

Laboratorio

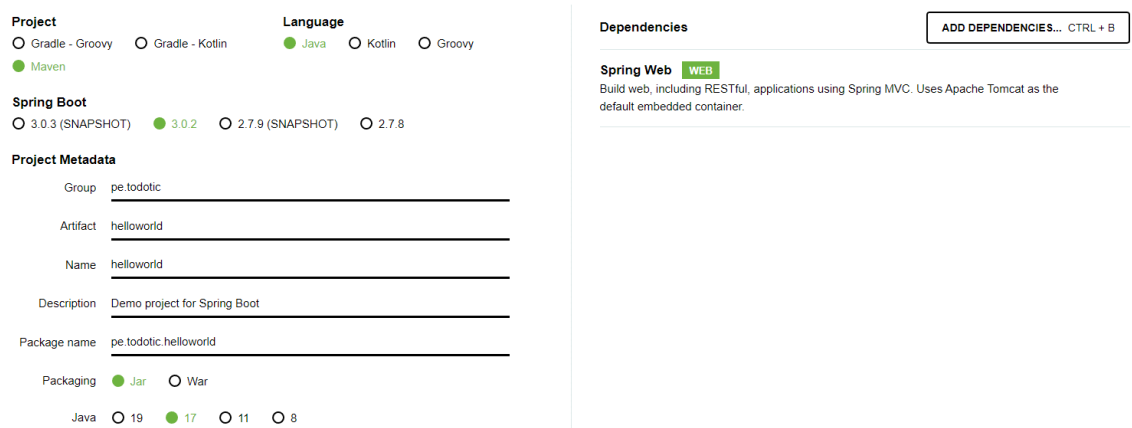
Para apoyarte en esta práctica a continuación se muestran las anotaciones y clases más usadas.

- **@RestController**: Especifica una clase como controlador Rest.
- **@RequestMapping**: Mapea una solicitud http en una clase o método de acuerdo a la ruta especificada.
- **@GetMapping**: Mapea una solicitud http GET en un método de acuerdo a la ruta especificada.
- **@PostMapping**: Mapea una solicitud http POST en un método de acuerdo a la ruta especificada.
- **@PutMapping**: Mapea una solicitud http PUT en un método de acuerdo a la ruta especificada.
- **@DeleteMapping**: Mapea una solicitud http DELETE en un método de acuerdo a la ruta especificada.
- **@ResponseStatus**: Marca una clase o método para modificar el status de la respuesta http que devuelve por defecto.
- **@RequestParam**: indica que un parámetro del método debe capturar el valor de un parámetro de la URL de una solicitud http.
- **@RequestBody**: indica que un parámetro del método debe capturar el cuerpo una solicitud http.
- **@PathVariable**: indica que un parámetro del método debe capturar el valor de una variable de la URL de una plantilla URI.
- **HttpStatus**: Enumeración de los códigos de estado HTTP. Puede usarse conjuntamente con la anotación **@ResponseStatus**.

PRÁCTICA N° 1

CREACIÓN DEL PROYECTO HOLA MUNDO

1. Ingresa a [Spring initializr](#) y realiza la configuración del proyecto de la siguiente forma:



The screenshot shows the Spring Initializr configuration form. The 'Project' section has 'Maven' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '3.0.2' selected. The 'Project Metadata' section has the following values: Group (pe.todotic), Artifact (helloworld), Name (helloworld), Description (Demo project for Spring Boot), and Package name (pe.todotic.helloworld). The 'Packaging' section has 'Jar' selected. The 'Java' section has '17' selected. The 'Dependencies' section has 'Spring Web' selected, with a description: 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.' A button 'ADD DEPENDENCIES... CTRL + B' is visible.

2. Luego hacer clic en el botón "Generate". En seguida se descargará el proyecto de forma comprimida en un archivo zip.

3. Descomprimir el archivo que contiene el proyecto.
4. Abrir el proyecto usando el IDE IntelliJ IDEA.
5. En el paquete principal de la aplicación, crear una clase llamada *HelloController*.
6. Convertir la clase en un Controlador Rest.
7. Crear un método que devuelva un saludo de forma interactiva.
8. Este método debe mapear la ruta *"/hello"* mediante el verbo http GET.
9. El método debe tener un parámetro para capturar el nombre del usuario y usar su valor para incluirlo en el saludo. En caso que el usuario no indique el valor del parámetro establecer un valor por defecto.
10. Desplegar la aplicación, desde la clase Principal del proyecto y realizar las pruebas usando POSTMAN.
11. El resultado de la aplicación debe ser similar al siguiente:

GET http://localhost:8080/hello

Hola mundo!

GET http://localhost:8080/hello?name=Darwin

Hola Darwin!

12. Probar las anotaciones `@RequestMapping`, `@ResponseStatus`, `@RequestBody`, `@PathVariable`

PRACTICA N° 2

PROYECTO MITIENDAAPI: CRUD LIBROS – SIN BASE DE DATOS

1. Descargar e importar el proyecto "bookstoreapi_s1_base".
2. Completar los //TODO disponibles de acuerdo al glosario.

RETO DE LA SESION

Implementar la búsqueda de libros por un término clave. El método de controlador debe retornar una lista de resultados que coincidan con el término de búsqueda.