DAW
Desarrollo de Aplicaciones Web
2º Curso

DWES
Desarrollo Web Entorno Servidor

UD 5 Spring Boot

2. Spring MVC

IES BALMIS
Dpto Informática
Curso 2025-2026
Versión 1 (12/2025)

5.2. EJERCICIOS – Static

UD5Ejer5201 – AlquilerCoches

Crear un proyecto **Spring Boot** versión **4.0.0** denominado **AlquilerCoches** de tipo **Jar** y **JDK21** con las dependencias:

- Spring Web
- Spring Boot DevTools

El API que debe ofrecer es el siguiente:

- http://localhost:9201/**alquiler**

NOTA: Deberás cambiar el puerto con la variable:

- server.port=9201

El profesor proporcionará todos los archivos de la web, incluyendo el **/error/404.html**

El proyecto deberá tener la siguiente estructura en recursos:

```
├── /static
│   ├── /error
│   ├── /css
│   ├── /img
│   ├── alquiler.html
│   ├── contacto.html
│   ├── descuentos.html
│   ├── empresa.html
│   ├── index.html
│   └── servicios.html
│
└── application.properties  # Configuración de Spring Boot
```

El proyecto deberá tener la siguiente estructura en src:

```
└── /com.balmis.alquilercar
    └── AlquilerCarApplication.java
```

UD5Ejer5202 – InnovaSolutions

Crear un proyecto **Spring Boot** versión **4.0.0** denominado **InnovaSolutions** de tipo **Jar** y **JDK21** con las dependencias:

- Spring Web
- Spring Boot DevTools

El API que debe ofrecer es el siguiente:

- http://localhost:9202/**innova**

NOTA: Deberás cambiar el puerto con la variable:

- server.port=9202

El profesor proporcionará todos los archivos de la web, incluyendo el **/error/404.html**

El proyecto deberá tener la siguiente estructura en recursos:

```
├── /static
│   ├── /error
│   ├── /css
│   ├── /img
│   ├── /js
│   ├── /webfonts
│   └── index.html
│
└── application.properties  # Configuración de Spring Boot
```

El proyecto deberá tener la siguiente estructura en src:

```
└── /com.balmis.innova
    └── InnovaApplication.java
```

5.2. EJERCICIOS – Dinamic sin Thymeleaf

UD5Ejer5203 – MathCalc

Crear un proyecto **Spring Boot** versión **4.0.0** denominado **MathCalc** de tipo **Jar** y **JDK21** con las dependencias:

- Spring Web
- Spring DevTools

El API que debe ofrecer es el siguiente:

- **Sumar**: recibirá dos parámetros en el PathVariable y devolverá el resultado de restar los dos parámetros recibidos.
  Formato del resultado: texto
  Ejemplo: 20 - 11
  - http://localhost:8080/xmath/aritmetica/sumar/**20/11**

  ```
  31
  ```

- **Restar**: también recibirá dos parámetros en el PathVariable y devolverá el resultado de restar los dos parámetros recibidos.
  Formato del resultado: texto
  Ejemplo: 20 - 11
  - http://localhost:8080/xmath/aritmetica/restar/**20/11**

  ```
  9
  ```

- **Multiplicar**: también recibirá dos parámetros en el PathVariable y devolverá el resultado de multiplicar los dos parámetros recibidos.
  Formato del resultado: texto
  Ejemplo: 7 * 5
  - http://localhost:8080/xmath/aritmetica/multiplicar/5/7

  ```
  35
  ```

- **Dividir**: también recibirá dos parámetros en el PathVariable y otro en QueryString (recoger con RequestParam) que será opcional indicando los decimales del resultado (por defecto 0) y devolverá el resultado de calcular la división de los dos parámetros.
  Formato del resultado: texto
  Ejemplo: calcular 8/3 con 0 decimales por defecto.
  8 / 3 = 2.666666667, redondeado con 0 decimales = 3
  - http://localhost:8080/xmath/aritmetica/dividir/8/3

  ```
  3
  ```

  Ejemplo: calcular 8/3 con 3 decimales por defecto.
  8 / 6 = 2.666666667, redondeado con 0 decimales = 3
  - http://localhost:8080/xmath/aritmetica/dividir/8/3?dec=3

  ```
  2.667
  ```

- **Porcentaje**: también recibirá dos parámetros en el PathVariable y otro en QueryString (recoger con RequestParam) que será opcional indicando los decimales del resultado (por defecto 0) y devolverá el resultado de calcular el porcentaje sobre el primer parámetro que indica en el segundo.
  Formato del resultado: texto
  Ejemplo: calcular el 9% de 54 con 0 decimales por defecto.
  9% de 54 = 4.86, redondeado con 0 decimales = 5
  - http://localhost:8080/xmath/aritmetica/porcentaje/54/9

  ```
  5
  ```

  Ejemplo: calcular el 9% de 54 con 1 decimales
  - http://localhost:8080/xmath/aritmetica/porcentaje/54/9?dec=1

  ```
  4.9
  ```

El profesor proporcionará los archivos:

- **index.html**

El proyecto deberá tener la siguiente estructura en recursos:

```
├── /static
│   └── index.html            # Archivo HTML principal con ejemplos
│
└── application.properties  # Configuración de Spring Boot
```

El proyecto deberá tener la siguiente estructura en src:

```
├── /com.example.mathcalc
│   └── MathcalcApplication.java
│
└── /com.example.mathcalc.controlador
    └── MathcalcControlador.java
```

5.2. EJERCICIOS – Dinamic con Thymeleaf

UD5Ejer5204 – TablaAleatoria

Crear un proyecto **Spring Boot** versión **4.0.0** denominado **TablaAleatoria** de tipo **Jar** y **JDK21** con las dependencias:

- Spring Web
- Spring DevTools
- Thymeleaf

La aplicación generará un HTML que contiene una tabla de 10x10 con letras aleatorias:

	0	1	2	3	4	5	6	7	8	9
0	N	J	P	R	I	Ñ	T	L	F	T
1	I	I	Q	E	R	G	Ñ	A	Y	V
2	B	C	P	X	M	X	B	P	W	H
3	G	W	Z	W	C	F	V	F	M	T
4	C	H	R	N	C	C	G	S	U	O
5	L	D	L	D	Ñ	D	I	V	Ñ	J
6	L	N	V	D	G	S	M	L	G	E
7	Y	Ñ	X	V	Z	V	X	F	B	P
8	T	Q	P	S	Q	R	B	N	P	Y
9	Y	Q	P	F	I	T	X	T	B	G

El profesor proporcionará los archivos:

- **modelo_letras.html**
  - ==> servirá para generar la plantilla de **Thymneleaf** denominada **letras.html**, al que habrá que añadir:

```
<!DOCTYPE html>
<html lang="es" xmlns:th="http://www.thymeleaf.org">
```

- El proyecto en Java **MatrizLetras** que contiene un método para generar aleatoriamente la matriz de letras y que se puede reutilizar

El API que debe ofrecer es el siguiente:

- ```
  @GetMapping("/")
  public String index() {
      return "redirect:/genera"; // Redirige a /genera
  }
  ```
  Formato del resultado: html
  URL: http://localhost:8080/tablaaleatoria

- ```
  @GetMapping("/genera")
  public String genera(Model model)
  ```
  Formato del resultado: html
  Thymeleaf: return **"letras"**
  URL: http://localhost:8080/tablaaleatoria/**genera**

El proyecto deberá tener la siguiente estructura en recursos:

```
├── /static
├── /templates
│   └── letras.html           # Plantilla de Thymeleaf
│
└── application.properties  # Configuración de Spring Boot
```

El proyecto deberá tener la siguiente estructura en src:

```
├── /com.example.tablaleatoria
│   └── TablaAletoriaApplication.java
│
└── /com.example.tablaleatoria.controlador
    └── TablaControlador.java
```

5.2. EJERCICIOS – Dinamic con Thymeleaf y Service

UD5Ejer5205 – ThProvincias

Crear un proyecto **Spring Boot** versión **4.0.0** denominado **ThProvincias** de tipo **Jar** y **JDK21** con las dependencias:

- Spring Web
- Spring DevTools
- Thymeleaf

La aplicación generará un HTML que contiene una tabla con los datos de las provincias almacenadas en una clase **@Service**:

El profesor proporcionará los archivos:

- **tabla_provincias.html**
  - ==> servirá para generar la plantilla de **Thymneleaf** denominada **tabprovincias.html**, al que habrá que añadir:

```
<!DOCTYPE html>
<html lang="es" xmlns:th="http://www.thymeleaf.org">
```

El API que debe ofrecer es los siguientes enlaces:

- **Función**: Redirige a /provincias
  Formato del resultado: el mismo que /provincias
  URL: http://localhost:8080/thservice

- **Función**: mostrar una tabla en HTML con todas las provincias que tenga el ArrayList de @Service
  Formato del resultado: plantilla tabprovincias en html de Thymeleaf
  URL: http://localhost:8080/theservice**/provincias**

- **Función**: mostrar una tabla en HTML con todas las provincias que sean de la comunidad obtenida con @PathVariable
  Formato del resultado: plantilla tabprovincias en html de Thymeleaf
  URL: http://localhost:8080/theservice**/provincias/{comunidad}**

- **Función**: mostrar una tabla en HTML con la provincia que tenga el código obtenido con @RequestParam. Se obtendrá una ArrayList pero solo con una provincia.
  Formato del resultado: plantilla tabprovincias en html de Thymeleaf
  URL: http://localhost:8080/theservice**/provincias/id**?codigo=XX

El proyecto deberá tener la siguiente estructura en recursos:

```
├── /static
├── /templates
│   └── tabprovincias.html    # Plantilla de Thymeleaf
│
└── application.properties  # Configuración de Spring Boot
```

El proyecto deberá tener la siguiente estructura en src:

```
├── /com.example.provincias
│   └── ThProvinciasApplication.java
├── /com.example.provincias.controlador
│   └── ProvinciaControlador.java
├── /com.example.provincias.model
│   └── Provincia.java
└── /com.example.tablaleatoria.service
    └── ProvinciaService.java
```