Spring Boot Data JPA

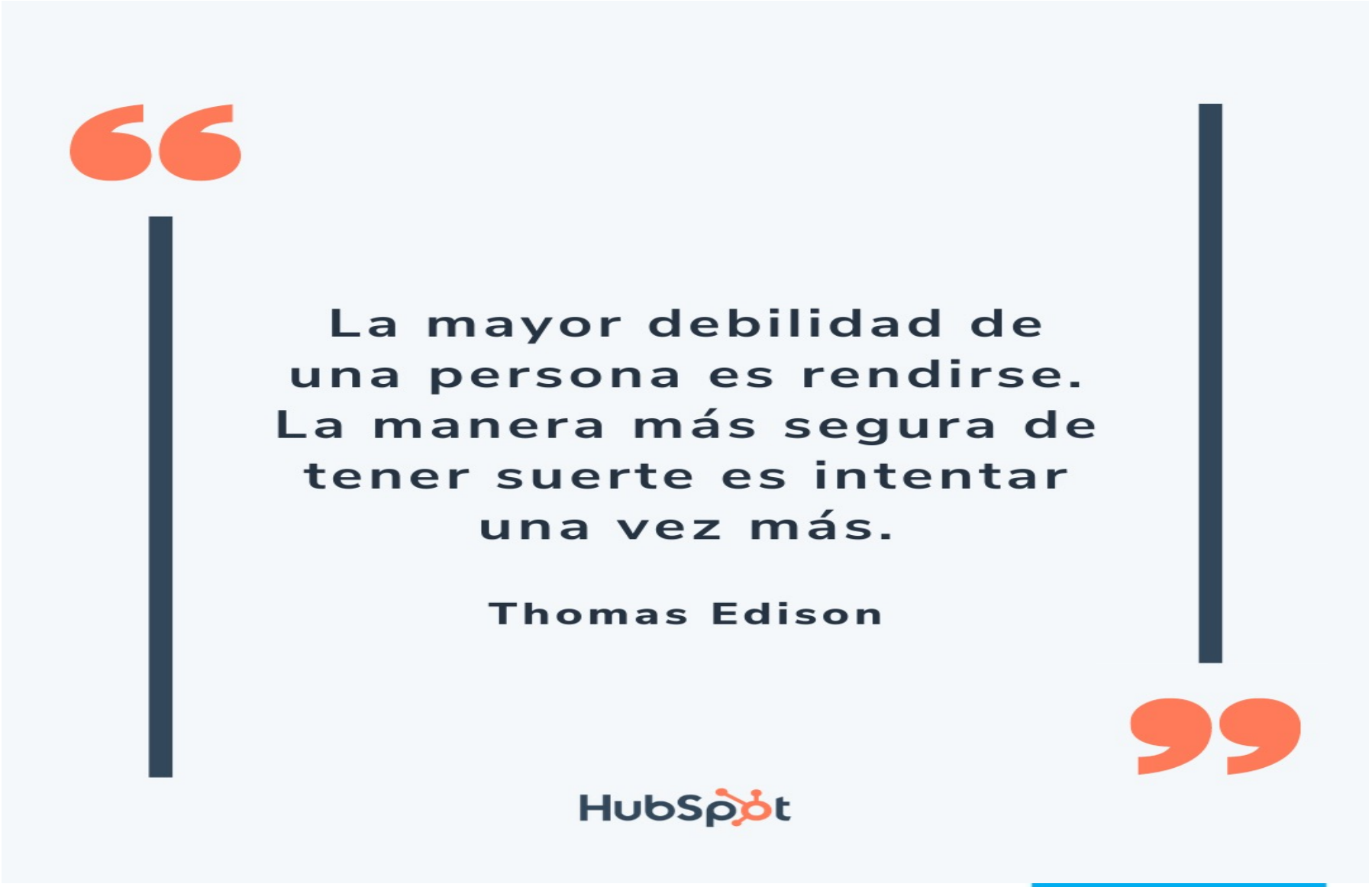
Desarrollo de Aplicaciones
Semana: 13 14

Prof. Dennis Apaza H.

Normas de Seguridad

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Apagar o poner en silencio los celulares.
- Cada estudiante será responsable del equipo asignado.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

Para Pensar . . .


La mayor debilidad de una persona es rendirse. La manera más segura de tener suerte es intentar una vez más.
Thomas Edison
HubSpot

A Recordar . . .

