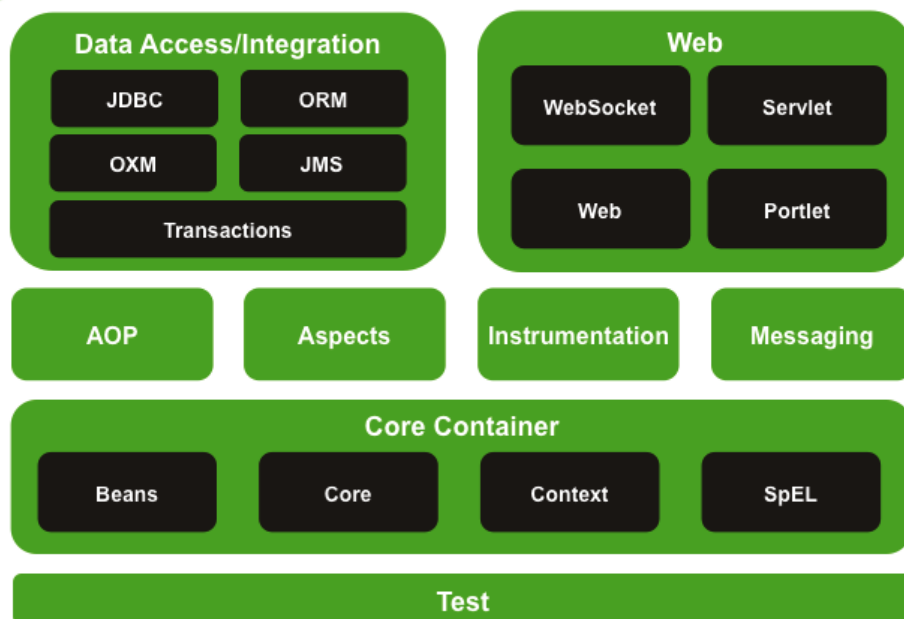




DESARROLLO WEB CON SPRING BOOT



Spring Framework Runtime



UNIDAD 09 DESARROLLO DE UN CRUD

Eric Gustavo Coronel Castillo

www.desarrollasoftware.com

gcoronelc@gmail.com

I N S T R U C T O R



CONTENIDO

INICIAR PROYECTO	4
CREACIÓN DEL PROYECTO	4
DEPENDENCIAS	4
PARÁMETROS DE CONEXIÓN	5
ESTRUCTURA DE PAQUETES	5
RECURSOS ESTÁTICOS	6
PAGINA DE INICIO	7
ARCHIVO HOME.HTML	7
CONTROLADOR	7
PLANTILLA	8
ARCHIVO PLATILLA.HTML	8
ARCHIVO HOME.HTML	10
BASE DE DATOS	11
URL DEL MODELO DE DATO	11
MODELO DE DATOS VENTAS	11
CREACIÓN DEL ESQUEMA	12
TABLA CATEGORIA	12
TABLA PRODUCTO	12
CLASES ENTITY	13
CLASE CATEGORÍA	13
CLASE PRODUCTO	14
IMPLEMENTACIÓN DE LOS SERVICIOS	17
CAPA REPOSITORY	17
<i>Categoría</i>	17
<i>Producto</i>	17
CAPA SERVICE	18
<i>Categoría</i>	18
<i>Producto</i>	19
LISTADO DE PRODUCTOS	21
DISEÑO	21
CONTROLADOR	21
VISTA	22
MENÚ DE OPCIONES	23
CREACIÓN DE NUEVOS PRODUCTOS	24
DISEÑO DEL FORMULARIO	24



CONTROLADOR PARA CARGAR EL FORMULARIO	25
FORMULARIO FRMEDITAR.HTML.....	25
BOTÓN PARA CARGAR EL FORMULARIO.....	27
CONTROLADOR PARA GRABAR PRODUCTO.....	27
EDITAR Y ELIMINAR PRODUCTO	28
CONTROLLER PARA EDITAR UN PRODUCTO	28
CONTROLLER PARA ELIMINAR UN PRODUCTO	29
BOTONES DE EDICIÓN Y ELIMINACIÓN	30
SOLICITAR LA CONFIRMACIÓN.....	31
VALIDACIÓN DEL FORMULARIO	32
LIBRERÍA	32
LA ENTIDAD PRODUCTO.....	32
GRABAR PRODUCTO	33
CAMPOS DEL FORMULARIO	33
PERSONALIZANDO LOS MENSAJES DE ERROR	35
MENSAJES DE ALERTA	37
ESTRUCTURA	37
<i>En el controller.....</i>	<i>37</i>
<i>En la Vista.....</i>	<i>37</i>
EN LA PLANTILLA.....	38
ALERTA AL GRABAR EL PRODUCTO	39
AL EDITAR UN PRODUCTO	40
AL ELIMINAR UN PRODUCTO.....	41
OTROS MENSAJES DE ERROR	42
ESTRUCTURA	42
ERROR HTTP-400	43
ERROR HTTP 404.....	45
EJERCICIO PROPUESTO	47
CURSOS VIRTUALES.....	48
CUPONES	48
FUNDAMENTOS DE PROGRAMACIÓN CON JAVA	48
JAVA ORIENTADO A OBJETOS	49
PROGRAMACIÓN CON JAVA JDBC.....	50
PROGRAMACIÓN CON ORACLE PL/SQL	51



INICIAR PROYECTO

Creación del proyecto

New Spring Starter Project Dependencies

Spring Boot Version: 2.4.0

Frequently Used:

- ☐ JDBC API
- ☒ Spring Web
- ☒ Oracle Driver
- ☒ Thymeleaf
- ☒ Spring Data JPA

Available:

- Spring Boot
- Developer Tools
 - ☒ Spring Boot DevTools
- NoSQL
 - ☐ Spring for Apache Geode
- Ops
 - ☐ Spring Boot Actuator
 - ☐ Codecentric's Spring Boot Admin (Client)
 - ☐ Codecentric's Spring Boot Admin (Server)

Selected:

- X Spring Boot DevTools
- X Spring Data JPA
- X Oracle Driver
- X Thymeleaf
- X Spring Web

< Back Next > Finish Cancel

Dependencias

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
</dependencies>
```



```
</dependency>
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc8</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>
```

Parámetros de Conexión

Estos parámetros se deben incluir en el archivo de propiedades (application.properties), también puedes establecer el puerto del servidor.

