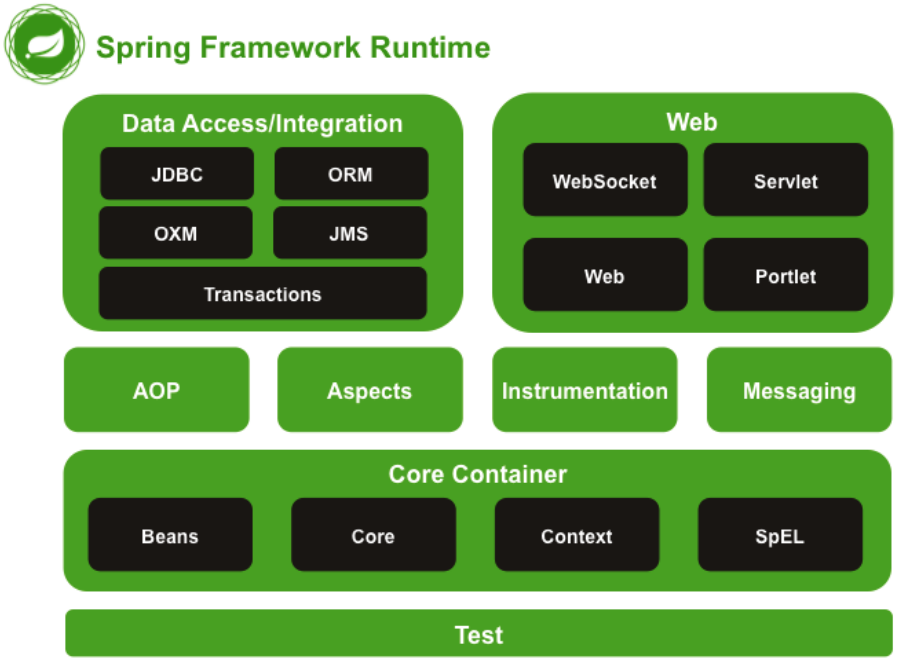
GUSTAVO CORONEL
DESARROLLA SOFTWARE

DESARROLLO
CON SPRING BOOT

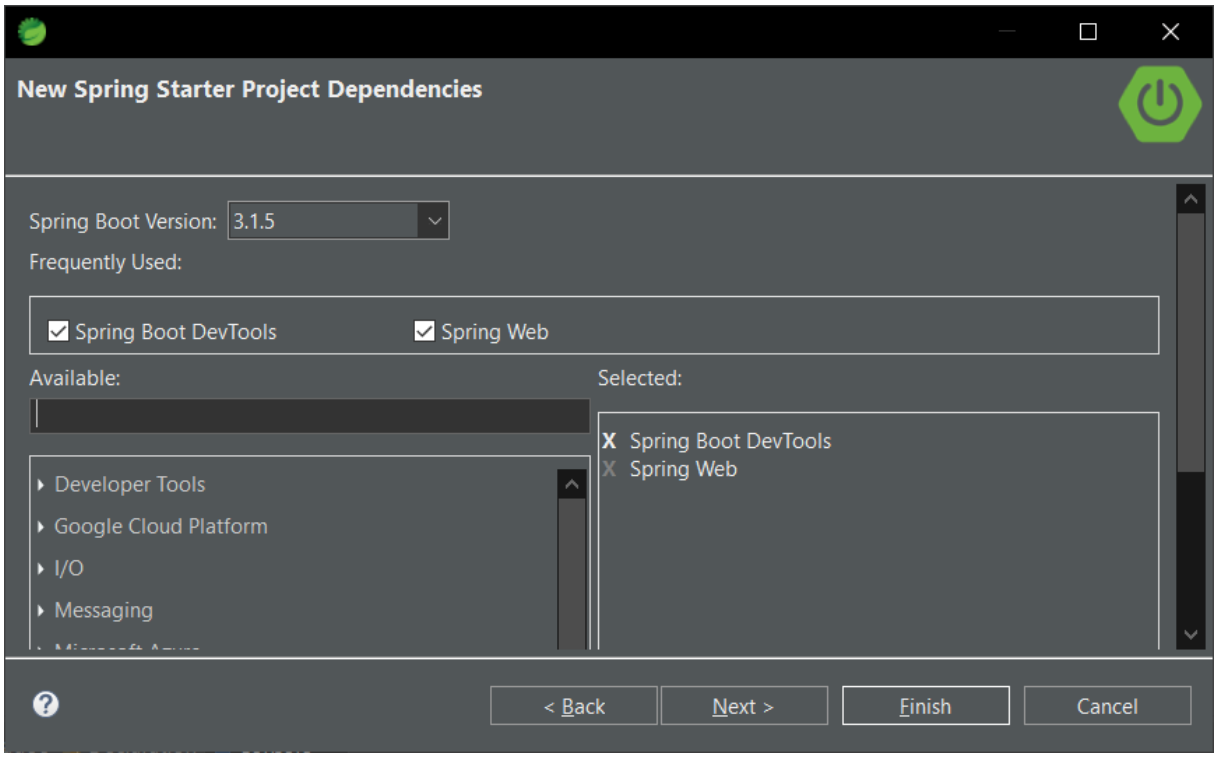


UNIDAD 05
IMPLEMENTADO SERVICIOS REST

Eric Gustavo Coronel Castillo
I N S T R U C T O R
youtube.com/DesarrollaSoftware
gcoronelc@gmail.com

CONTENIDO

CREACIÓN DEL PROYECTO



Para desarrollar un proyecto básico para crear servicios REST debes considerar las siguientes dependencias:

- Spring Web
- Spring Boot DevTools

Si estas considerando trabajar con objetos DTO te recomiendo utilizar Lombok.

CONTROLADOR REST

Los controladores son clases que nos permiten exponer nuestra API para ser usada de forma externa, en Spring Boot para declarar una clase como controlador debes utilizar la anotación @RestController y usaras diferentes anotaciones adicionales para declarar métodos HTTP con tus métodos de tu clase controladora.

Anotación @RestController

La anotación @RestController creará una instancia en el contexto de Spring de la clase que contenga esta anotación, tal como se ilustra a continuación:

```
@RestController
@RequestMapping("/usuarios")
class UsuarioControlador{


}
```

La implementación de `UsuarioControlador` no es pública porque no necesita serlo.

Anotación @RequestMapping

La anotación `@RequestMapping` te permite asignar una solicitud HTTP a un método o clase usando algunos criterios básicos cómo la ruta REST o el método HTTP.

@RequestMapping a nivel de clase

Puedes utilizar la anotación `@RequestMapping` a nivel de clase para asignar la ruta raíz que se usará a nivel de clase.

```
@RestController
@RequestMapping("/usuarios")
class UsuarioControlador {


}
```

@RequestMapping a nivel de método

A nivel de método concatena la ruta declarada a nivel de clase con la ruta declarada en el método, además puedes declarar diferentes métodos HTTP que serán vinculados a la solicitud HTTP.

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioControlador {

   @RequestMapping(value = "/{id}", method = RequestMethod.GET)
   public List<UsuarioDto> encontrarPorId(@PathVariable String id) {


   }
}
```

Métodos específicos

Puedes utilizar diferentes anotaciones para capturar solicitudes con métodos HTTP específicos.

@GetMapping

Anotación para asignar solicitudes HTTP GET a métodos de controlador específicos.

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioControlador {

   @GetMapping("/{id}")
   public UsuarioDto encontrarPorId(@PathVariable String id) {

   }

}
```

@PostMapping

Anotación para asignar solicitudes HTTP POST a métodos de controlador específicos.

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioControlador {

   @PostMapping
   public UsuarioDto crear(@RequestBody UsuarioDto usuario) {

   }

}
```

@PutMapping

Anotación para asignar solicitudes HTTP PUT a métodos de controlador específicos.

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioControlador {

   @PutMapping("/{id}")
   public void actualizar
   ( @PathVariable("id") String id, @RequestBody UsuarioDto usuario ) {

   }

}
```

@DeleteMapping

Anotación para asignar solicitudes HTTP DELETE a métodos de controlador específicos.

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioControlador {

   @DeleteMapping("/{id}")
   public void eliminar(@PathVariable( "id" ) String id) {

   }

}
```

@ResponseStatus

Cuando una solicitud HTTP tiene una respuesta exitosa, Spring proporciona una respuesta HTTP 200 (OK).

Si quieres especificar el estado diferente, puedes marcar ese método con `@ResponseStatus`. Tiene dos argumentos intercambiables para el estado de respuesta deseado: code y value.

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioControlador {

   @PostMapping
   @ResponseStatus(HttpStatus.CREATED)
   public UsuarioDto crear(@RequestBody UsuarioDto usuario) {

   }

}
```

GESTIONANDO PARÁMETROS

@PathVariable

Introducción

La anotación `@PathVariable` te va hacer útil para configurar variables dentro de los propios segmentos de la URL algo que a nivel de arquitecturas REST es imprescindible.

Un mapeo simple

Un caso de uso simple de la anotación @PathVariable sería un punto final que identifica una entidad con una clave principal:

```
@GetMapping("/employees/{id}")
public String getEmployeesById(@PathVariable String id) {
    return "ID: " + id;
}
```

En este ejemplo, se utiliza la anotación `@PathVariable` para extraer la parte con plantilla del URI, representada por la variable `{id}`.

Especificación del nombre de la variable de ruta

En el ejemplo anterior, se omitió definir el nombre de la variable de ruta de la plantilla ya que los nombres del parámetro del método y la variable de ruta son iguales.

Sin embargo, si el nombre de la variable en la ruta es diferente, puedes especificarlo en el argumento de la anotación `@PathVariable`:

```
@GetMapping("/employees/{id}")
public String getEmployeesById(@PathVariable("id") String employeeId) {
   return "ID: " + employeeId;
}
```

Múltiples variables de ruta en una sola solicitud

Dependiendo del caso de uso, puedes tener más de una variable en la ruta de la URI de la solicitud para un método del controlador, que también tiene múltiples parámetros:

```
@GetMapping("/employees/{id}/{name}")
public String getEmployeesByIdAndName
(@PathVariable String id, @PathVariable String name) {
    return "ID: " + id + ", name: " + name;
}
```

Utilizando @PathVariable con Map

```
@GetMapping("/employees/{id}/{name}")
public String getEmployeesByIdAndName
(@PathVariable Map<String, String> pathVarsMap) {
   String id = pathVarsMap.get("id");
   String name = pathVarsMap.get("name");
   if (id != null && name != null) {
      return "ID: " + id + ", name: " + name;
   } else {
      return "Missing Parameters";
   }
}
```

Configurar @PathVariable como no requerido

Puedes establecer la propiedad requerida de `@PathVariable` en falso para que sea opcional.

```
@GetMapping(value = { "/employees", "/employees/{id}" })
public String getEmployeesById(@PathVariable(required = false) String id) {
   if (id != null) {
      return "ID: " + id;
   } else {
      return "ID missing";
   }
}
```

@RequestParam

La anotación `@RequestParam` la utilizas para extraer datos de los parámetros de consulta en la URL de la solicitud. Los parámetros de consulta son los pares clave-valor que aparecen después del signo “?” en una URL.

```
@RestController
@RequestMapping("/users")
public class UserController {

   @GetMapping("/search")
   public ResponseEntity<List<User>> searchUsers
   (@RequestParam("name") String name) {
      // Implementation to search users based on the provided name
      // ...
      return ResponseEntity.ok(users);
   }
}
```

En el script anterior, la anotación `@RequestParam` se utiliza para extraer el parámetro `name` de los parámetros de consulta. El parámetro `name` se especifica como argumento de la anotación `@RequestParam`. Cuando se realiza una solicitud a `/users/search?name=Gustavo`, el valor `Gustavo` se pasará al método `searchUsers` como parámetro de `name`. Luego puede utilizar este parámetro para realizar una operación de búsqueda basada en el nombre proporcionado.

@RequestBody

Tu puede usar la anotación @RequestBody para leer cuerpo de la solicitud pasar sus datos a un objeto.

```
@PostMapping("/accounts")
public void handle(@RequestBody Account account) {
   // ...
}
```