- **Que permiten las etiquetas @Controler y @GetMapping**
- Define cuando una clase creada actuara como controlador, y permite mapear rutas de navegación.
- **Que permite la etiqueta @RequestParam**
- Permite recepcionar las variables enviadas por el método GET

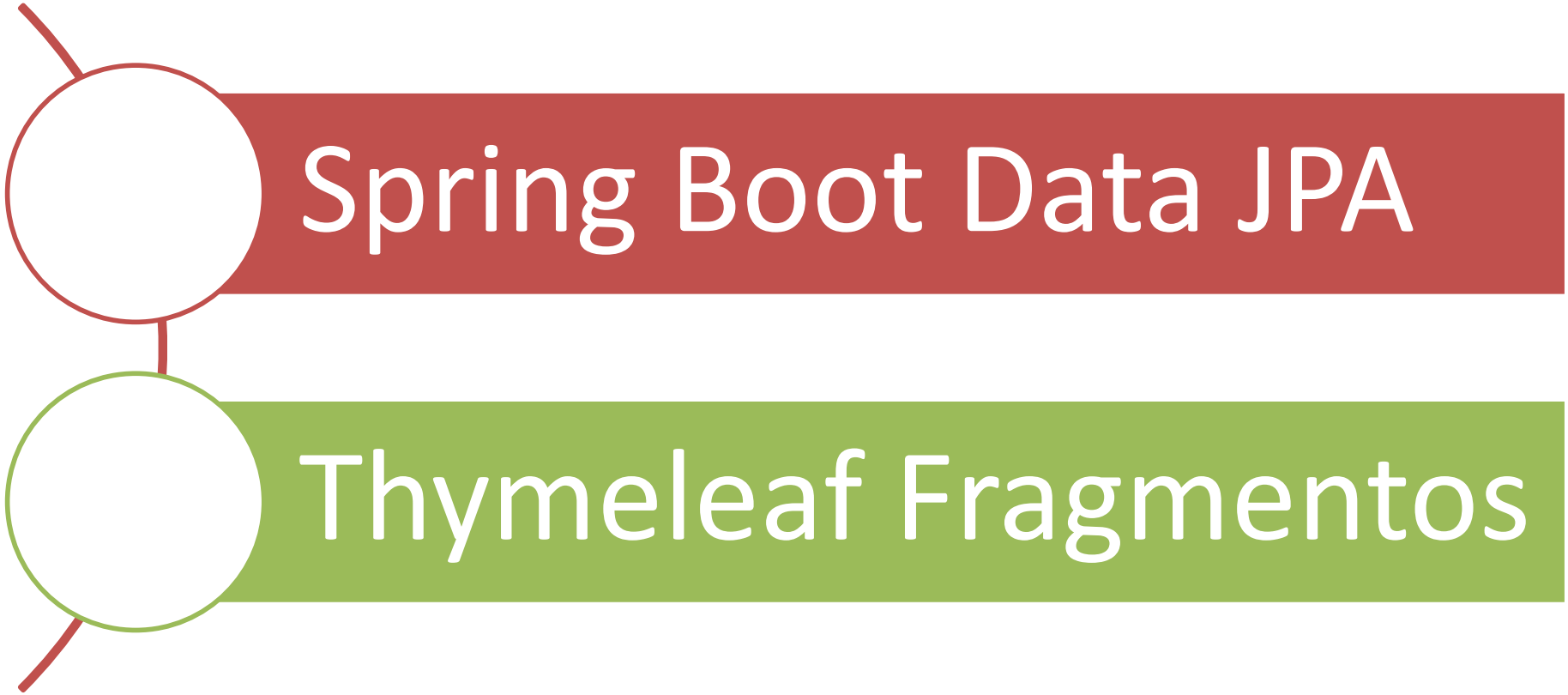
Capacidad


Implementa aplicaciones usando Framework empresariales

Competencia de la Sesión

Identifica las características avanzadas del Framework Spring - Data con diseño de patrones.

Contenido a Tratar



Thymeleaf
Fragmentos

- Los fragmentos de **Thymeleaf** son bloques de código que podemos definir para posteriormente poder reutilizarlos en las distintas páginas de nuestras aplicaciones.
- Los fragmentos se pueden definir en ficheros separados o agrupar varios dentro de un mismo fichero para tenerlos juntos como en una plantilla.

Definición

- Los fragments se definen mediante **th:fragment="nombre_fragmento"**, el fragment incluye tanto la etiqueta sobre la que se pone como su contenido.

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head th:fragment="head">
    <title>Spring JPA</title>
    <meta charset="UTF-8">
   <link th:rel="stylesheet" th:href="@{assets/bootstrap-select-1.13.9/dist/css/bootstrap-select.css}"/>
    <link th:rel="stylesheet" th:href="@{webjars/bootstrap/4.0.0-2/css/bootstrap.min.css} "/>
    <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
</head>
<body>
<div th:fragment="menu">
    <div class="container">
        <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
          <div class="collapse navbar-collapse" id="navbarNavDropdown">
            <ul class="navbar-nav">
              <li class="nav-item active">
                <a class="btn btn-secondary btn-lg" href="/listarEmpleados">
                Tabla Empleados<span class="sr-only">(current)</span></a>
              </li>
              <li class="nav-item active">
                <a class="btn btn-info btn-lg" href="#">Tabla Tareas<span class="sr-only">(current)</span></a>
              </li>
              <li class="nav-item active">
                <a class="btn btn-warning btn-lg" href="#">Asignar Tareas<span class="sr-only">(current)</span></a>
              </li>
            </ul>
            <form class="form-inline my-2 my-lg-0">
          <input class="form-control mr-sm-2" type="search" placeholder="Buscar" aria-label="Buscar">
          <button class="btn btn-success  my-2 my-sm-0" type="submit">Buscar</button>
        </form>
          </div>
        </nav>
    </div>
</div>
</body>
</html>
```

COMO SE USA LOS FRAGMENTOS

- Para utilizar los fragments hay 2 opciones:
  - th:insert
  - th:replace

th:insert

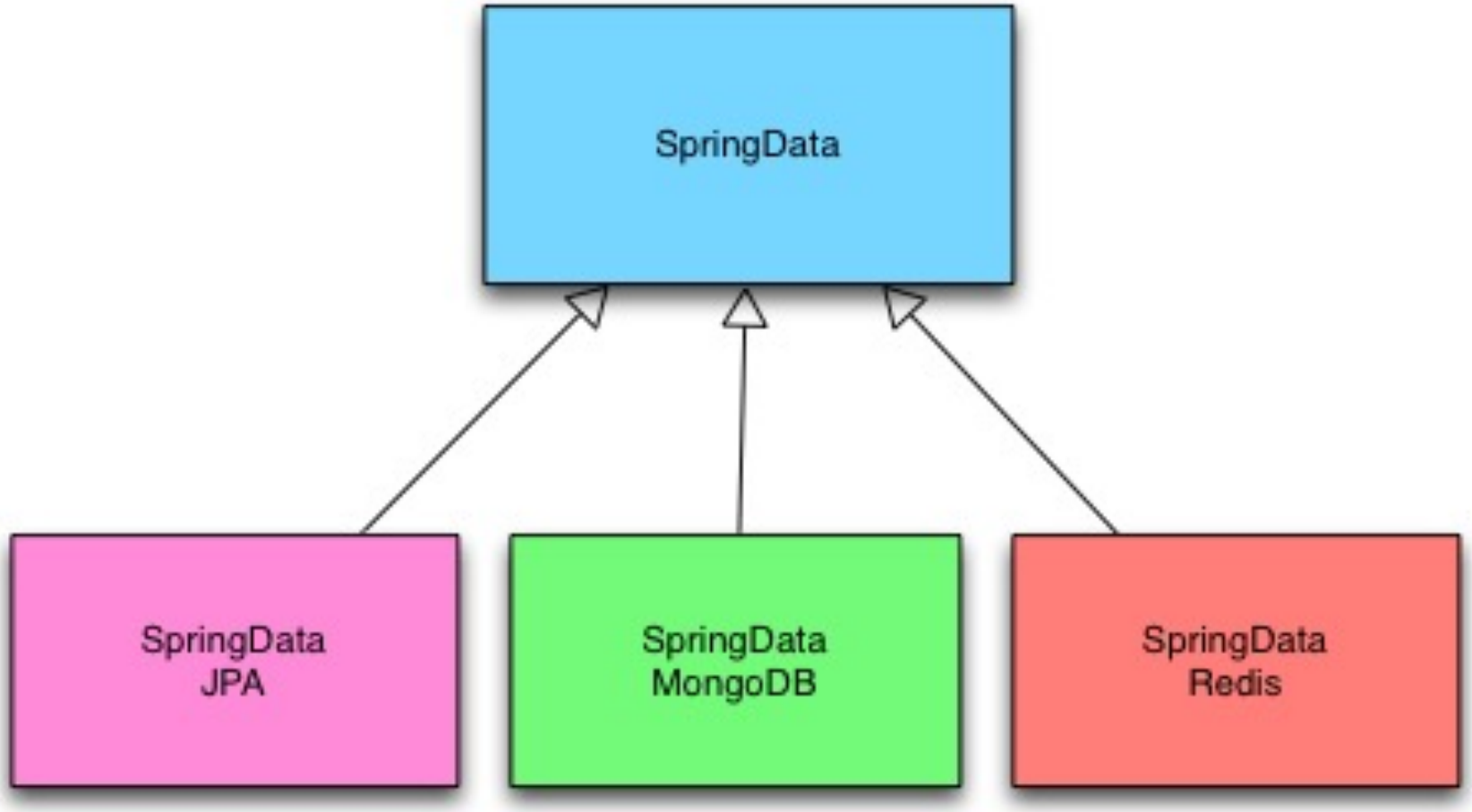
- Con **th:insert="ruta/archivo::nombre_fragmento"** el código del fragmento se incluye dentro del elemento en el que se usa esta etiqueta.
- Por ejemplo
  con **<body th:insert="layouts/base::header">**
  se incluye el fragmet header dentro del body

th:replace

- Con **th:replace="ruta/archivo::nombre_fragmento"** se usa para reemplazar la etiqueta por el fragmento.
- Si usamos **<header th:replace="layouts/base::header"></header>** el resultado va a ser que esa etiqueta header se va a sustituir por el contenido del fragmento header en lugar de incluirlo dentro como en el caso anterior.

Spring Boot Data JPA

- Spring Data es uno de los frameworks que se encuentra dentro de la plataforma de Spring.
- Su objetivo es simplificar al desarrollador la persistencia de datos contra distintos repositorios de información .



Configuración de Spring Data JPA

- Agregar dependencia en Maven