```
server.port=9090

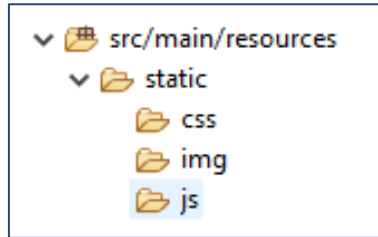
# Oracle settings
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/XE
spring.datasource.username=ventas
spring.datasource.password=admin
spring.jpa.database-platform=org.hibernate.dialect.Oracle12cDialect
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Estructura de paquetes

```
▼ src/main/java
  > com.desarrollasoftware.app
    com.desarrollasoftware.app.controller
    com.desarrollasoftware.app.entity
    com.desarrollasoftware.app.repository
    com.desarrollasoftware.app.service
    com.desarrollasoftware.app.util
```



Recursos estáticos

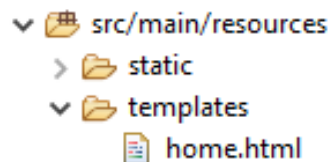




PAGINA DE INICIO

Archivo home.html

Ubicación



Archivo

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Página de Inicio</title>
</head>
<body>
  <h1>Hola desde Spring Boot</h1>
</body>
</html>
```

Controlador

```
package com.desarrollasoftware.app.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class AppController {

    @GetMapping({"/", "/home"})
    public String home() {
        return "home";
    }

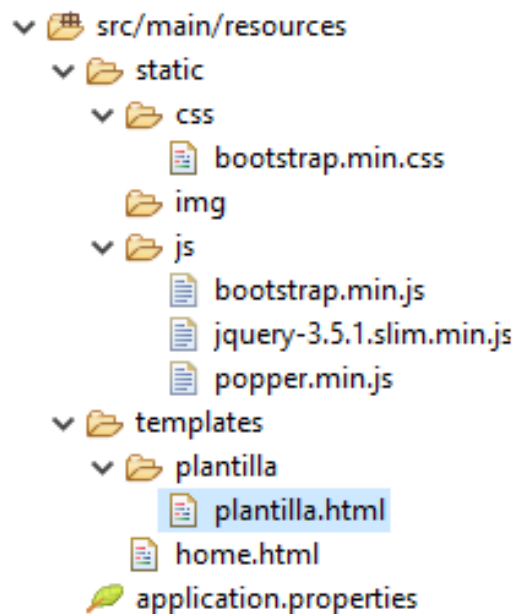
}
```



PLANTILLA

Archivo platilla.html

Ubicación



Archivo

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:fragment="head">
<meta charset="UTF-8">
<!-- Bootstrap CSS -->
<link rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
<title>APP CRUD</title>
</head>
<body>
<!-- Barra de navegación -->
<header th:fragment="header">
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<div class="container">
<a class="navbar-brand" href="#"> <strong>MyLogo</strong>
</a>

<button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent">
```




```
        aria-controls="navbarSupportedContent" aria-expanded="false"
        aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <!-- ENLACES DEL MENU -->
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item"><a class="nav-link"
                th:href="@{/home}">Inicio</a></li>
            <li class="nav-item"><a class="nav-link"
                href="#">Productos</a></li>
        </ul>
    </div>

</div>
</nav>
</header>

<!-- Contenido -->
<div class="container"></div>

<!-- Pie de pagina -->
<footer th:fragment="footer"
    class="bg-dark text-center text-white fixed-bottom">

    <div class="container">
        <p>Copyright Desarrolla Software &copy;2020</p>
    </div>

    <script type="text/javascript"
        th:src="@{/js/jquery-3.5.1.slim.min.js}"></script>
    <script type="text/javascript" th:src="@{/js/popper.min.js}"></script>
    <script type="text/javascript" th:src="@{/js/bootstrap.min.js}"></script>

</footer>

</body>
</html>
```



Archivo home.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>

    <header th:replace="plantilla/plantilla :: header">
</header>

    <div class="container">
        <h1>Página principal</h1>
    </div>

    <footer th:replace="plantilla/plantilla :: footer">
</footer>
</body>
</html>
```

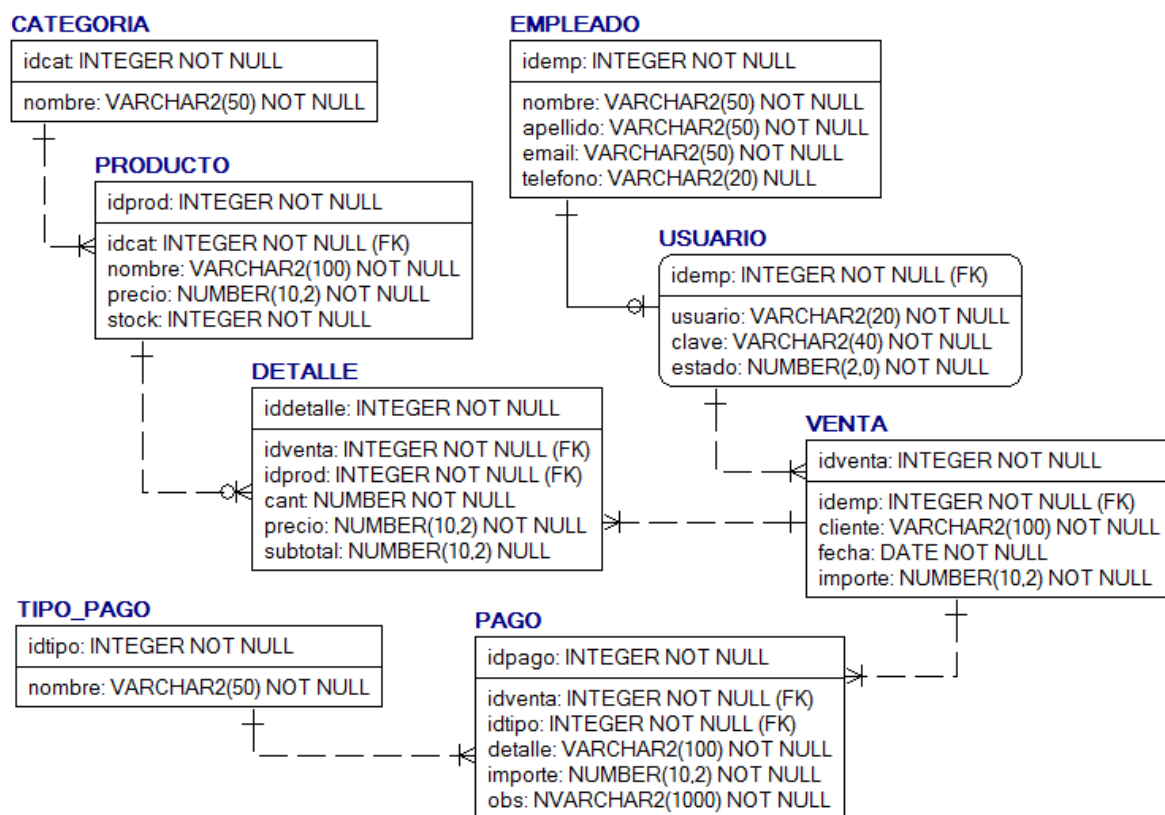


BASE DE DATOS

URL del modelo de dato

<https://github.com/gcoronelc/databases>

Modelo de datos VENTAS





Creación del esquema

Proceda a crear el esquema en Oracle:

```
SQL> @D:\databases\Ventas\Oracle\Modelo\Modelo.SQL
```

Tabla CATEGORIA

```
SQL> desc categoria;
```

Name	Null?	Type
-----	-----	-----
IDCAT	NOT NULL	NUMBER(38)
NOMBRE	NOT NULL	VARCHAR2(100)

Tabla PRODUCTO

```
SQL> describe producto;
```

Name	Null?	Type
-----	-----	-----
IDPROD	NOT NULL	NUMBER(38)
IDCAT	NOT NULL	NUMBER(38)
NOMBRE	NOT NULL	VARCHAR2(100)
PRECIO	NOT NULL	NUMBER(10,2)
STOCK	NOT NULL	NUMBER(38)



CLASES ENTITY

Clase Categoría

```
package com.desarrollasoftware.app.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "categoria")
public class Categoria implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Column(name = "idcat")
    private Long id;

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

    public Categoria() {
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```



```
}

@Override
public String toString() {
    return "Categoria [id=" + id + ", nombre=" + nombre + "]";
}

}
```

Clase Producto

```
package com.desarrollasoftware.app.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

@Entity
@Table(name = "producto")
@SequenceGenerator(name = "sq_producto", sequenceName = "sq_producto",
allocationSize = 1)
public class Producto implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Column(name = "idprod")
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "sq_producto")
    private Long id;

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

    @ManyToOne
    @JoinColumn(name = "idcat")
```



```
private Categoria categoria;

@Column(name = "precio")
private Double precio;

@Column(name = "stock")
private Long stock;

public Producto() {
    this.id = 0L;
    this.precio = 0.0;
    this.stock = 0L;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public Categoria getCategoria() {
    return categoria;
}

public void setCategoria(Categoria categoria) {
    this.categoria = categoria;
}

public Double getPrecio() {
    return precio;
}

public void setPrecio(Double precio) {
    this.precio = precio;
}
```



```
public Long getStock() {  
    return stock;  
}  
  
public void setStock(Long stock) {  
    this.stock = stock;  
}  
  
@Override  
public String toString() {  
    return "Producto [id=" + id + ", nombre=" + nombre  
        + ", categoria=" + categoria + ", precio=" + precio  
        + ", stock=" + stock + "];"  
}  
}
```




IMPLEMENTACIÓN DE LOS SERVICIOS

Capa Repository

Categoría

```
package com.desarrollasoftware.app.repository;

import org.springframework.data.repository.CrudRepository;

import com.desarrollasoftware.app.entity.Categoria;

public interface CategoriaRepository extends CrudRepository<Categoria, Long>{

}
```

Producto

```
package com.desarrollasoftware.app.repository;

import org.springframework.data.repository.CrudRepository;

import com.desarrollasoftware.app.entity.Producto;

public interface ProductoRepository extends CrudRepository<Producto, Long>{

}
```



Capa Service

Categoría

La interface

```
package com.desarrollasoftware.app.service;

import java.util.List;
import com.desarrollasoftware.app.entity.Categoria;

public interface CategoriaService {

    public List<Categoria> listarTodos();
}
```

