Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

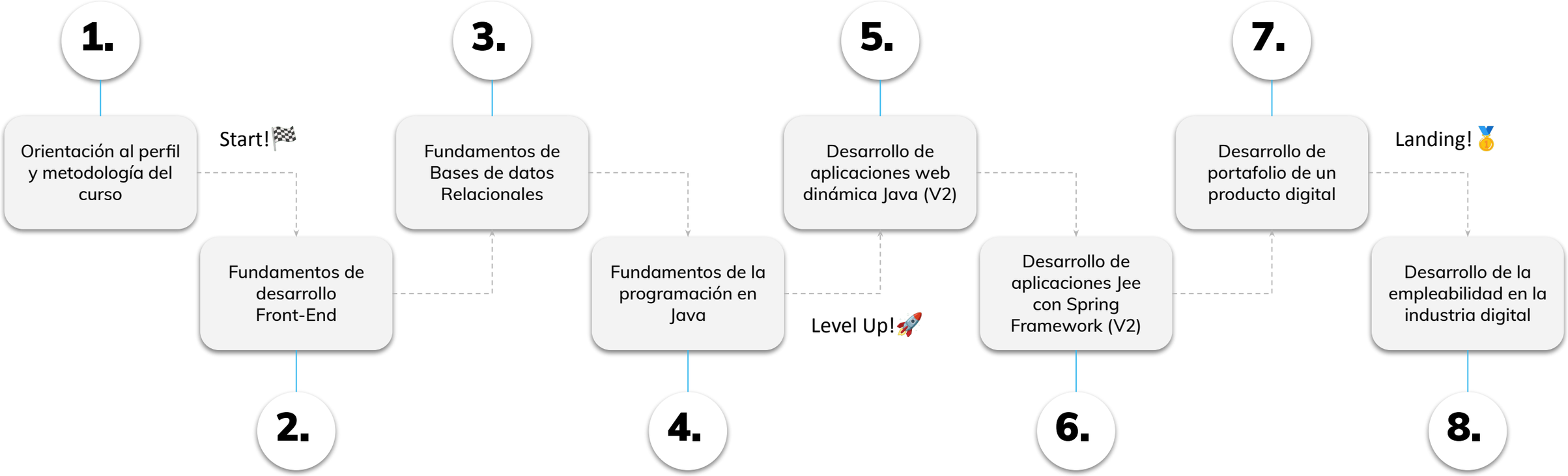
Conociendo los Servlets

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 📚:

- ✓ *Compartiendo información entre servlets*
- ✓ *Concurrencia entre servlets*

English Always | alkemy

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.3 Start! 🏁

Conociendo los Servlets

Controlando parámetros de un GET request

GET request

GET request

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

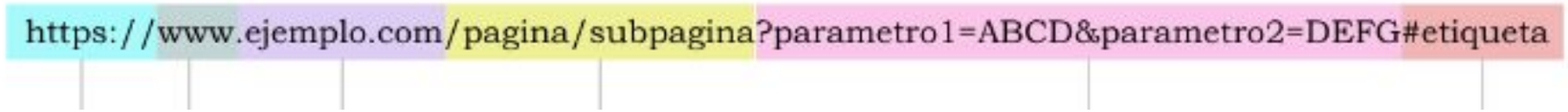
- ***Aprender a controlar parámetros de un GET request***

Rompehielo🥶

¿Qué significa cada parte de una URL?: 🙌

Respondan en el chat o levantando la mano: ✍️

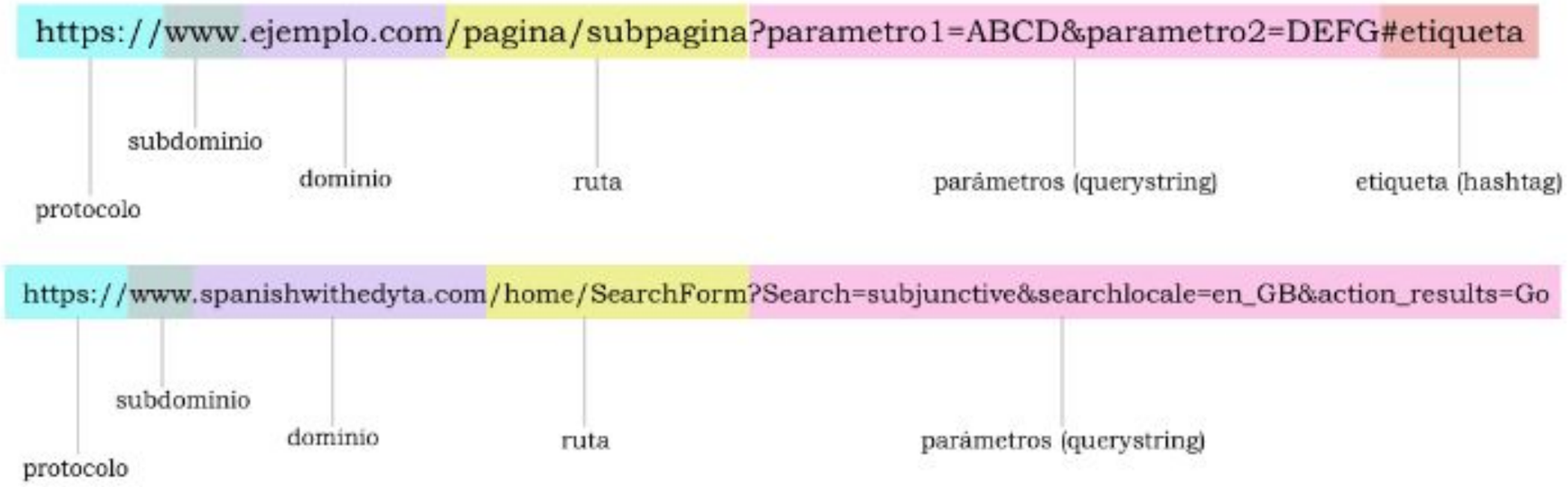
¿Conocen que tipo de información podemos ver en una URL?



Rompehielo🥶

¿Qué significa cada parte de una URL?: 🙌

¡En las URL viajan diferentes valores que se envían a través de peticiones GET!



Controlando parámetros de un GET request

Controlando parámetros de un GET request

Si llamamos un servlet desde un formulario HTML, podremos hacerlo de dos formas: GET y POST. Con la primera los parámetros del formulario están incluidos la url que se utiliza para invocar el servlet y en el segundo caso los parámetros se almacenan en un buffer especial del servidor.

Para procesar el primer tipo de peticiones (GET) está el método doGet. La implementación por defecto del método service es capaz de determinar el tipo de petición HTTP que en un momento dado recibe el servlet. Una vez identificada llama o al método doGet o al doPost según el caso.

Controlando parámetros de un GET request

Para controlar los parámetros en un GET request:

- Primero debemos obtener el valor de los parámetros en el servlet en el método doGet() utilizando el objeto HttpServletRequest.
- Podemos acceder a los parámetros mediante los métodos getParameter() o getParameterValues().
- Luego sólo resta procesar y validar los parámetros según sea necesario para cada aplicación.

Controlando parámetros de un GET request

Veamos un ejemplo de cómo controlar los parámetros de una solicitud GET en un servlet:

Supongamos que tienes un servlet llamado MainServlet que maneja una solicitud GET con dos parámetros, "nombre" y "edad". El servlet mostrará estos parámetros en la respuesta al cliente.