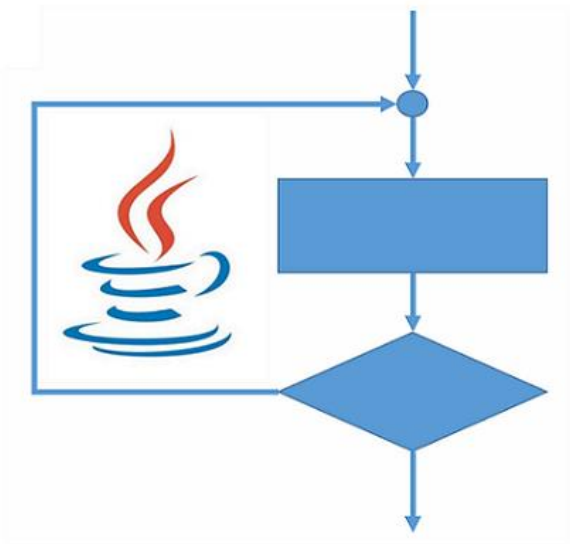
CURSOS VIRTUALES

Acceso a los Cursos Virtuales

En esta URL tienes los accesos a los cursos virtuales:

http://gcoronelc.github.io

Fundamentos de Programación con Java



Tener bases sólidas de programación muchas veces no es fácil, creo que es principalmente por que en algún momento de tu aprendizaje mezclas la entrada de datos con el proceso de los mismos, o mezclas el proceso con la salida o reporte, esto te lleva a utilizar malas prácticas de programación que luego te serán muy difíciles de superar.

En este curso aprenderás las mejores prácticas de programación para que te inicies con éxito en este competitivo mundo del desarrollo de software.

URL del Curso: https://n9.cl/gcoronelc-java-fund

Avance del curso: https://n9.cl/gcoronelc-fp-avance

Cupones de descuento: http://gcoronelc.github.io

Java Orientado a Objetos

CURSO PROFESIONAL DE JAVA ORIENTADO A OBJETOS

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
INSTRUCTOR

En este curso aprenderás a crear software aplicando la Orientación a Objetos, la programación en capas, el uso de patrones de software y Swing.

Cada tema está desarrollado con ejemplos que demuestran los conceptos teóricos y finalizan con un proyecto aplicativo.

URL del Curso: https://bit.ly/2B3ixUW

Avance del curso: https://bit.ly/2RYGXIt

Cupones de descuento: http://gcoronelc.github.io

Programación con Java JDBC

PROGRAMACIÓN DE BASE DE DATOS ORACLE CON JAVA JDBC

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
INSTRUCTOR

En este curso aprenderás a programas bases de datos Oracle con JDBC utilizando los objetos Statement, PreparedStatement, CallableStatement y a programar transacciones correctamente teniendo en cuenta su rendimiento y concurrencia.

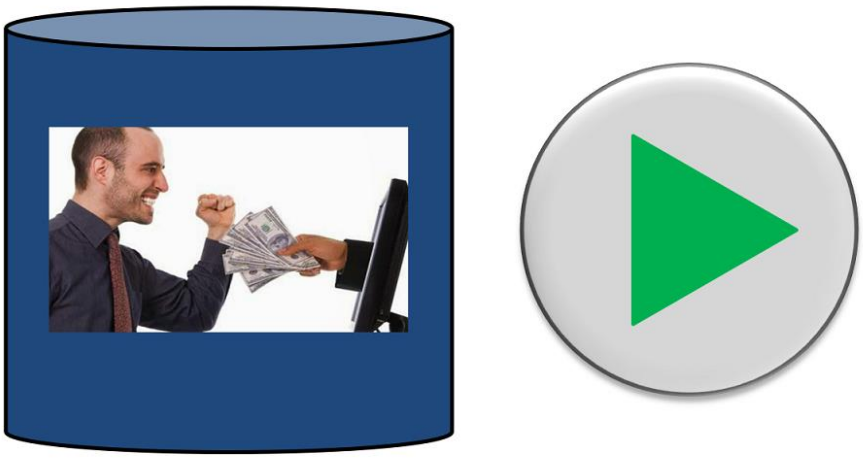
Al final del curso se integra todo lo desarrollado en una aplicación de escritorio.

URL del Curso: https://bit.ly/31apy0O

Avance del curso: https://bit.ly/2vatZOT

Cupones de descuento: http://gcoronelc.github.io

Programación con Oracle PL/SQL



En este curso aprenderás a programas las bases de datos ORACLE con PL/SQL, de esta manera estarás aprovechando las ventas que brinda este motor de base de datos y mejoraras el rendimiento de tus consultas, transacciones y la concurrencia.

Los procedimientos almacenados que desarrolles con PL/SQL se pueden ejecutarlo de Java, C#, PHP y otros lenguajes de programación.

URL del Curso: https://bit.ly/2YZjfxT

Avance del curso: https://bit.ly/3bciqYb

Cupones de descuento: http://gcoronelc.github.io