Clase grabada

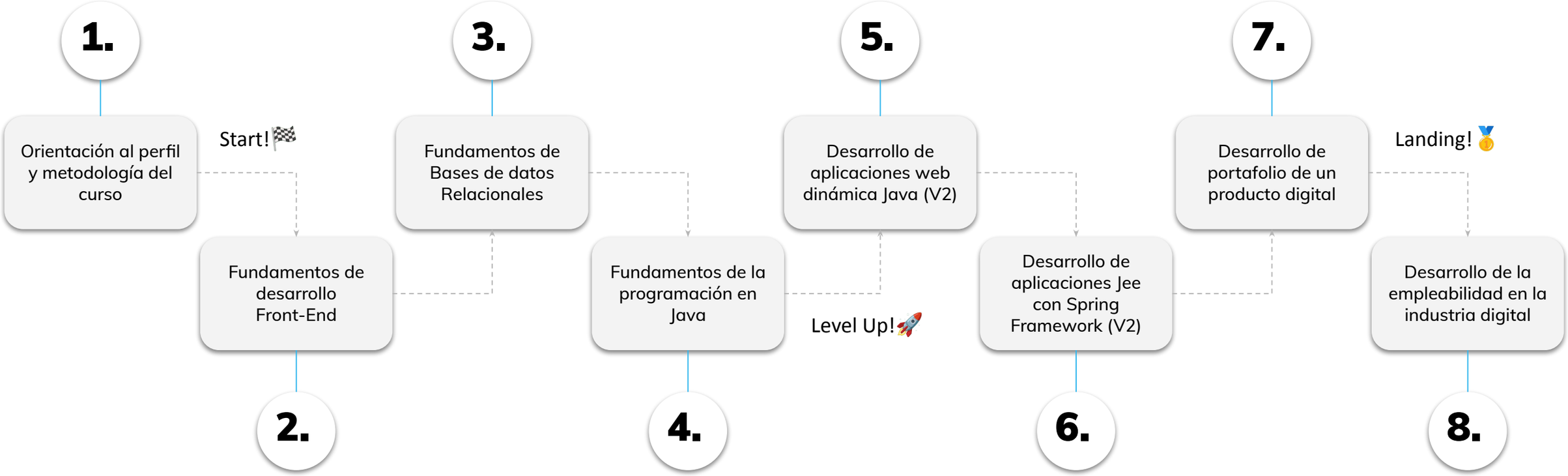
Acceso a datos en Spring Framework

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 📚:

- ✓ *Test y empaquetamiento*
- ✓ *Empaquetamiento WAR de una aplicación Spring*

English Always | alkemy

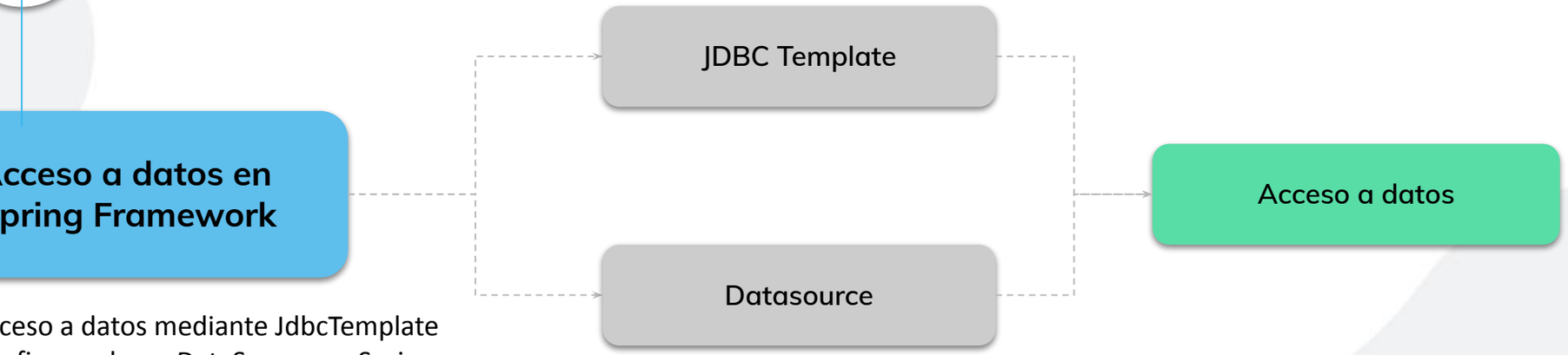
LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

