# [Microservicios con Spring Boot, Cloud y Docker](https://www.udemy.com/course/microservicios-spring-boot-cloud/)

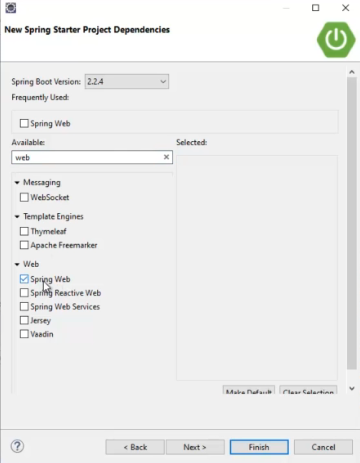
**12. Ejercicio práctico mapeado de objetos a JSON**

Microservicio\_cursos es una pequeña aplicación que es generada con spring starter Project,

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Y los starter solo el spring web



**Se crea la clase Curso dentro del paquete com.curos.model**

public class Curso {

private String nombre;

private int duracion;

private String horario;

public Curso(String nombre, int duracion, String horario) {

super();

this.nombre = nombre;

this.duracion = duracion;

this.horario = horario;

}

public Curso() {

super();

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public int getDuracion() {

return duracion;

}

public void setDuracion(int duracion) {

this.duracion = duracion;

}

public String getHorario() {

return horario;

}

public void setHorario(String horario) {

this.horario = horario;

}

}

**Y se crea la case CursosController dentro del paquete controllers**

@RestController

public class CursosController {

@GetMapping(value="curso", produces=MediaType.APPLICATION\_XML\_VALUE)

public Curso getCurso() {

return new Curso("Java",100,"mañana");

}

}

**Finalmente para que el controller funcione se debe declarar en Application.java**

package com.cursos;

import org.springframework.boot.SpringApplication;

*@*SpringBootApplication

@ComponentScan(basePackages= {"controllers"})

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

Para ejecutarla es con Run As / Spring Boot App

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Y para probarla en un navegador se pone [http://localhost:8080/curso](http://localhost:8080/cursos)

Y solo devuelve el registro en formato JSON



Se complementa la clase para que muestre un listado de cursos

@RestController

public class CursosController {

private List<Curso> cursos;

@PostConstruct

public void init() {

cursos= new ArrayList<>();

cursos.add(new Curso("Spring",25,"tarde"));

cursos.add(new Curso("Spring boot",20,"tarde"));

cursos.add(new Curso("Phyton",30,"tarde"));

cursos.add(new Curso("Java EE",50,"fin de semana"));

cursos.add(new Curso("Java Basico",30,"mañana"));

}

//DEVUELVE LA LISTA DE TODOS LOS CURSOS

//para XML : MediaType.APPLICATION\_XML\_VALUE

//Para JSON : MediaType.APPLICATION\_JSON\_VALUE

@GetMapping(value="cursos", produces=MediaType.APPLICATION\_JSON\_VALUE)

public List<Curso> getCursos(){

return cursos;

}

//DEVUELVE UN CURSO

@GetMapping(value="curso", produces=MediaType.APPLICATION\_XML\_VALUE)

public Curso getCurso() {

return new Curso("Java",100,"mañana");

}

//DEVUELVE LOS DATOS DEL CURSO QUE LE PASEMOS X NOMBRE

@GetMapping(value="cursos/{name}", produces=MediaType.APPLICATION\_JSON\_VALUE)

public List<Curso> buscarCursos(@PathVariable("name") String nombre){

List<Curso> aux = new ArrayList<>();

for(Curso c:cursos) {

if(c.getNombre().contains(nombre)) {

aux.add(c);

}

}

return aux;

}

}

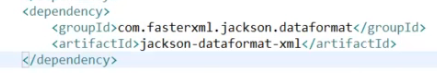
Y para probarla en un navegador se pone <http://localhost:8080/cursos>

Texto

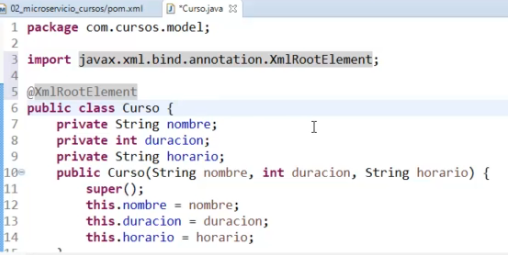
Descripción generada automáticamente

**13. Ejercicio práctico mapeado de objetos a XML**

Si se requiere regresar la informacion en formato XML se debe agregar una dependencia la pom



Despues en e bean Cursoljava se pone la anotacion @XmlRootElement



Y en el contoloer sa cambia:

