**SPRING FRAMERWORK**

**Consfiguracion del archivo application.properties por application.yml**

server:

port: 9876

spring:

datasource:

url: jdbc:mysql://localhost:3306/db\_springboot\_backend?useSSL=false

username: pruebas

password: 790821Jason

driver-class-name: com.mysql.jdbc.Driver

jpa:

database-platform: org.hibernate.dialect.MySQL5InnoDBDialect

hibernate:

ddl-auto: create-drop

logging:

level:

org:

hibernate:

SQL: debug

Aqui es muy importante saber que estas configuraciones pueden ralizar cambios en la base de datos

ddl-auto: create-drop = borrar toda las tablas cuando se apaga el servidor y reconstruirlas al iniciar el proyecto (para ambientes de desarrollo)

ddl-auto: none = no realiza ninguna acción se asume que ya están creadas las tablas y structura desde antes de levantar la aplicación

ddl-auto: create = se crea cuando se levanta la aplicación pero si se baja el servicio no se elimina.

ddl-auto: update = crea la estructura la primera vez y luego simplemente va actualizando en cada cambio de las clases entities

ddl-auto: validate =

Creando entidades

package com.testing.desarrollo.model.entity;

import java.io.Serializable;

import java.util.Date;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

@Entity

@Table(name="clientes")

public class Cliente implements Serializable { // Vamos a crear como buena practica para el mapeo la interfaz serializabl

// Ojo si trabajamos con formularios esta clase serializable s mas practica de pasar igual

// que en metodos de session

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long id;

private String nombre;

private String apellido;

private String email;

@Column(name="create\_at")

@Temporal(TemporalType.DATE)

private Date createAt;

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 String getApellido() {

return apellido;

}

public void setApellido(String apellido) {

this.apellido = apellido;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Date getCreateAt() {

return createAt;

}

public void setCreateAt(Date createAt) {

this.createAt = createAt;

}

private static final long serialVersionUID = 1L;

}

**Creado la capa DAO implementando** CrudRepository

**package** com.testing.desarrollo.model.Dao;

**import** org.springframework.data.repository.CrudRepository;

**import** com.testing.desarrollo.model.entity.Cliente;

**public** **interface** IClienteDao **extends** CrudRepository<Cliente, Long> {

}

Documentacion CrudRepository: <https://docs.spring.io/spring-data/jpa/docs/2.1.8.RELEASE/reference/html/#repositories.core-concepts>

Documentacion Creacion de querys por nombre: <https://docs.spring.io/spring-data/jpa/docs/2.1.8.RELEASE/reference/html/#jpa.query-methods.named-queries> Buscar *Example 16. Query creation from method names*

*Documentacion Creacion de querys personalizados:* <https://docs.spring.io/spring-data/jpa/docs/2.1.8.RELEASE/reference/html/#jpa.query-methods.at-query>*:*

***Implementacion de capaservicio***

*Interfaz*

**package** com.testing.desarrollo.model.service;

**import** com.testing.desarrollo.model.entity.Cliente;

**public** **interface** IClienteService { //Aqui llamamos los metodos del

//CrudRepository que utilizare

**public** Iterable<Cliente> findAll();

}

Implementacion del servicio:

**package** com.testing.desarrollo.model.service;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.testing.desarrollo.model.Dao.IClienteDao;

**import** com.testing.desarrollo.model.entity.Cliente;

@Service

**public** **class** ClienteServiceImpl **implements** IClienteService {

@Autowired

**private** IClienteDao clienteDao;//Inyectamos CrudRepository desde IClienteDao

@Override

@Transactional(readOnly = **true**) // sobro escribe la transaccion de CrudRepository

//para hacerlo de una forma mas explicita

**public** Iterable<Cliente> findAll() {

**return** clienteDao.findAll();

}

}

**Implementacion Capa Controller**

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.testing.desarrollo.model.entity.Cliente;

**import** com.testing.desarrollo.model.service.IClienteService;

@RestController

@RequestMapping("/api")

**public** **class** ClienteRestController {

@Autowired

IClienteService clienteService;

@GetMapping("/clientes")

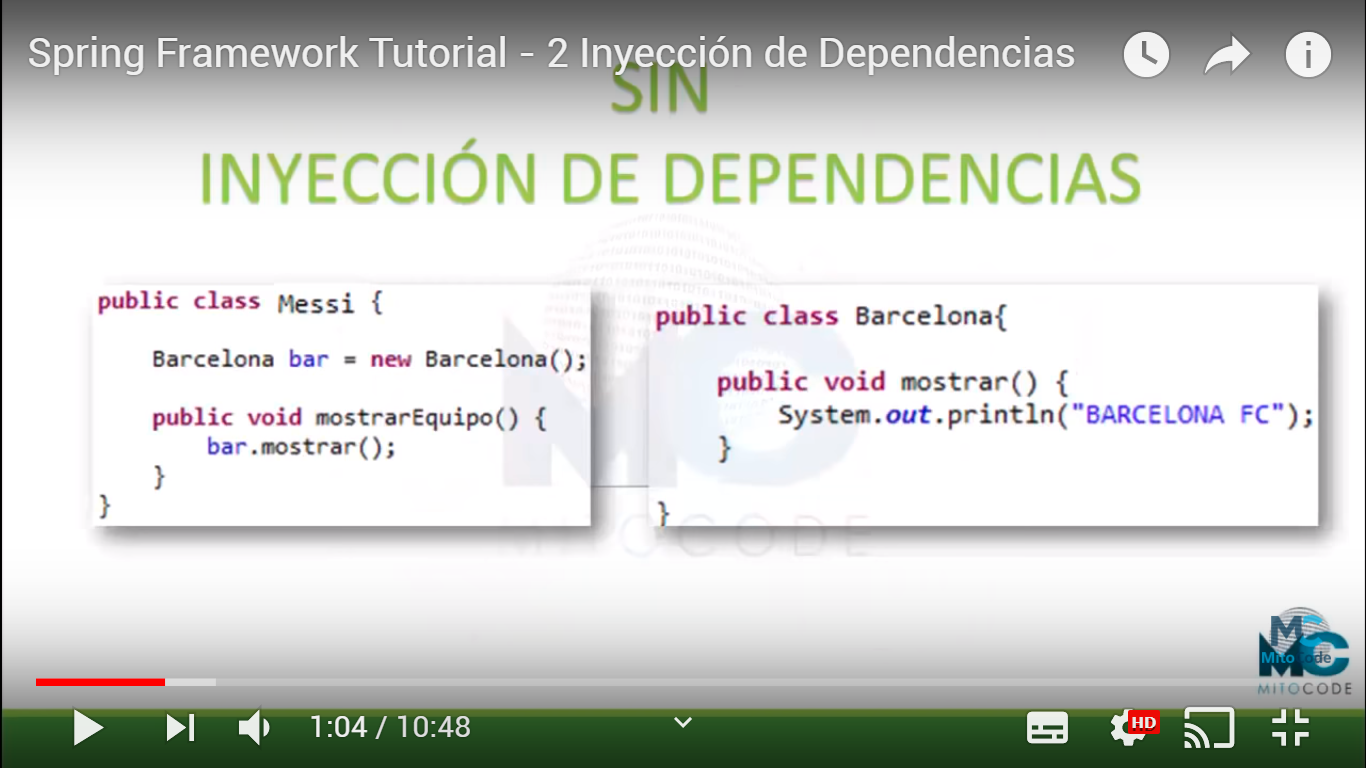
**public** Iterable<Cliente> index() {

**return** clienteService.findAll();

}

}

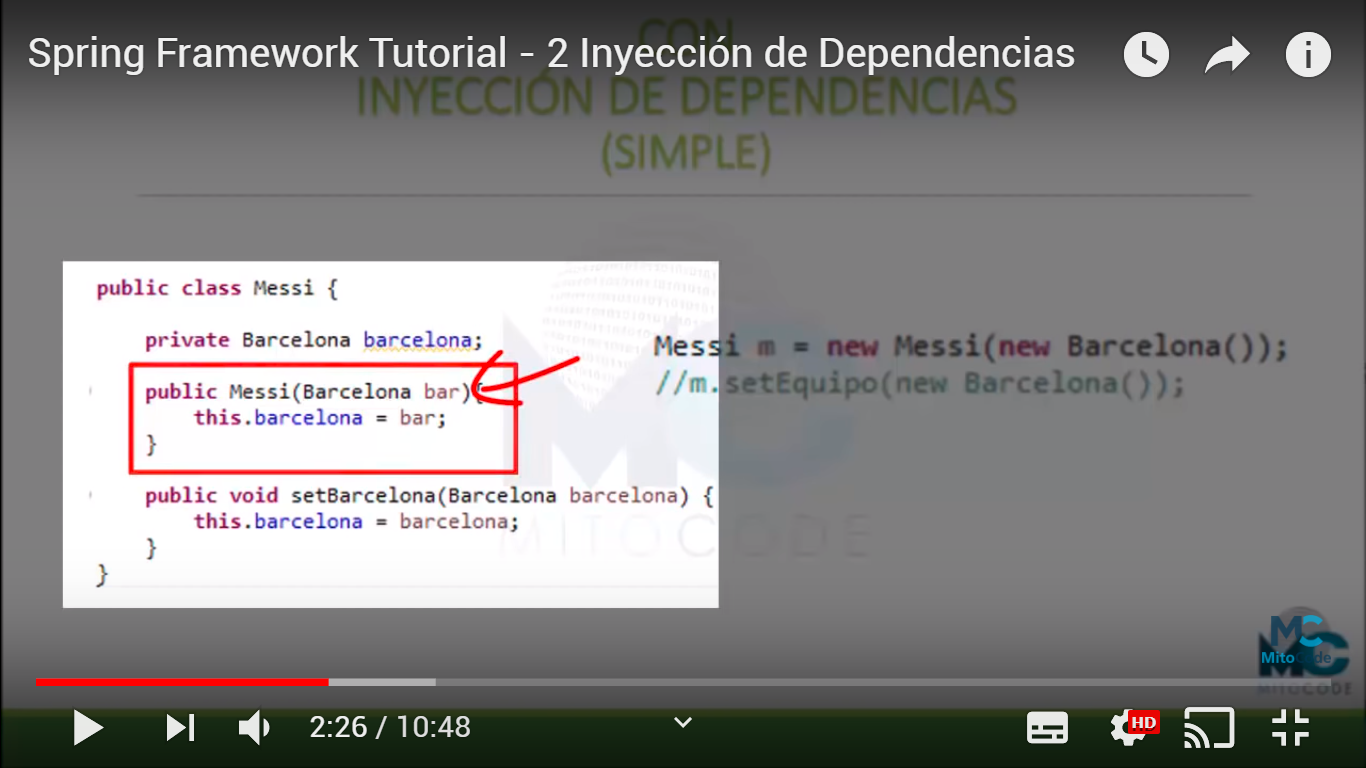
**Ejemplo sin inyeccion de dependencias**

**problemas:**

**1** mesi no puede cambiar de equipo con esta forma

**2** Imposible la reutilización de código toca es repetir código

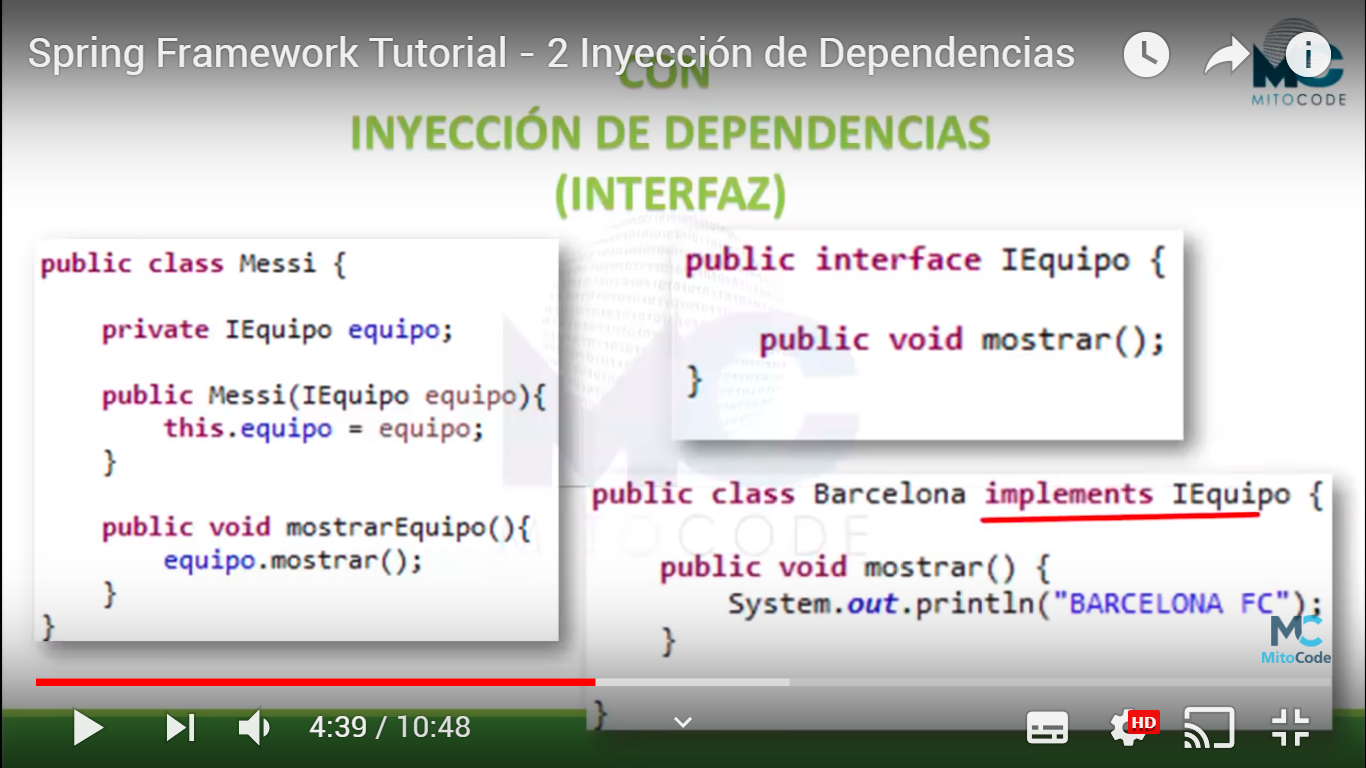
**Con Inyecion de dependencias**



Se realiza aquí por constructor y método set

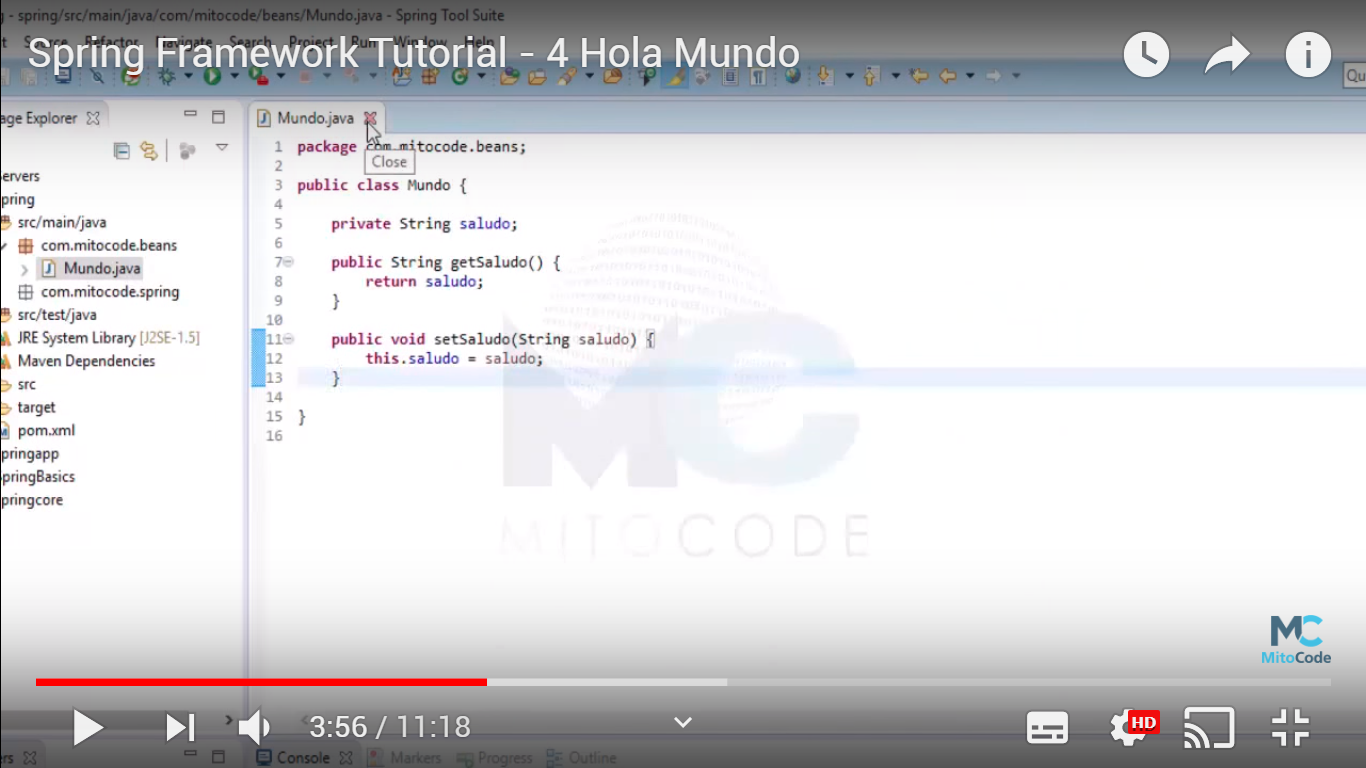
Problemas: aun mesi no puede cambiar de equipo

