información del usuario activo, como su nombre o contraseña.

Una buena alternativa para no manejar información sensible de nuestros usuarios del lado del cliente son las sesiones, que no es otra cosa que una estructura de datos que es accedida exclusivamente del lado del servidor. Si sus sistemas son muy concurridos o están detrás de un balanceador de carga, manejar sesiones podría representar un reto interesante de resolver, sin embargo, la mayoría de las veces funciona muy bien con su configuración por defecto.

Veamos un ejemplo práctico de cómo manipular la sesión en nuestro servlet.

Manejando las sesiones

Método ejemplo:

```
private static final void home(HttpServletRequest request, HttpServletResponse response) {
  HttpSession session = request.getSession();
  session.setAttribute("usuario", "estudiante");
}
```

Hemos creado una sesión y le hemos insertado un registro que describe un nombre de usuario. Luego podemos acceder a esta sesión en un request completamente diferente.

```
private static final void panel(HttpServletRequest request, HttpServletResponse response) {
  HttpSession sesion = request.getSession();
  Object usuario = (String) session.getAttribute("usuario");
}
```

Aquí obtenemos el nombre de usuario. Vemos que, como el **getAttribute** devuelve un objeto, lo casteamos al tipo correcto para su posterior utilización.

Manejando las sesiones

Finalmente si quisiéramos borrar (invalidar) la sesión, podemos hacerlo de la siguiente manera:

```
private static final void salir(HttpServletRequest request, HttpServletResponse response) {
    HttpSession sesion = request.getSession();
    sesion.invalidate();
}
```

El método invalidate() destruye la sesión y sus contenidos.

Manejando las Cookies

Manejando las cookies

Crear una cookie es bastante fácil cuando utilizamos la **clase Cookie** que encontramos en el paquete javax.servlet.http, simplemente hacemos una instancia de esta clase y la mandamos en la respuesta del servidor con el método addCookie().

private static final void **galleta**(HttpServletRequest request, HttpServletResponse response) {

```
  Cookie galletaColor = new Cookie("color", "rojo");
  response.addCookie(galletaColor);
```

}

En este ejemplo podemos ver una cookie simple con todas sus opciones por defecto.

Manejando las cookies

Además, una cookie puede tener los siguientes parámetros:

- **Dominio**: Establece el nombre de dominio (o dominios) para el cual una cookie es válida. Se establece a través del método setDomain(String dominio) de la clase Cookie.

- **Path**: Establece la uri especifica en la que la cookie será válida (e.g: /admin). Es útil si no queremos exponer la cookie en toda la aplicación. Se establece con el método setPath(String path) de la clase Cookie.

- **Expiración**: Indica cuánto debe durar una cookie en el navegador. Este valor se establece en segundos a través del método setMaxAge(int segundos) de la clase Cookie.

Manejando las cookies

- **HostOnly**: Esta propiedad es muy importante en términos de seguridad, ya que establece si una cookie podrá ser leída por el cliente o solo por el servidor. Por eso HostOnly debería ser true siempre que el dominio desde el cual se está intentando leer no sea igual al dominio de origen establecido.

- **Secure**: Otra parámetro muy importante que indica que la cookie en cuestión puede ser leída únicamente a través de un protocolo seguro HTTPS o SSL. Y se establece con el método setSecure(boolean isSecure) de la clase Cookie.

- **HttpOnly**: Es similar a HostOnly excepto que esta si se puede manipular. Funciona como una medida de seguridad extra a la especificación establecida. Java al ser un lenguaje de clase mundial ofrece el método setHttpOnly(boolean isHttpOnly) para estos efectos.

Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro....

Trabajo Integrador del Módulo 💪🏽

Iremos completándolo progresivamente clase a clase.

English Always | alkemy

Ejercicio N° 1

Servlet

Servlet

Manos a la obra: 🙌

Debes crear la estructura de un Servlet para poder registrar un usuario. En este caso, es necesario únicamente crear la clase ‘UserServlet’, y la declaración (no implementación) de los métodos doGet y doPost.

Consigna: ✍️

1- En el proyecto AlkeWallet, crear el servlet UserServlet

2- Agregar la anotación necesaria, mapeando la URL (“/userServlet”)

3- Crear los métodos doGet y doPost.

Tiempo🕐: *20 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender la configuración e implementación de los Servlets*
- ✓ *Conocer el manejo de sesiones y de cookies*

English Always | alkemy

#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 👇👇👇

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
    a. *Lectura Modulo 5, Lección 3: páginas 1 - 10*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 👋🏻

English Always | alkemy

Momento:

Time-out!

⏰5 min.