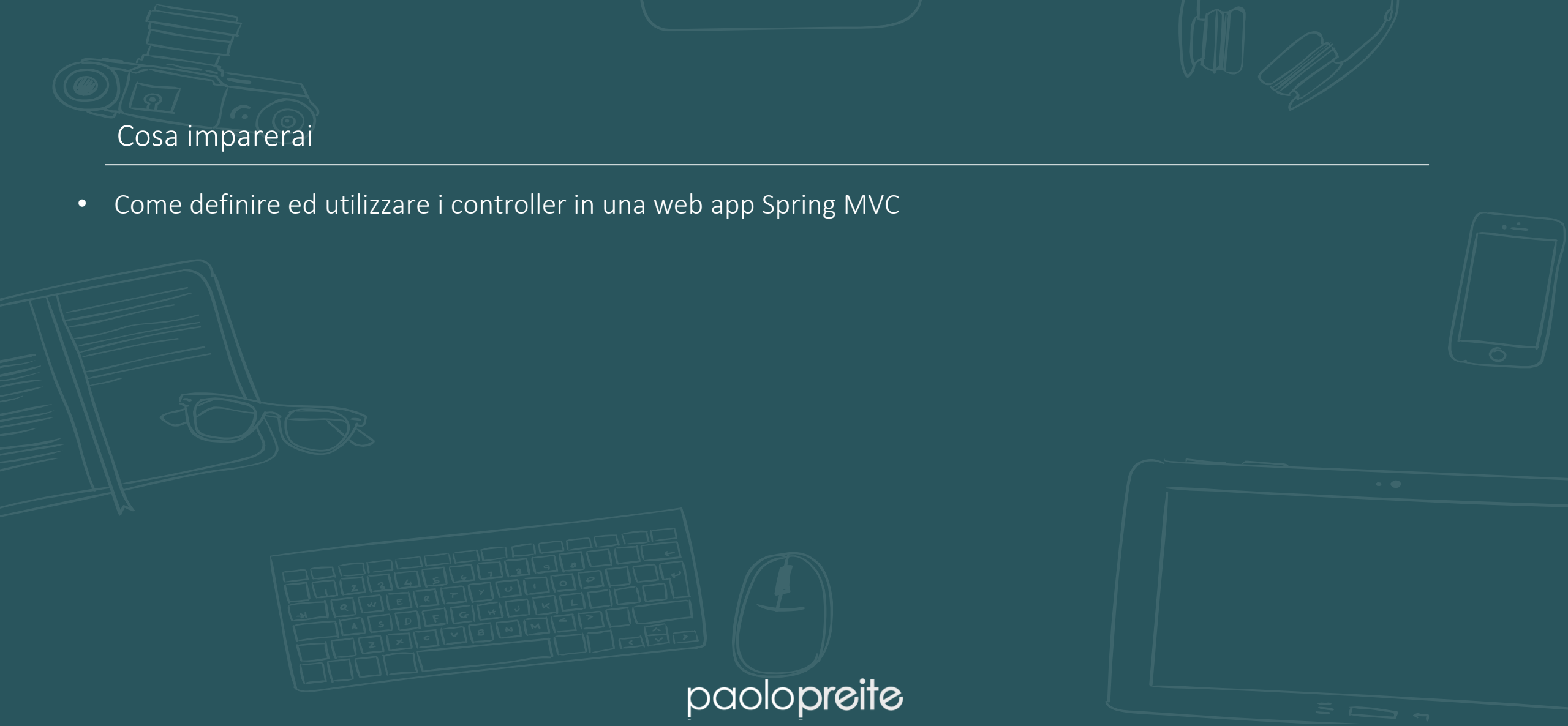
Gestione dei controller – Parte III

Cosa imparerai

- Come definire ed utilizzare i controller in una web app Spring MVC



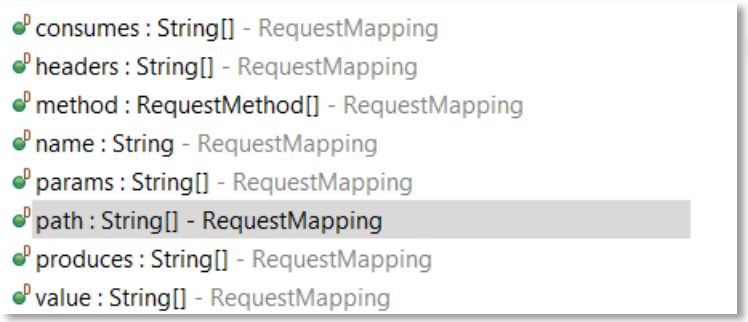
paolopreite

Gestione dei controller

Mapping delle URL

L'annotation ha vari attributi che consentono di specificare:

- **path**: i path a cui sarà associato il metodo o la classe
- **method**: il metodo HTTP (GET, POST, PUT,…)
- **params**: i parametri di richiesta
- **headers**: le intestazioni della richiesta
- **consumes** e **produces**: i tipi di media



Gestione dei controller

Mapping delle URL – l'attributo «method»

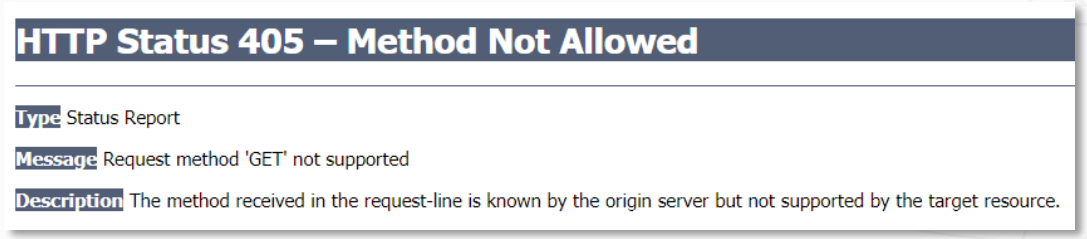
L'annotation *@RequestMapping*, per impostazione predefinita, risponde a tutti i metodi HTTP (GET, POST, …).

Attraverso l'attributo *method* si può specificare i metodi HTTP a cui dovrà rispondere la classe o il metodo annotato con *@RequestMapping*.

```
@Controller
@RequestMapping(path = {"/", "v1"})
public class HelloController {
    @RequestMapping(path = "/hello", method = RequestMethod.POST)
    @ResponseBody
    public String hello() {
        return "hello";
    }

    @RequestMapping(path = "/bye", method = {RequestMethod.GET, RequestMethod.POST})
    @ResponseBody
    public String bye() {
        return "bye";
    }
}
```

Se invocato via GET



Gestione dei controller

Mapping delle URL – l'attributo «params»

Consente di specificare parametri e valori (sotto forma di stringa **parametro=valore**) ammessi nell'HTTP request, in query string.

È possibile utilizzare anche la notazione **parametro!=valore** per accettare URL con valori differenti.

Facciamo qualche esempio…

```
@RequestMapping(
        path = "/bye",
        method = {RequestMethod.GET},
        params = {"nome=Paolo", "cognome=Preite"})
@ResponseBody
public String bye() {
    return "bye ";
}
```

*http://mysite.com/hmyapp/bye**?nome=Paolo&cognome=Preite -> HTTP Status 200***

*http://mysite.com/hmyapp/bye**?nome=Paolo&cognome=Rossi -> HTTP Status 400 – Bad Request***

```
@RequestMapping(
        path = "/bye",
        method = {RequestMethod.GET},
        params = {"nome=Paolo", "cognome!=Preite"})
@ResponseBody
public String bye() {
    return "bye ";
}
```

*http://mysite.com/hmyapp/bye**?nome=Paolo&cognome=Preite -> HTTP Status 400 – Bad Request***

*http://mysite.com/hmyapp/bye**?nome=Paolo&cognome=Rossi -> HTTP Status 200***

Gestione dei controller

Mapping delle URL – l'attributo «headers»

Consente di definire dei parametri nell'header della request.

```
@RequestMapping(
        path = "/bye",
        method = {RequestMethod.GET},
        params = {"nome=Paolo", "cognome=Preite"},
        headers = {"Accept=application/json"})
@ResponseBody
public String bye() {
    return "bye ";
}
```

Mapping delle URL – l'attributo «produces»

Consente di specificare il Content-Type del contenuto della response.

```
@RequestMapping(
        path = "/bye",
        method = {RequestMethod.GET},
        params = {"nome=Paolo", "cognome=Preite"},
        produces = {"application/json"})
@ResponseBody
public String bye() {
    return "bye";
}
```


```
@RequestMapping(
        path = "/bye",
        method = {RequestMethod.GET},
        params = {"nome=Paolo", "cognome=Preite"},
        produces = {MediaType.APPLICATION_JSON_VALUE})
@ResponseBody
public String bye() {
    return "bye";
}
```

Gestione dei controller

Mapping delle URL – l'attributo «consumes»

Consente di specificare il Content-Type del contenuto della request.

```
@RequestMapping(
        path = "/bye",
        method = {RequestMethod.GET},
        params = {"nome=Paolo", "cognome=Preite"},
        consumes = {MediaType.APPLICATION_JSON_VALUE})
@ResponseBody
public String bye() {
    return "bye";
}
```

Di cosa abbiamo parlato in questa lezione

- Come definire ed utilizzare i controller in una web app Spring MVC