Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

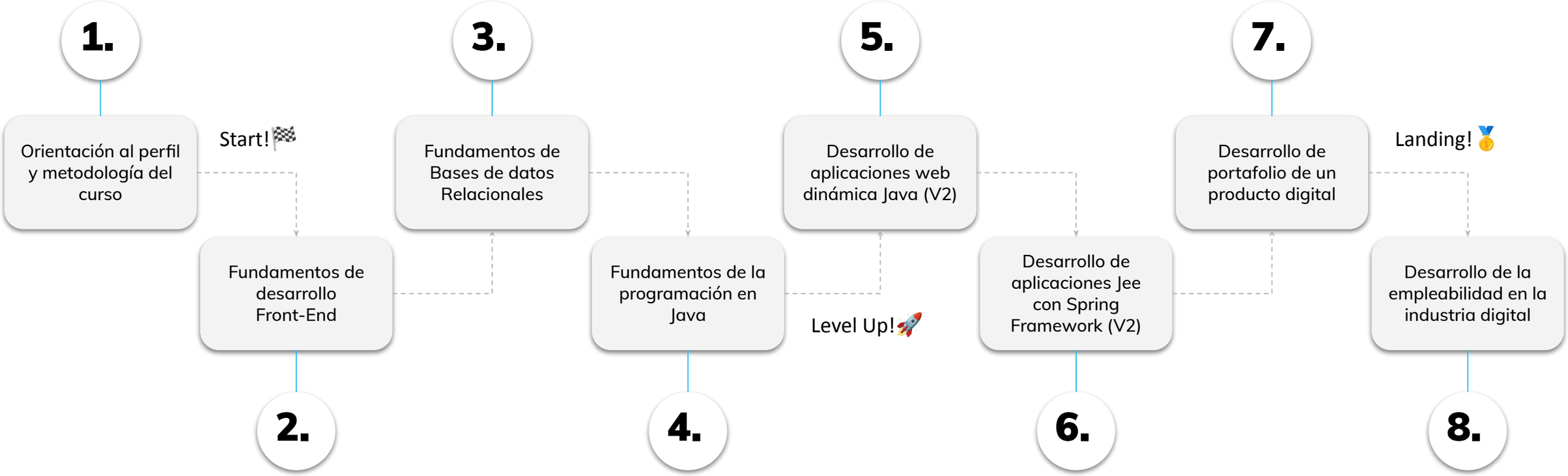
Conociendo los Servlets

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 y configuración de Servlets*
- ✓ *Manejo de sesiones y cookies*

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.3 Start! 🏁

Conociendo los Servlets

Compartiendo información entre Servlets

Dialogando

Compartiendo información
entre Servlets
Concurrencia con los Servlets

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Aprender a compartir información entre servlets*
- *Comprender el concepto de concurrencia*

Compartiendo información entre Servlets

Compartiendo información

En general, **compartir información entre servlets puede ser una forma eficaz de mejorar la funcionalidad, el rendimiento y la seguridad de su aplicación.**

Algunas razones por las que es útil compartir información entre servlets:

- **Para simplificar el código**: se puede evitar tener que pasar la información repetidamente entre los servlets. Esto puede simplificar el código y hacerlo más fácil de mantener.
- **Para mejorar el rendimiento**: se puede evitar tener que volver a cargar la información de la base de datos o de otros recursos. Esto puede mejorar el rendimiento de la aplicación.
- **Para aumentar la seguridad**: se puede restringir el acceso a la información a los usuarios autorizados. Esto puede ayudar a proteger su información contra el acceso no autorizado.

Compartiendo información

Algunas **formas de compartir información entre servlets** J2EE:

- **Atributos del contexto**: Los atributos del contexto de aplicación son compartidos por todos los servlets y JSP que se ejecutan en la misma aplicación en el servidor.

  Para agregar un atributo al contexto, se utiliza el método setAttribute() en el objeto ServletContext. Para obtener un atributo del contexto, se utiliza el método getAttribute().

```
// En el primer servlet
String dato = "Hola desde el primer servlet";
ServletContext context = getServletContext();
context.setAttribute("datoCompartido", dato);
```

```
// En el segundo servlet (segundoServlet)
ServletContext context = getServletContext();
String dato = (String) context.getAttribute("datoCompartido");
```

Compartiendo información

- **Sesiones**: los atributos de sesión son específicos para cada usuario y permanecen disponibles durante toda la sesión, incluso si el usuario navega entre diferentes páginas o servlets.

  Para crear una sesión, se utiliza el método getSession() en el objeto HttpServletRequest. Para agregar información a una sesión, se utiliza el método setAttribute() en el objeto HttpSession. Para obtener información de una sesión, se utiliza el método getAttribute().

```
// En el primer servlet
String dato = "Hola desde el primer servlet";
HttpSession session = request.getSession();
session.setAttribute("datoCompartido", dato);
```

```
// En el segundo servlet (segundoServlet)
HttpSession session = request.getSession();
String dato = (String) session.getAttribute("datoCompartido");
```

Compartiendo información

- **Atributos de solicitud:** Cuando un servlet envía una solicitud a otro servlet o JSP, puede establecer atributos en la solicitud que serán accesibles por el servlet de destino.

```
// En el primer servlet
String dato = "Hola desde el primer servlet";
request.setAttribute("datoCompartido", dato);
RequestDispatcher dispatcher = request.getRequestDispatcher("/segundoServlet"
dispatcher.forward(request, response);
```

```
// En el segundo servlet (segundoServlet)
String dato = (String) request.getAttribute("datoCompartido");
```

LIVE CODING

Ejemplo en vivo

Creando Servlets:

Vamos a crear un Servlet básico con los métodos vacíos para comenzar a compartir información entre servlets :

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

2- Comunicar el nuevo servlet con UserServlet y compartir el nombre de usuario a través de atributos del contexto, sesiones, atributos de solicitud.