La implementación

```
package com.desarrollasoftware.app.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.desarrollasoftware.app.entity.Categoria;
import com.desarrollasoftware.app.repository.CategoriaRepository;
import com.desarrollasoftware.app.service.CategoriaService;

@Service
public class CategoriaServiceImpl implements CategoriaService{

    @Autowired
    private CategoriaRepository repository;

    @Override
    public List<Categoria> listarTodos() {
        List<Categoria> lista = (List<Categoria>) repository.findAll();
        return lista;
    }
}
```



Producto

La interface

```
package com.desarrollasoftware.app.service;

import java.util.List;

import com.desarrollasoftware.app.entity.Producto;

public interface ProductoService {

    public List<Producto> listarTodos();
    public void grabar(Producto producto);
    public Producto buscarPorId(Long id);
    public void eliminar(Long id);

}
```

La implementación

```
package com.desarrollasoftware.app.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.desarrollasoftware.app.entity.Producto;
import com.desarrollasoftware.app.repository.ProductoRepository;
import com.desarrollasoftware.app.service.ProductoService;

@Service
public class ProductoServiceImpl implements ProductoService {

    @Autowired
    private ProductoRepository repository;

    @Override
    public List<Producto> listarTodos() {
        List<Producto> lista = (List<Producto>) repository.findAll();
        return lista;
    }

}
```



```
@Override
public void grabar(Producto producto) {
    repository.save(producto);
}

@Override
public Producto buscarPorId(Long id) {
    Producto bean = repository.findById(id).orElse(null);
    return bean;
}

@Override
public void eliminar(Long id) {
    repository.deleteById(id);
}
}
```



LISTADO DE PRODUCTOS

Diseño

Desarrolla Software Inicio Clientes				
LISTA DE PRODUCTOS				
ID	CATEGORIA	NOMBRE	PRECIO	STOCK
1001	LINEA BLANCA	COCINA	900.0	456
1002	ROPA DE SEÑORITAS	PANTALON	150.0	4567
1003	LINEA BLANCA	REFRIGERADORA	1300.0	690
1004	ROPA DE SEÑORITAS	POLO DE VERANO	95.0	150
1005	ROPA DE CABALLEROS	CAMISA COLOR VERDE	140.0	250
1006	ROPA DE CABALLEROS	CAMISA DE CUADROS PEQUEÑOS	140.0	350
1007	ROPA DE CABALLEROS	PANTALON MODELO A1	1180.0	450
1021	LINEA BLANCA	COCINA	900.0	456
Copyright Desarrolla Software ©2020				

Controlador

```
package com.desarrollasoftware.app.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import com.desarrollasoftware.app.entity.Producto;
import com.desarrollasoftware.app.service.ProductoService;

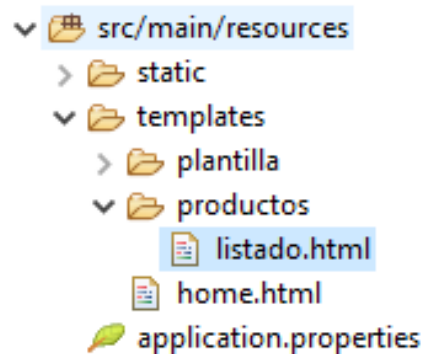
@Controller
@RequestMapping("/productos")
public class ProductoController {

    @Autowired
    private ProductoService productoService;
```



```
@GetMapping("/{","todos"})
public String listar(Model model) {
    List<Producto> lista = productoService.listarTodos();
    model.addAttribute("titulo", "LISTA DE PRODUCTOS");
    model.addAttribute("productos", lista);
    return "/productos/listado";
}
}
```

Vista



```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>

    <header th:replace="plantilla/plantilla :: header">
    </header>

    <div class="container">
        <h1 th:text="${titulo}"></h1>

        <table class="table">
            <thead class="thead-light">
                <tr>
                    <th scope="col">ID</th>
                    <th scope="col">CATEGORIA</th>
                    <th scope="col">NOMBRE</th>
                    <th scope="col">PRECIO</th>
```



```
<th scope="col">STOCK</th>
</tr>
</thead>
<tbody>
  <tr th:each="pr:${productos}">
    <th scope="row" th:text="${pr.id}"></th>
    <td th:text="${pr.categoria.nombre}"></td>
    <td th:text="${pr.nombre}"></td>
    <td th:text="${pr.precio}"></td>
    <td th:text="${pr.stock}"></td>
  </tr>
</tbody>
</table>

</div>

<footer th:replace="plantilla/plantilla :: footer">
</footer>
</body>
</html>
```

Menú de opciones

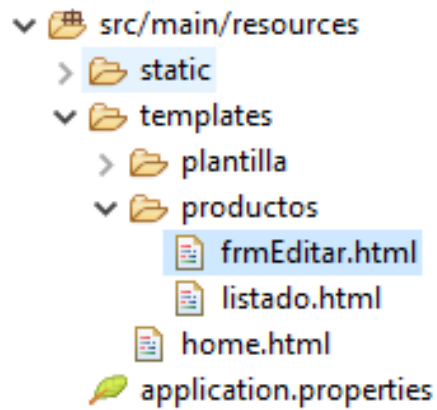
Actualizar el menú en el archivo **plantilla.html**:

```
<!-- ENLACES DEL MENU -->
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item"><a class="nav-link"
      th:href="@{/home}">Inicio</a></li>
    <li class="nav-item"><a class="nav-link"
      th:href="@{/productos/todos}">Productos</a></li>
  </ul>
</div>
```



CREACIÓN DE NUEVOS PRODUCTOS

Diseño del formulario



NUEVO PRODUCTO

Categoría:

LINEA BLANCA

Nombre:

Nombres del producto

Precio:

0.0

Stock:

0

Guardar



Controlador para cargar el formulario

```
@GetMapping("/crear")
public String crear(Model model) {

    Producto producto = new Producto();
    List<Categoria> listaCategorias = categoriaService.listarTodos();

    model.addAttribute("titulo", "NUEVO PRODUCTO");
    model.addAttribute("producto", producto);
    model.addAttribute("categorias", listaCategorias);

    return "/productos/frmEditor";
}
```

Formulario frmEditor.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>

    <header th:replace="plantilla/plantilla :: header"> </header>

    <div class="container mt-2">
        <form th:object="${producto}" method="post">
            <div class="card bg-light">

                <div class="card-header bg-info text-white" th:text="${titulo}"></div>

                <div class="card-body">

                    <div class="form-group row">
                        <label class="col-md-2" for="categoria">Categoría:</label> <select
                            th:field="*{categoria}"
                            class="form-control form-control-sm col-md-10" id="categoria">
                            <option th:each="categoria:${categorias}"
                                th:value="${categoria.id}" th:text="${categoria.nombre}" />
                        </select>
                    </div>

                </div>

            </div>
        </form>
    </div>
```



```
<div class="form-group row">
  <label class="col-md-2" for="nombre">Nombre:</label> <input
    type="text" th:field="*{nombre}"
    class="form-control form-control-sm col-md-10" id="nombre"
    placeholder="Nombres del producto"> <small> </small>
</div>

<div class="form-group row">
  <label class="col-md-2" for="precio">Precio:</label> <input
    type="text" th:field="*{precio}"
    class="form-control form-control-sm col-md-10" id="precio"
    placeholder="Precio del producto"> <small></small>
</div>

<div class="form-group row">
  <label class="col-md-2" for="stock">Stock:</label> <input
    type="text" th:field="*{stock}"
    class="form-control form-control-sm col-md-10" id="stock"
    placeholder="Stock del producto"> <small></small>
</div>

</div>

<div class="card-footer bg-light">
  <input type="submit" class="btn btn-primary btn-sm" value="Guardar">
</div>

</div>
</form>
</div>

<footer th:replace="plantilla/plantilla :: footer"> </footer>
</body>
</html>
```



Botón para cargar el formulario

LISTA DE PRODUCTOS

Nuevo Producto

ID	CATEGORIA	NOMBRE	PRECIO	STOCK
1001	LINEA BLANCA	COCINA	900.0	456
1002	ROPA DE SEÑORITAS	PANTALON	150.0	4567

En el archivo listado.html debes agregar el enlace para crear el botón **Nuevo Producto**:

```
<a class="btn btn-primary btn-sm" th:href="@{/productos/crear}"  
th:text="'Nuevo Producto'"></a>
```

Controlador para grabar producto

```
@PostMapping("/grabar")  
public String guardar(@ModelAttribute Producto producto) {  
  
    productoService.grabar(producto);  
    System.out.println("Producto grabado con exito!");  
    return "redirect:/productos/";  
}
```



EDITAR Y ELIMINAR PRODUCTO

Controller para editar un producto

```
@GetMapping("/editar/{id}")
public String editar(@PathVariable("id") Long idProd, Model model) {

    Producto producto = productoService.buscarPorId(idProd);
    List<Categoria> listaCategorias = categoriaService.listarTodos();

    model.addAttribute("titulo", "EDITAR PRODUCTO (" + idProd + ")");
    model.addAttribute("producto", producto);
    model.addAttribute("categorias", listaCategorias);

    return "/productos/frmEditar";
}
```

Puedes editar un producto ingresando esta URL:

```
http://localhost:9090/productos/editar/1005
```

Después de modificar los datos de un producto y hacer click en el botón grabar, ¿qué sucede?

Agregar la siguiente línea de código al formulario de edición:

```
<input type="hidden" th:field="*{id}" id="id"/>
```



Controller para eliminar un producto

```
@GetMapping("/eliminar/{id}")
public String eliminar(@PathVariable("id") Long idProd) {

    productoService.eliminar(idProd);
    System.out.println("Producto eliminado con exito!");
    return "redirect:/productos/";

}
```

Puede eliminar un producto utilizando la siguiente URL:

```
http://localhost:9090/productos/eliminar/1033
```



Botones de edición y eliminación

En la tabla de productos debes incluir dos columnas, una para editar un registro y otra para eliminar un registro.

```
<table class="table">
  <thead class="thead-light">
    <tr>
      <th scope="col">ID</th>
      <th scope="col">CATEGORIA</th>
      <th scope="col">NOMBRE</th>
      <th scope="col">PRECIO</th>
      <th scope="col">STOCK</th>
      <th scope="col">EDITAR</th>
      <th scope="col">ELIMINAR</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="pr:${productos}">
      <th scope="row" th:text="${pr.id}"></th>
      <td th:text="${pr.categoria.nombre}"></td>
      <td th:text="${pr.nombre}"></td>
      <td th:text="${pr.precio}"></td>
      <td th:text="${pr.stock}"></td>
      <td><a class="btn btn-success btn-sm"
        th:href="@{/productos/editar/} + ${pr.id}" th:text="'Editar'"
        title="Editar producto."></a></td>
      <td><a class="btn btn-danger btn-sm"
        th:href="@{/productos/eliminar/} + ${pr.id}" th:text="'Eliminar'"
        title="Eliminar producto."></a></td>
    </tr>
  </tbody>
</table>
```

Ya puede probar la aplicación, tanto la opción de editar y la opción de eliminar deben funcionar correctamente.

¿Qué notas en las nuevas funcionalidades?

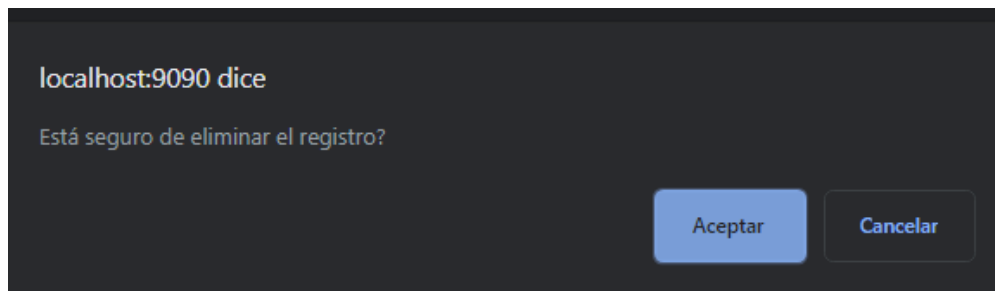


Solicitar la confirmación

En este caso debes modificar el botón eliminar como se ilustra a continuación:

```
<td>
  <a class="btn btn-danger btn-sn"
    th:href="@{/productos/eliminar/} + ${pr.id}" th:text="'Eliminar'"
    title="Eliminar producto."
    onclick="return confirm('Está seguro de eliminar el registro?');"></a>
</td>
```

Ahora, cuando intentes eliminar un registro debes confirmar la acción con una ventana similar a la que se muestra a continuación:





VALIDACIÓN DEL FORMULARIO

Librería

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

La entidad producto

Atributo nombre

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

Atributo precio

```
@Column(name = "precio")
@NotNull
@Min(value = 0)
private double precio;
```

Atributo stock

```
@Column(name = "stock")
@NotNull
@Min(value = 0)
private long stock;
```




Grabar producto

```
@PostMapping("/grabar")
public String guardar(@Valid @ModelAttribute Producto producto,
BindingResult result, Model model) {

    if (result.hasErrors()) {
        System.err.println("Se presentaron errores en el formulario!");
        String titulo = "NUEVO PRODUCTO";
        if (producto.getId() > 0) {
            titulo = "EDITAR PRODUCTO (" + producto.getId() + ")";
        }
        List<Categoria> listaCategorias = categoriaService.listarTodos();
        model.addAttribute("titulo", titulo);
        model.addAttribute("producto", producto);
        model.addAttribute("categorias", listaCategorias);
        return "/productos/frmEditar";
    }

    productoService.grabar(producto);
    System.out.println("Producto grabado con exito!");

    return "redirect:/productos/";
}
```

Campos del formulario

Campo nombre

```
<div class="form-group row">
    <label class="col-md-2" for="nombre">Nombre:</label>
    <div class="col-md-10">
        <input type="text" th:field="*{nombre}"
            class="form-control form-control-sm" id="nombre"
            placeholder="Nombres del producto" /> <small
            class="form-text text-danger"
            th:if="${#fields.hasErrors('nombre')}}" th:errors="*{nombre}"></small>
        </div>
    </div>
```



Campo precio

```
<div class="form-group row">
  <label class="col-md-2" for="precio">Precio:</label>
  <div class="col-md-10">
    <input type="text" th:field="*{precio}"
      class="form-control form-control-sm" id="precio"
      placeholder="Precio del producto" /> <small
      class="form-text text-danger"
      th:if="${#fields.hasErrors('precio')}}" th:errors="*{precio}"></small>
  </div>
</div>
```

Campo stock

```
<div class="form-group row">
  <label class="col-md-2" for="stock">Stock:</label>
  <div class="col-md-10">
    <input type="text" th:field="*{stock}"
      class="form-control form-control-sm" id="stock"
      placeholder="Stock del producto" /> <small
      class="form-text text-danger"
      th:if="${#fields.hasErrors('stock')}}" th:errors="*{stock}"></small>
  </div>
</div>
```

En este momento ya puedes probar el formulario:



NUEVO PRODUCTO

Categoría:

LINEA BLANCA

Nombre:

Nombres del producto

no debe estar vacío

Precio:

0.0

Stock:

0

Guardar

Personalizando los mensajes de error

En la carpeta **resources** debes crear el archivo **messages.properties** con los mensajes respectivos:

```
#Validación del producto
NotEmpty.producto.nombre=El nombre del producto es requerido.
NotNull.producto.precio=El precio del producto es requerido.
NotNull.producto.stock=El stock del producto es requerido.
DecimalMin.producto.precio=El precio no puede tomar valores negativos
Min.producto.stock=El stock no puede tomar valores negativos
typeMismatch.producto.precio=El precio del producto es un numero decimal
typeMismatch.producto.stock=El stock del producto es un numero entero
```

Ahora puedes probar nuevamente el formulario:



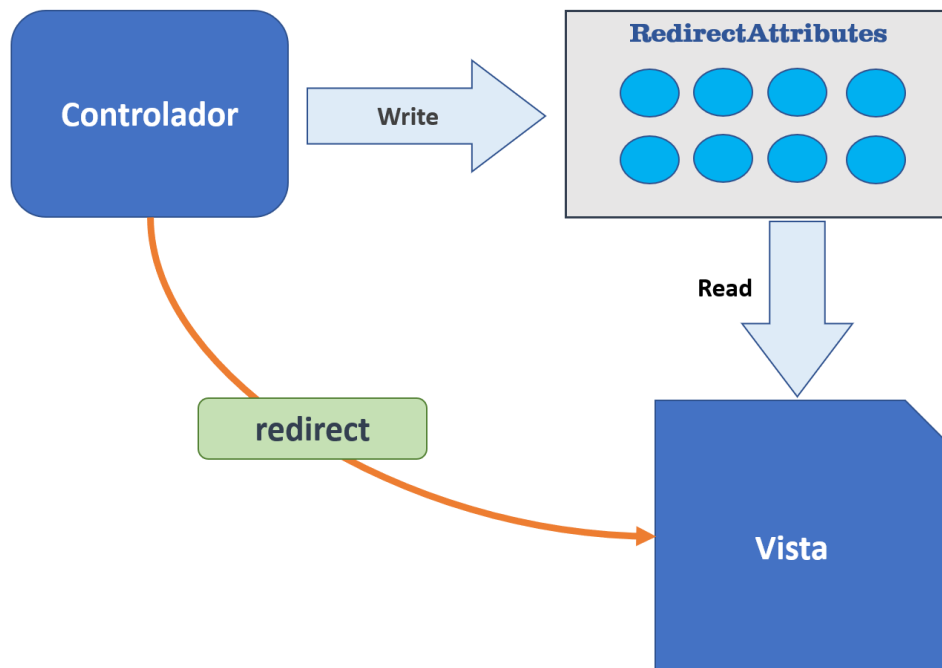
NUEVO PRODUCTO

Categoría:	<input type="text" value="LINEA BLANCA"/>
Nombre:	<input type="text" value="Nombres del producto"/> <small>El nombre del producto es requerido.</small>
Precio:	<input type="text" value="abc"/> <small>El precio del producto es un numero decimal</small>
Stock:	<input type="text" value="34.6"/> <small>El stock del producto es un numero entero</small>

Guardar



MENSAJES DE ALERTA



Estructura

En el controller