```
<dependencies>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

</dependencies>
```

- **Creacion de Entidad**

```java
package com.miempresa.modelo;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name = "Empleado")
public class Empleado {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "nombre")
    private String nombre;

    @Column(name = "rol")
    private String rol;
```

- **Creación de interface**

```java
package com.miempresa.interfaceServicio;
import java.util.List;
import java.util.Optional;

import com.miempresa.modelo.Empleado;

public interface IEmpleadoServicio {
    public List<Empleado> listar();
    public Optional<Empleado> listarId(int id);
    public int guardar(Empleado p);
    public void borrar(int id);
}
```

- Crear una clase que implemente la interfaz previamente creada

- Crear interface que herede de la clase CrudRepository
- La interfaz **CrudRepository** proporciona métodos para las operaciones CRUD, por lo que le permite crear, leer, actualizar y eliminar registros sin tener que definir sus propios métodos.

```java
package com.miempresa.interfaces;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.miempresa.modelo.Empleado;

@Repository
public interface IEmpleado extends CrudRepository<Empleado, Integer> {

}
```

- La interfaz previa creada, sera utilizada en la interfaz que contiene los metodos CRUD

```
12  @Service
13  public class EmpleadoServicio implements IEmpleadoServicio{
14
15      @Autowired
16      private IEmpleado repo;
17
18      @Override
19      public List<Empleado> listar() {
20          return (List<Empleado>)repo.findAll();
21      }
```

Conexión a MySQL

- Agregar dependencia en Maven

```
<dependencies>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.39</version>
</dependency>

</dependencies>
```

- Crear cadenas de conexión en **application.properties**

```
spring.datasource.url=jdbc:mysql://localhost:3306/db_servicios?useSSL=false&serverTimezone=UTC
spring.jpa.show-sql=true
spring.datasource.username=root
spring.datasource.password=12345678
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
```

Preguntas . . . ?

A Recordar . . .

- **Que son los fragmentos**
- Secciones de código que pueden reutilizarse en determinadas secciones
- **Que es Spring Data JPA**
- Clase para acceso a base de datos, usando conceptos de Entidades y mapeo relacional.

¿Qué aprendimos?

- Spring Boot posee servicios para manejo de acceso a base de datos.
- La configuración de acceso a MySQL se hace a través de archivos de configuración.
- Data JPA evita el uso de sentencias SQL directas a la base de datos

¿Qué veremos la próxima clase?

Bibliografía

- https://programandoointentandolo.com/2019/02/thymeleaf-fragmets.html
- https://www.arquitecturajava.com/introduccion-spring-data-y-jpa/

FIN DE LA UNIDAD