ESCANEO AUTOMÁTICO DE COMPONENTES

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
        http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd">

    <context:component-scan base-package="com.openwebinars.springmvc" />

    <mvc:annotation-driven />

    <bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass"
            value="org.springframework.web.servlet.view.JstlView" />
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>



</beans>
```

ESQUEMA BÁSICO DE UN CONTROLADOR

`@Controller`

Estereotipo que sirve para indicar:

- Que la clase anotada es un bean
- Que tiene el rol de controlador

`@RequestMapping`

Anotación que nos permite:

- Indicar el tipo de petición HTTP
- Indicar la URL en la cual se va a escuchar

USO DE @Controller y @RequestMapping

```java
@Controller
@RequestMapping("/appointments")
public class AppointmentsController {

    private final AppointmentBook appointmentBook;

    @Autowired
    public AppointmentsController(AppointmentBook appointmentBook) {
        this.appointmentBook = appointmentBook;
    }
```

USO DE @Controller y @RequestMapping

```java
@RequestMapping(method = RequestMethod.GET)
public Map<String, Appointment> get() {
    return appointmentBook.getAppointmentsForToday();
}

@RequestMapping(path = "/{day}", method = RequestMethod.GET)
public Map<String, Appointment> getForDay(@PathVariable @DateTimeFormat(iso=ISO.DATE) Date day, Model model) {
    return appointmentBook.getAppointmentsForDay(day);
}

@RequestMapping(path = "/new", method = RequestMethod.GET)
public AppointmentForm getNewForm() {
    return new AppointmentForm();
}
```

USO DE @Controller y @RequestMapping

```
@RequestMapping(method = RequestMethod.POST)
public String add(@Valid AppointmentForm appointment, BindingResult result) {
    if (result.hasErrors()) {
        return "appointments/new";
    }
    appointmentBook.addAppointment(appointment);
    return "redirect:/appointments";
}
```

USO DE @Controller y @RequestMapping

```java
@Controller
public class ClinicController {

    private final Clinic clinic;

    @Autowired
    public ClinicController(Clinic clinic) {
        this.clinic = clinic;
    }

    @RequestMapping("/")
    public void welcomeHandler() {
    }

    @RequestMapping("/vets")
    public ModelMap vetsHandler() {
        return new ModelMap(this.clinic.getVets());
    }
}
```

VARIANTES COMPUESTAS DE @RequestMapping

- `@GetMapping`
- `@PostMapping`
- `@PutMapping`
- `@DeleteMapping`
- `@PatchMapping`

VARIANTES COMPUESTAS DE @RequestMapping: @GetMapping

```java
@RequestMapping(path = "/{day}", method = RequestMethod.GET)
public Map<String, Appointment> getForDay(@PathVariable @DateTimeFormat(iso=ISO.DATE) Date day, Model model) {
    return appointmentBook.getAppointmentsForDay(day);
}

@RequestMapping(path = "/new", method = RequestMethod.GET)
public AppointmentForm getNewForm() {
    return new AppointmentForm();
}
```

```java
@GetMapping
public Map<String, Appointment> get() {
    return appointmentBook.getAppointmentsForToday();
}

@GetMapping("/{day}")
public Map<String, Appointment> getForDay(@PathVariable @DateTimeFormat(iso=ISO.DATE) Date day, Model model) {
    return appointmentBook.getAppointmentsForDay(day);
}
```

VARIANTES COMPUESTAS DE @RequestMapping: @PostMapping

```
@RequestMapping(method = RequestMethod.POST)
public String add(@Valid AppointmentForm appointment, BindingResult result) {
    if (result.hasErrors()) {
        return "appointments/new";
    }
    appointmentBook.addAppointment(appointment);
    return "redirect:/appointments";
}
```

```
@PostMapping
public String add(@Valid AppointmentForm appointment, BindingResult result) {
    if (result.hasErrors()) {
        return "appointments/new";
    }
    appointmentBook.addAppointment(appointment);
    return "redirect:/appointments";
}
```