```
@RequestMapping("/controlador")
public String controlador(RedirectAttributes atributos) {

    atributos.addFlashAttribute("atributo", "mensaje al usuario")

    return "redirect:/vista";
}
```

En la Vista

```
<etiqueta th:text="${atributo}"></etiqueta>
```



En la plantilla

Se incluye las alertas para diferentes tipos de mensajes.

```
<!-- MENSAJES -->

<div class="alert alert-success alert-dismissible"
  th:if="${success != null}">
  <label th:text="${success}"></label>
  <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>

<div class="alert alert-danger alert-dismissible"
  th:if="${error != null}">
  <label th:text="${error}"></label>
  <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>

<div class="alert alert-warning alert-dismissible"
  th:if="${warning != null}">
  <label th:text="${warning}"></label>
  <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>

<div class="alert alert-info alert-dismissible"
  th:if="${info != null}">
  <label th:text="${info}"></label>
  <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>
```



Alerta al grabar el producto

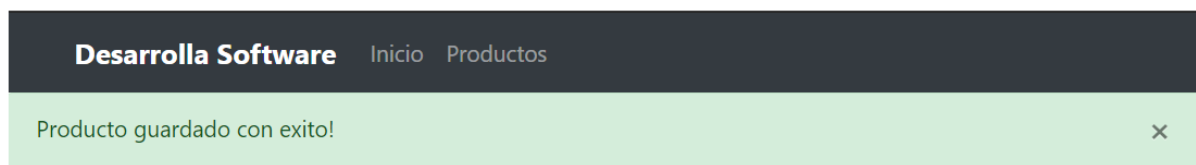
```
@PostMapping("/grabar")
public String guardar(@Valid @ModelAttribute Producto producto,
BindingResult result, Model model, RedirectAttributes atributos) {

    if (result.hasErrors()) {
        System.err.println("Se presentaron errores en el formulario!");
        String titulo = "NUEVO PRODUCTO";
        if (producto.getId() > 0) {
            titulo = "EDITAR PRODUCTO (" + producto.getId() + ")";
        }
        List<Categoria> listaCategorias = categoriaService.listarTodos();
        model.addAttribute("titulo", titulo);
        model.addAttribute("producto", producto);
        model.addAttribute("categorias", listaCategorias);
        return "/productos/frmEditor";
    }

    productoService.grabar(producto);
    System.out.println("Producto grabado con exito!");
    atributos.addFlashAttribute("success", "Producto guardado con exito!");

    return "redirect:/productos/";
}
```

El resultado es el siguiente:





Al editar un producto

```
@GetMapping("/editar/{id}")
public String editar(@PathVariable("id") Long idProd, Model model,
    RedirectAttributes atributos) {

    // Id de producto invalido
    if( idProd <= 0 ) {
        atributos.addFlashAttribute("error", "El id del producto es incorrecto.");
        return "redirect:/productos/todos";
    }

    Producto producto = productoService.buscarPorId(idProd);

    // Id de producto no existe
    if( producto == null ) {
        atributos.addFlashAttribute("error", "El id del producto no existe.");
        return "redirect:/productos/todos";
    }

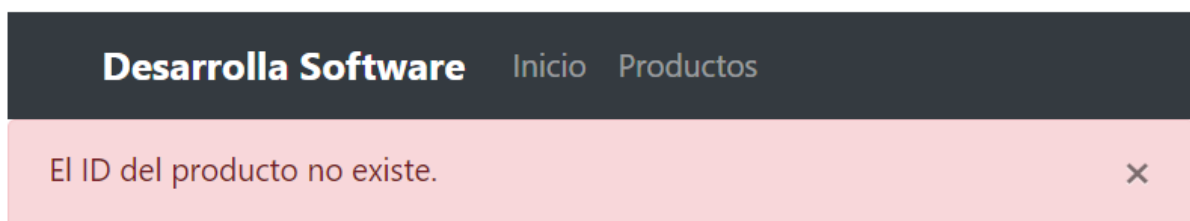
    List<Categoria> listaCategorias = categoriaService.listarTodos();
    model.addAttribute("titulo", "EDITAR PRODUCTO (" + idProd + ")");
    model.addAttribute("producto", producto);
    model.addAttribute("categorias", listaCategorias);

    return "/productos/frmEditar";
}
```

Para probar su funcionamiento, debes intentar editar un producto que no existe:

```
http://localhost:9090/productos/editar/7777
```

A continuación, tienes el resultado:





Al eliminar un producto

