Gestione dei controller – Parte IV

Cosa imparerai

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





paolopreite

Gestione dei controller

Creiamo un controller Spring MVC

I passi per creare un controller:

1. Creare una classe ed annotarla con *@Controller*
2. Creare almeno un metodo nella classe ed annotarlo con *@RequestMapping*, specificando il path (o i path) a cui dovrà rispondere. Nel caso più semplice il metodo ritorna una stringa.

*In questo caso, grazie all'annotation **@ResponseBody**, la stringa di ritorno costituirà l'output inserito nella response*

```
@Controller
public class HelloController {
    @ResponseBody
    @RequestMapping("/")
    public String showForm() {
        return "hello";
    }
}
```

```
@Controller
public class HelloController {
    @RequestMapping("/")
    public String showForm() {
        return "hello";
    }
}
```

In questo caso la stringa di ritorno costituirà il nome della view che conterrà l'HTML da inserire come output della response

3. Creare una classe ed annotarla con *@Configuration* e *@ComponentScan*, specificando il package dove si trova il controller.

```
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = { "it.test.web.controller" })
public class WebConfig  {
```

Gestione dei controller

Mapping delle URL

Per mappare uno o più path di una URL ad un controller è possibile utilizzare l'annotation **@RequestMapping**.

L'annotation può essere utilizzata:

- **A livello di classe**: utilizzata se vogliamo mappare dei path condiviso tra più metodi
- **A livello di metodo**: utilizzata per mappare dei path su un metodo

```
@RestController
@RequestMapping(path = "/rest")
public class HelloRestController {
    @RequestMapping(path = "/1")
    public String showForm() {
        return "CIAO!!!!";
    }

    @RequestMapping(path = "/2")
    public String showForm2() {
        return "CIAO 2!!!!";
    }
}
```

http://mysite.com/rest/1

http://mysite.com/rest/2

Gestione dei controller

Mapping delle URL - altre annotation…

Per mappare un path di una URL ai metodi di un controller è possibile utilizzare anche varianti di scorciatoie specifiche per il metodo HTTP in sostituzione dell'annotation @RequestMapping.

In particolare abbiamo:

- **@GetMapping**: usata per definire un metodo accessibile solo via GET
- **@PostMapping**: usata per definire un metodo accessibile solo via POST
- **@PutMapping**: usata per definire un metodo accessibile solo via PUT
- **@DeleteMapping**: usata per definire un metodo accessibile solo via DELETE
- **@PatchMapping**: usata per definire un metodo accessibile solo via PATCH

Queste annotation sono da preferire a @RequestMapping sui metodi perché identificano meglio il comportamento del metodo.

L'annotation @RequestMapping è sempre da utilizzare a livello di classe per definire i path condivisi.

Gestione dei controller

Alcuni esempi…

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

@GetMapping
@ResponseBody
public String hello2() {
    return "hello";
}
```

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

@PostMapping
@ResponseBody
public String hello2() {
    return "hello";
}
```

Di cosa abbiamo parlato in questa lezione

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