6.3 Start! 🏁

Acceso a datos en Spring Framework

Acceso a datos mediante JdbcTemplate
Configurando un DataSource en Spring
Utilizando JdbcTemplate de Spring para el acceso a datos



English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Comprender el proceso de acceso a datos con JDBC Template en Spring*
- *Aprender a configurar un datasource en Spring*

Acceso a datos mediante JDBC Template

JDBC Template

JdbcTemplate es una abstracción de Spring que simplifica la interacción con bases de datos relacionales mediante **JDBC** (Java Database Connectivity). JDBC proporciona una interfaz estándar para interactuar con bases de datos desde Java.

La clase **JdbcTemplate** ejecuta consultas SQL, itera sobre el ResultSet y recupera los valores llamados, actualiza las instrucciones y las llamadas a procedimientos, "captura" las excepciones y las traduce a las excepciones definidas en el paquete org.springframwork.dao.

JDBC Template

Características Principales de JdbcTemplate:

Sencillez y Productividad: reduce la complejidad de la escritura de código JDBC manual al manejar tareas comunes, como la apertura y cierre de conexiones, el manejo de excepciones y la creación de declaraciones SQL.

Manejo de Excepciones Automático: maneja automáticamente la gestión de excepciones JDBC, convirtiéndolas en excepciones de la jerarquía de Spring DataAccessException.

Callback y Mapeo de Resultados: proporciona capacidades de mapeo de resultados que convierten automáticamente los resultados de la consulta en objetos Java.

Uso de RowMapper y ResultSetExtractor: RowMapper es una interfaz funcional que permite mapear filas de un ResultSet a objetos Java, mientras que ResultSetExtractor es una interfaz para extraer información del ResultSet.

Configurando un DataSource en Spring

Configurando un DataSource

Antes de utilizar JdbcTemplate, debes configurar un **DataSource** en tu aplicación Spring.

Un DataSource proporciona la conexión a la base de datos. Puedes configurarlo en el archivo de configuración de Spring, ya sea utilizando el XML o las anotaciones preestablecidas.

Configurar un DataSource en Spring es esencial para permitir el acceso a la base de datos desde tu aplicación.

Configurando un DataSource

Configuración del Proyecto:

Primero, asegúrate de tener las dependencias necesarias en tu archivo pom.xml:

```
<dependencies>
    <!-- Otras dependencias de Spring -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

Configurando un DataSource

2. Configurar el DataSource en el archivo application.properties o application.yml:

```
spring.datasource.url=jdbc:mysql://localhost:3306/nombre_de_tu_bd
spring.datasource.username=usuario
spring.datasource.password=contraseña
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

Configurando un DataSource

3 - Configuración XML: Si estás utilizando una configuración XML en tu aplicación, puedes configurar un DataSource mediante elementos <bean> en el archivo de configuración de Spring. Por ejemplo:

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/mydatabase" />
    <property name="username" value="username" />
    <property name="password" value="password" />
</bean>
```

Configurando un DataSource

4 - Configuración mediante Anotaciones: Esta clase de configuración utiliza la anotación @Configuration para indicar que es una configuración de Spring y define dos beans: uno para el DataSource y otro para el DataSourceTransactionManager, que se utiliza para administrar transacciones.

```java
@Configuration
public class ConfiguracionBD {
    @Bean
        public DataSource dataSource() {
            DriverManagerDataSource dataSource = new DriverManagerDataSource();
            dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
            dataSource.setUrl("jdbc:mysql://localhost:3306/tu_basedatos");
            dataSource.setUsername("usuario");
            dataSource.setPassword("contraseña");
            return dataSource;
        }

        @Bean
        public DataSourceTransactionManager transactionManager() {
            return new DataSourceTransactionManager(dataSource());
        }
}
```

Configurando un DataSource

Ambos métodos son válidos, y la elección entre ellos depende de las preferencias y requisitos de tu aplicación. La configuración mediante anotaciones es más común en aplicaciones Spring modernas debido a su simplicidad y legibilidad. En cualquier caso, el DataSource es un componente clave para establecer la conexión con una base de datos, y configurarlo adecuadamente es esencial para el correcto funcionamiento de tu aplicación.