**Con Inyección de dependencias implementando una interfaz**

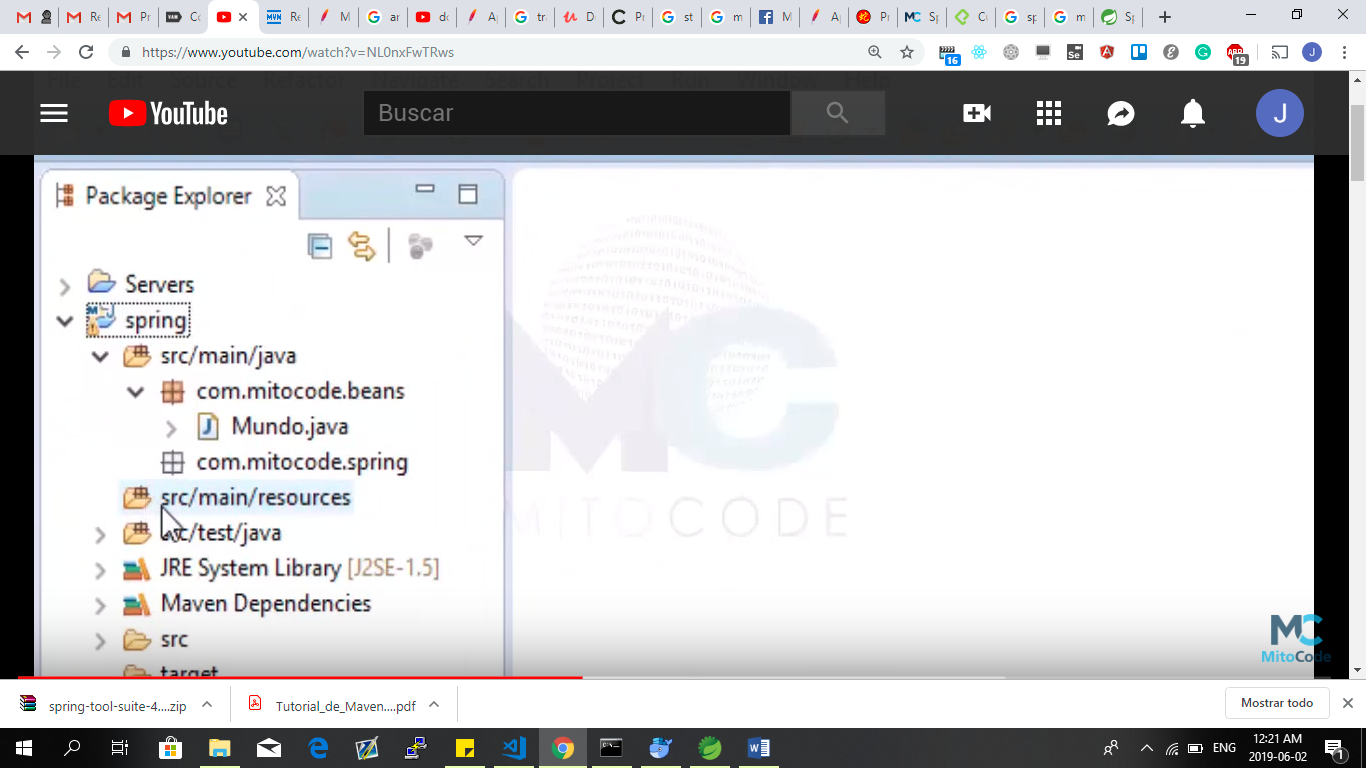


2- **IMPLEMENTAR LA INYECCIÓN DE DEPENDENCIAS YA SEA POR XML O ANOTACIONES**

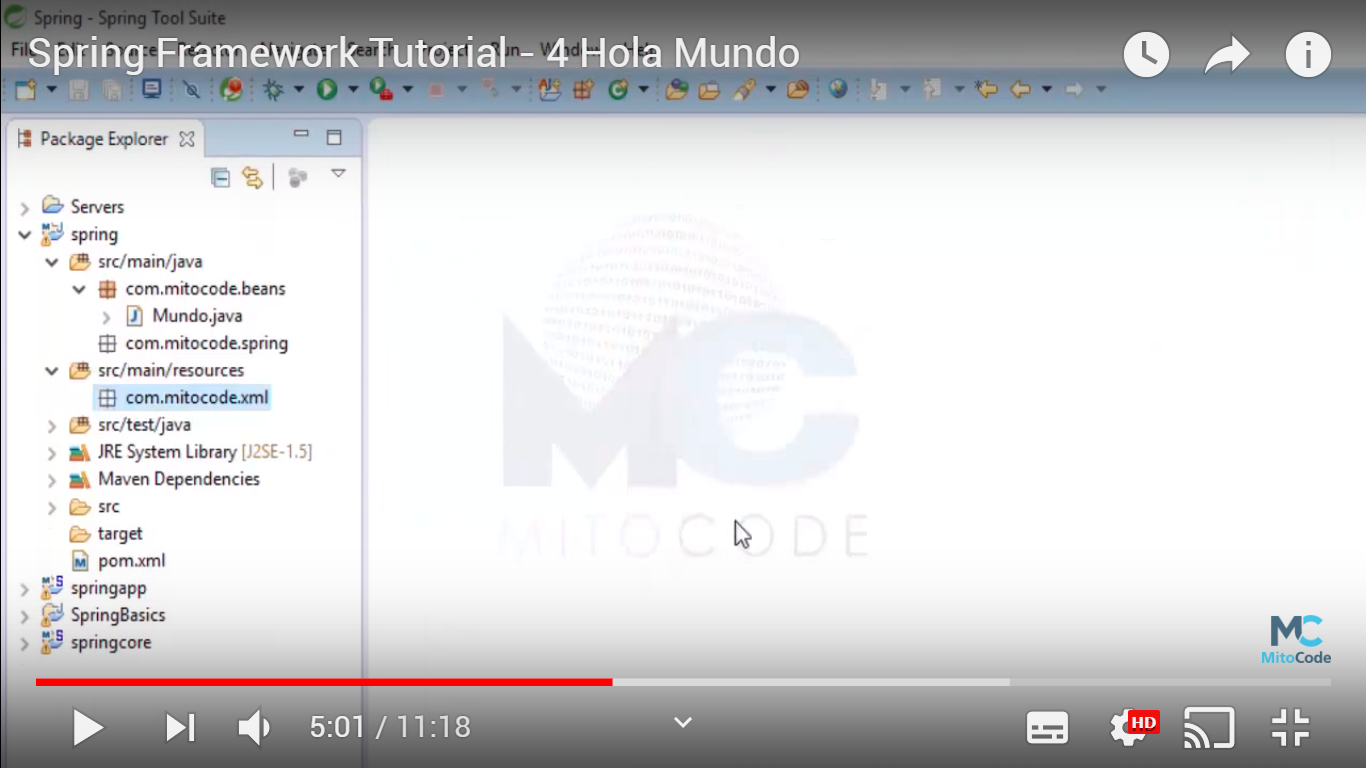
**Crear entidad pojo**



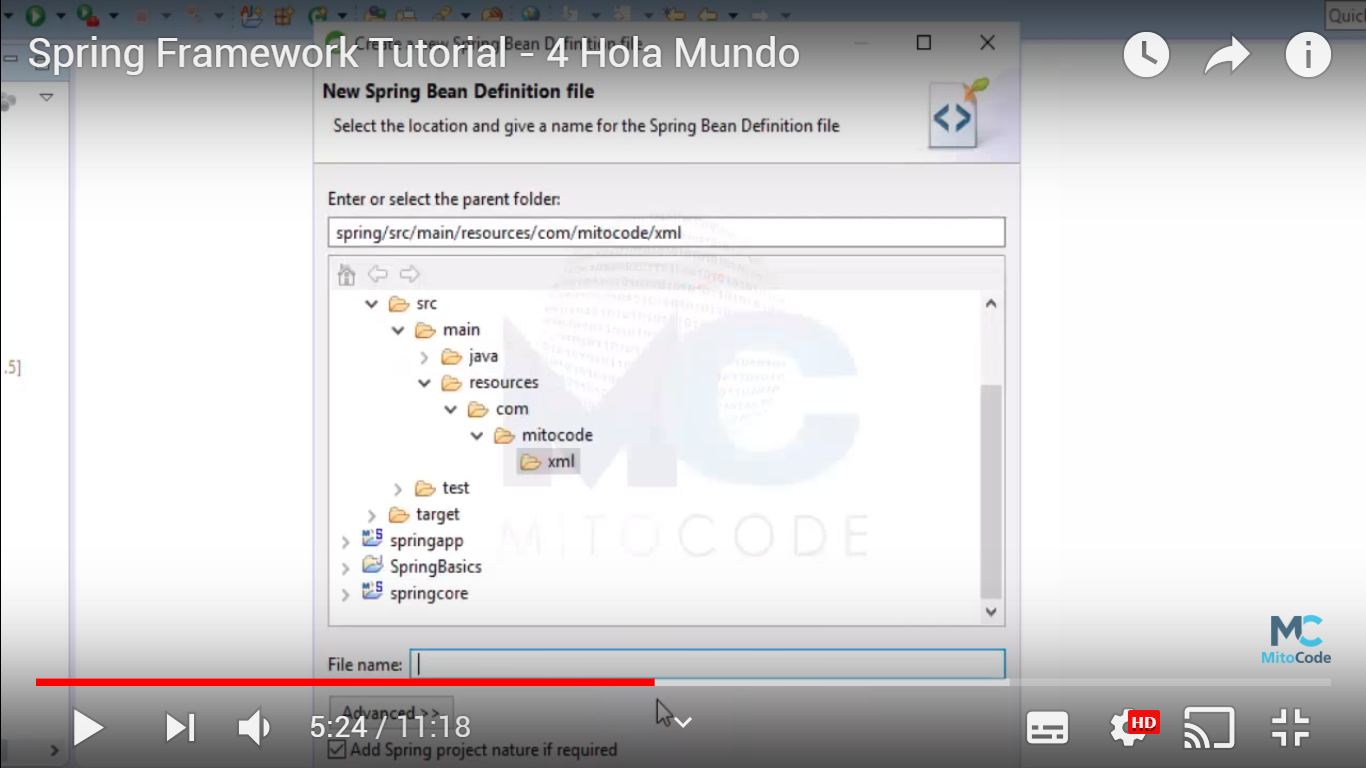
Luego creamos una carpeta en la raíz del proyecto src/main/resources



Luego dentro de esta creamos un paquete el cual contendrá el xml



Damos clic en new -> otros -> spring -> spring bean definition file y seleccionamos la carpeta xml



Crea un archivo con os namespace necesarios de spring para la gestión de los beans en xml



ResponseEntity