Tiempo: 25 minutos

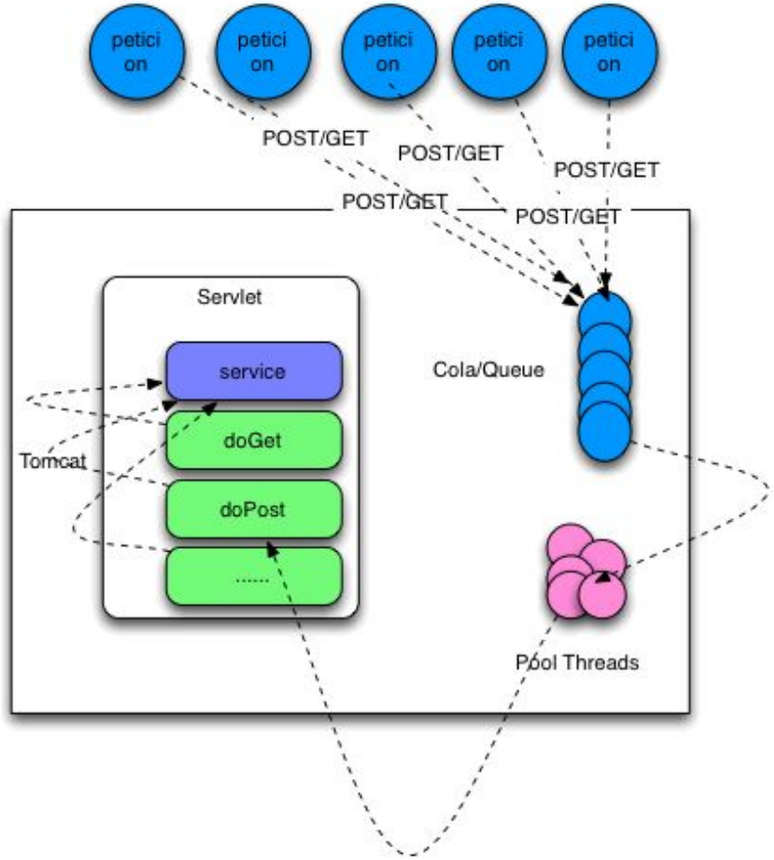
Concurrencia entre Servlets

Concurrencia entre Servlets

¿Qué es?

La concurrencia se refiere a la ejecución simultánea de múltiples hilos o procesos que intentan acceder y modificar recursos compartidos.

Es un tema importante a tener en cuenta cuando trabajas con aplicaciones web, especialmente cuando varios hilos pueden interactuar con los mismos servlets al mismo tiempo.



Concurrencia entre Servlets

Se puede presentar en varias situaciones:

1. **Solicitudes concurrentes**: Cuando varios clientes (navegadores u otras aplicaciones) envían solicitudes HTTP al mismo servlet al mismo tiempo. Cada solicitud se maneja en un hilo separado en el servidor, lo que significa que múltiples hilos pueden ejecutar el mismo servlet simultáneamente.

2. **Recursos compartidos**: Si varios servlets comparten recursos globales, como variables estáticas o atributos de contexto de aplicación, podría haber problemas de concurrencia cuando varios hilos intentan acceder y modificar esos recursos compartidos al mismo tiempo.

3. **Sesiones**: Cuando varios usuarios interactúan con la misma aplicación web y utilizan sesiones para mantener información específica del usuario, es posible que varios hilos intenten acceder y modificar la misma sesión al mismo tiempo.

Concurrencia entre Servlets

Algunas estrategias para manejar la concurrencia en servlets incluyen:

- **Sincronización**: Utiliza bloqueos (`synchronized`) para garantizar que solo un hilo pueda acceder a recursos compartidos en un momento dado.
- **Evitar recursos compartidos**: Diseña tu aplicación para evitar el uso excesivo de recursos compartidos (como variables globales).
- **Uso de sesiones seguras:** Asegúrate de que las sesiones sean seguras para ser accedidas y modificadas por múltiples hilos al mismo tiempo. Utiliza las sesiones correctamente y evita la posibilidad de pérdida de datos o inconsistencias.
- **Pruebas y análisis**: Realiza prueba para identificar posibles problemas de concurrencia. Utiliza herramientas de análisis de concurrencia y asegúrate de que tu aplicación funcione de manera correcta y predecible en entornos concurrentes.

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

Dialogando

Dialogando

Manos a la obra: 🙌

En este caso, es necesario crear la clase ‘CurrencyServlet’, y la declaración (no implementación) de los métodos doGet y doPost.

Consigna: ✍️

1- En el proyecto AlkeWallet, crear un servlet llamado CurrencyServlet.

2- Compartir la información del nombre de usuario de UserServlet con el CurrencyServlet a través del uso de Sesiones y de atributos de solicitud.

Tiempo🕐: *30 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ ***Comprender cómo compartir información entre servlets***
- ✓ ***Conocer el concepto de concurrencia***

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 10 - 13*
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.