```java
@Configuration
public class ConfiguracionBD {
    @Bean
        public DataSource dataSource() {
            DriverManagerDataSource dataSource = new DriverManagerDataSource();
            dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
            dataSource.setUrl("jdbc:mysql://localhost:3306/tu_basedatos");
            dataSource.setUsername("usuario");
            dataSource.setPassword("contraseña");
            return dataSource;
        }

        @Bean
        public DataSourceTransactionManager transactionManager() {
            return new DataSourceTransactionManager(dataSource());
        }
}
```

Utilizando JDBC Template para acceso a datos

JDBC Template para acceso a datos

Paso 1: Configurar el DataSource en tu aplicación

Configura el DataSource en tu aplicación. Puedes hacerlo utilizando anotaciones en una clase de configuración o mediante un archivo de configuración XML como aprendimos anteriormente.

```java
@Configuration
public class ConfiguracionBD {
    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
        dataSource.setUrl("jdbc:mysql://localhost:3306/tu_basedatos");
        dataSource.setUsername("usuario");
        dataSource.setPassword("contraseña");
        return dataSource;
    }

    @Bean
    public DataSourceTransactionManager transactionManager() {
        return new DataSourceTransactionManager(dataSource());
    }
}
```

JDBC Template para acceso a datos

Paso 2: Crear un Componente que Utilice JdbcTemplate

Puedes inyectar JdbcTemplate en tus componentes y utilizarlo para interactuar con la base de datos. Veamos un ejemplo de un servicio que utiliza JdbcTemplate para realizar operaciones de base de datos:

En este ejemplo, el servicio MiServicio utiliza JdbcTemplate para realizar una consulta que obtiene todos los usuarios de la base de datos y otra operación que inserta un nuevo usuario.

```java
@Service
public class MiServicio {

    private final JdbcTemplate jdbcTemplate;

    @Autowired
    public MiServicio(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public List<Usuario> obtenerTodosLosUsuarios() {
        String sql = "SELECT * FROM usuarios";
        return jdbcTemplate.query(sql, (rs, rowNum) ->
                new Usuario(
                        rs.getString("id"),
                        rs.getString("nombre"),
                        rs.getString("email")
                )
        );
    }

    public void insertarUsuario(Usuario usuario) {
        String sql = "INSERT INTO usuarios (nombre, email) VALUES (?, ?)";
        jdbcTemplate.update(sql, usuario.getNombre(), usuario.getEmail());
    }
}
```

JDBC Template para acceso a datos

Paso 3: Utilizar el Servicio en un Controlador o en Otros Componentes:
Este controlador expone dos endpoints, uno para obtener todos los usuarios y otro para insertar un nuevo usuario.

```java
@RequestMapping("/usuarios")
public class MiControlador {
    @Autowired
    private final MiServicio miServicio;

    public MiControlador(MiServicio miServicio) {
        this.miServicio = miServicio;
    }

    @GetMapping
    public List<Usuario> obtenerTodosLosUsuarios() {
        return miServicio.obtenerTodosLosUsuarios();
    }

    @PostMapping
    public void insertarUsuario(@RequestBody Usuario usuario) {
        miServicio.insertarUsuario(usuario);
    }

}
```

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

LIVE CODING

Ejemplo en vivo

Completando la Wallet:

- *Es hora de completar nuestro proyecto de billetera virtual. Para esto, vamos a agregar la clase Cuenta que contendrá la información de una cuenta bancaria: id:numeroCuenta, saldoActual, tipoMoneda.*
- *Además, vamos a crear la clase de Servicio para Cuenta que nos permitirá realizar operaciones como: consultarSaldo, cambiarMoneda, extraerDinero, depositarDinero, etc.*

Tiempo: 30 minutos

LIVE CODING

Ejemplo en vivo

Completando la Wallet:

- *Luego, crearemos una clase de **configuración** para poder definir el datasource. Recuerda crear la base de datos con anterioridad.*

Tiempo: 30 minutos

English Always | alkemy

¿Alguna consulta?

Ejercicio

Acceso a Datos

Acceso a Datos

Contexto: 🙌

Vamos a seleccionar un proyecto actual para configurar su datasource. Puedes continuar con el ejercicio de manera asincrónica

Consigna: ✍️

Configurar el datasource del proyecto en el archivo application.properties,

Tiempo🕐: *15 minutos*

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender la implementación de JDBC Template para acceso a datos*
- ✓ *Aprender a configurar un datasource en Spring*

#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 6, Lección 3: páginas 1 - 5*
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