En este caso, es como escribir código HTML en nuestro archivo Java.

```java
@WebServlet("/mainServlet")
public class MainServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Obtenemos el valor del parámetro "nombre" de la solicitud GET
        String nombre = request.getParameter("nombre");

        // Obtenemos el valor del parámetro "edad" de la solicitud GET
        String edad = request.getParameter("edad");

        // Procesamos y validamos los parámetros según sea necesario
        // Por ejemplo, podríamos verificar si los parámetros son nulos o vacíos

        // Creamos la respuesta para el cliente
        response.setContentType("text/html;charset=UTF-8");
        response.getWriter().println("<html><body>");
        response.getWriter().println("<h1>Información recibida:</h1>");
        response.getWriter().println("<p>Nombre: " + nombre + "</p>");
        response.getWriter().println("<p>Edad: " + edad + "</p>");
        response.getWriter().println("</body></html>");
    }
}
```

Controlando parámetros de un GET request

Este método recibe los parámetros dados por el cliente a través de la clase HttpServletRequest y encapsula la respuesta que se le dará al cliente a través de la clase HttpServletResponse.

El servlet puede retornar al cliente cualquier tipo de información, desde texto plano hasta un ejecutable, por lo que es necesario señalar inicialmente qué tipo de respuesta se dará a través del método setContentType.

```java
@WebServlet("/mainServlet")
public class MainServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Obtenemos el valor del parámetro "nombre" de la solicitud GET
        String nombre = request.getParameter("nombre");

        // Obtenemos el valor del parámetro "edad" de la solicitud GET
        String edad = request.getParameter("edad");

        // Procesamos y validamos los parámetros según sea necesario
        // Por ejemplo, podríamos verificar si los parámetros son nulos o vacíos

        // Creamos la respuesta para el cliente
        response.setContentType("text/html;charset=UTF-8");
        response.getWriter().println("<html><body>");
        response.getWriter().println("<h1>Información recibida:</h1>");
        response.getWriter().println("<p>Nombre: " + nombre + "</p>");
        response.getWriter().println("<p>Edad: " + edad + "</p>");
        response.getWriter().println("</body></html>");
    }
}
```

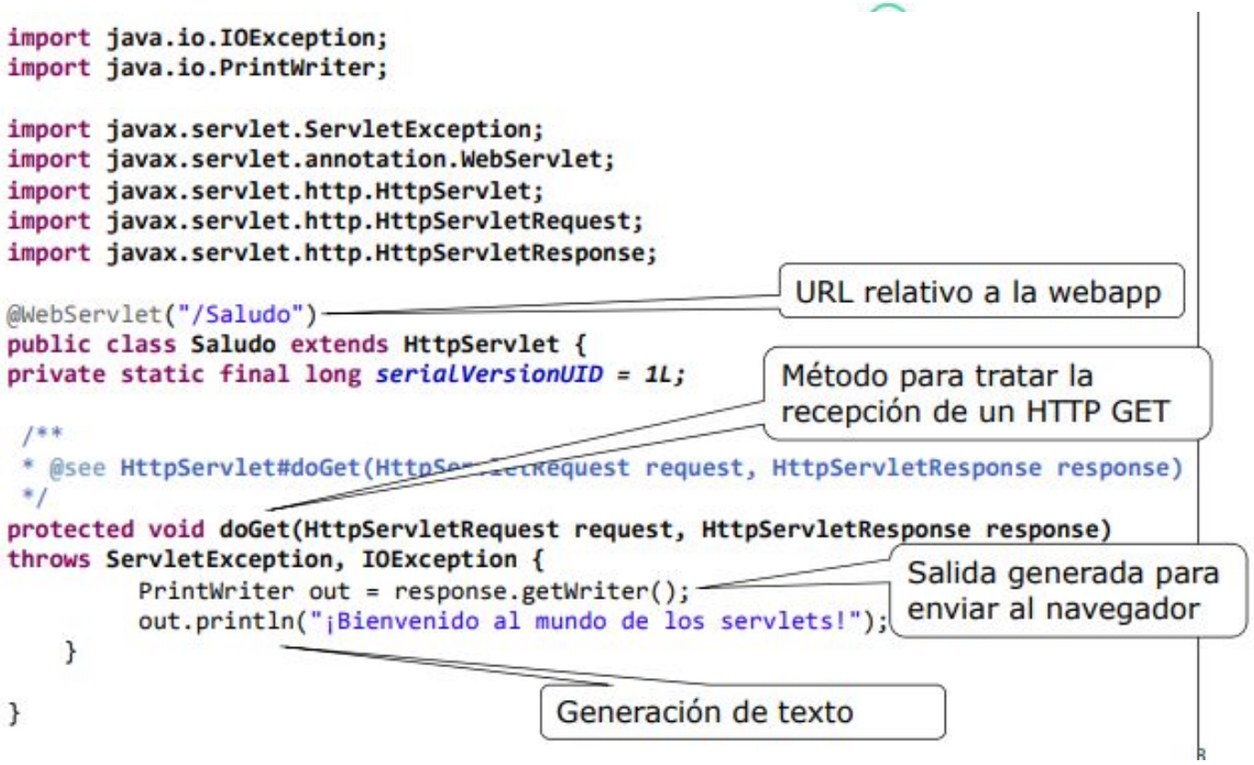
Controlando parámetros de un GET request