```
@GetMapping("/eliminar/{id}")
public String eliminar(@PathVariable("id") Long idProd,
RedirectAttributes atributos) {

    // Id de producto invalido
    if (idProd <= 0) {
        atributos.addFlashAttribute("error", "El ID del producto es incorrecto.");
        return "redirect:/productos/todos";
    }

    Producto producto = productoService.buscarPorId(idProd);

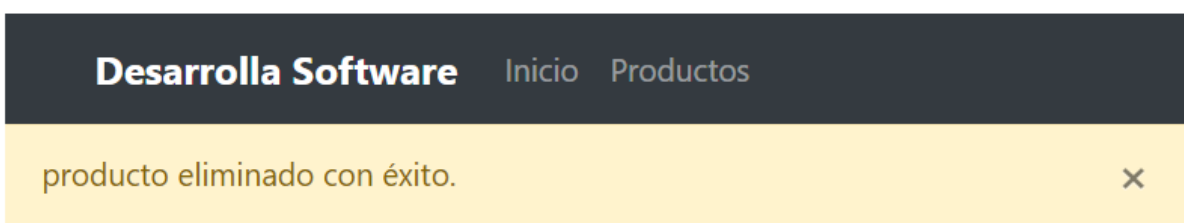
    // Id de producto no existe
    if (producto == null) {
        atributos.addFlashAttribute("error", "El ID del producto no existe.");
        return "redirect:/productos/todos";
    }

    productoService.eliminar(idProd);
    System.out.println("Producto eliminado con exito!");
    atributos.addFlashAttribute("warning", "producto eliminado con éxito.");
    return "redirect:/productos/";
}
```

Para probar su funcionamiento, debes intentar editar un producto existente:

```
http://localhost:9090/productos/editar/1020
```

A continuación, tienes el resultado:



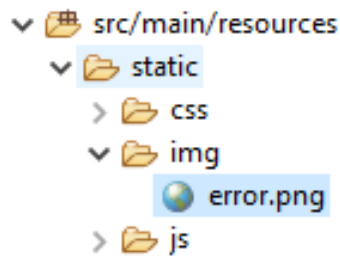


OTROS MENSAJES DE ERROR

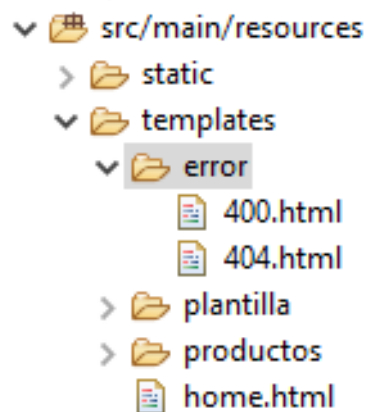
Estructura

Imagen de error

En la carpeta **img** debes agregar tu imagen para las páginas de error:



Estructura de carpetas



Puedes incluir un mensaje en las páginas 400.html y 404.html para probar el funcionamiento de estas páginas.



Este sería el resultado:

Desarrolla Software Inicio Productos

Bad Request ×

400 - Bad Request

 Ha realizado una solicitud incorrecta!

Copyright Desarrolla Software ©2020




Este sería el resultado:

Desarrolla Software Inicio Productos

Not Found ×

404 - Not Found

 La página solicitada no existe!

Copyright Desarrolla Software ©2020



EJERCICIO PROPUESTO

Desarrollar el CRUD de la tabla Empleado o un caso de los sistemas a su cargo en la empresa.



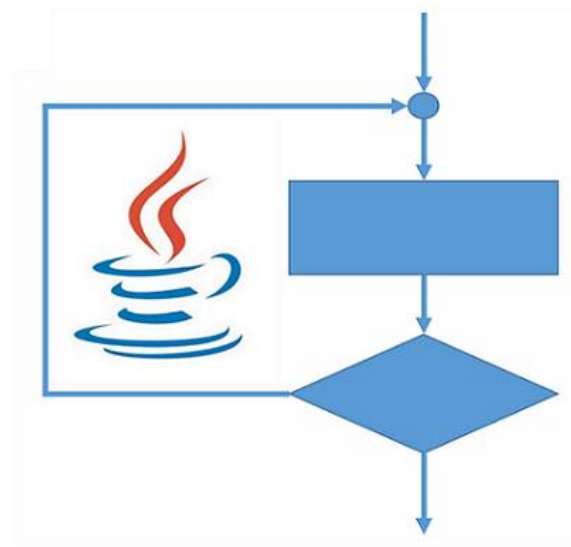
CURSOS VIRTUALES

Cupones

En esta URL se publican cupones de descuento:

<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.

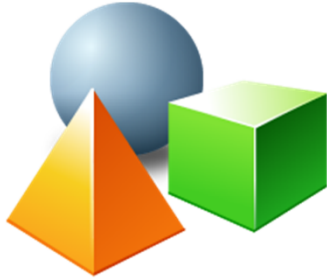
URL del Curso: <https://n9.cl/gcoronelc-java-fund>

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

I N S T R U C T O R

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

I N S T R U C T O R

En este curso aprenderás a programar 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>

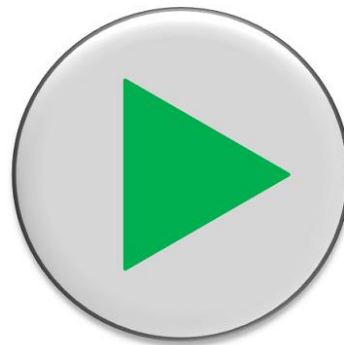
Avance del curso: <https://bit.ly/2vatZOT>

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



Programación con Oracle PL/SQL

ORACLE PL/SQL



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

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

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

Avance del curso: <https://bit.ly/3bcqYb>

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