Entonces, el objeto request tiene información enviada en la petición.

El objeto response permite crear la respuesta

- setContentType(): permite indicar el tipo MIME de lo que se va a generar (por ejemplo, "text/html")

- getWriter(): devuelve el PrintWriter donde escribir lo que se genere



Controlando parámetros de un GET request

El método doGet() envía los parámetros de forma explícita junto a la página, mostrando en la barra de navegación los parámetros y sus valores. Son esas largas cadenas que aparecen en algunas páginas en nuestra barra de navegación, luego del llamado al servlet,por ejemplo:

/buscar?id=1806&valor=0987&texto=todo&...

Las cadenas de la URL tienen la siguiente forma de referencia:

parametro1=valor1&parametro2=valor2&....&parametroN=valorN.

Es decir es una concatenación de pares **parámetro-valor** a través del operador '&'.

Controlando parámetros de un GET request

Las peticiones HTTP GET tienen un **límite en la cantidad de caracteres que pueden aceptar en el URL**. Si se envían los datos de un formulario muy extenso mediante HTTP GET pueden producirse errores por este motivo, por lo que habría que utilizar el método HTTP POST.

Se suele decir que el método GET es seguro e idempotente:

Seguro, porque no tiene ningún efecto secundario del cual pueda considerarse al usuario responsable del mismo. Es decir, por ejemplo, una llamada del método GET no debe ser capaz en teoría de alterar una base de datos. GET debería servir únicamente para obtener información.
Idempotente, porque puede ser llamado tantas veces como se quiera de una forma segura.

Es como si GET fuera algo así como **ver pero no tocar.**

LIVE CODING

Ejemplo en vivo

Primer método doGet():

Vamos a crear un Servlet básico con los métodos vacíos para comenzar a implementar un método doGet :

1- Crear un servlet CurrencyServlet con el método para procesar solicitudes GET que reciba un importe y el tipo de moneda a extraer.

2- Se deben mostrar los datos del usuario, el importe y el tipo de moneda seleccionado.

Tiempo: 30 minutos

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

GET request

GET request

Manos a la obra: 🙌

Debes crear la estructura de un Servlet para poder registrar un usuario. En la clase ‘UserServlet’, implementar el método doGet.

Consigna: ✍️

1- En el proyecto AlkeWallet, trabajar sobre un servlet llamado UserServlet

2- Crear un formulario de login en doGet() y responder un saludo personalizado con el nombre de usuario que viaja en el formulario.

Tiempo🕐: *30 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ ***Comprender cómo controlar los parámetros de un GET request***

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 14 - 15*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

Momento:

Time-out!

⏰5 min.