Introduzione alla creazione e gestione delle view

Cosa imparerai

- Come creare e gestire le viste (le pagine HTML) in una web app Spring MVC




Review del pattern MVC

Il pattern MVC

Il nome deriva dalle tre componenti principali di un'applicazione web:

- **Model**: la componente che modella i dati ricevuti dall'utente
- **View**: la componente che visualizza le informazioni all'utente (generalmente HTML)
- **Controller**: la componente che gestisce il reindirizzamento delle URL chiamate dall'utente verso le view

In questa sezione ci occupiamo della componente **View**.

Review del pattern MVC

Il diagramma seguente rappresenta il pattern MVC in una tipica applicazione web.

1. L'utente richiede una URL (es. www.mysite.com/**my-app**/home.html). La richiesta arriva al Web Container (es. Tomcat) che la indirizza alla web app di riferimento (ad es. my-app)

2. Nella web app che usano Spring MVC, c'è una servlet che si occupa di gestire e reindirizzare le chiamate (la DispatcherServlet). La servlet inoltra la richiesta al controller che mappa la URL richiesta

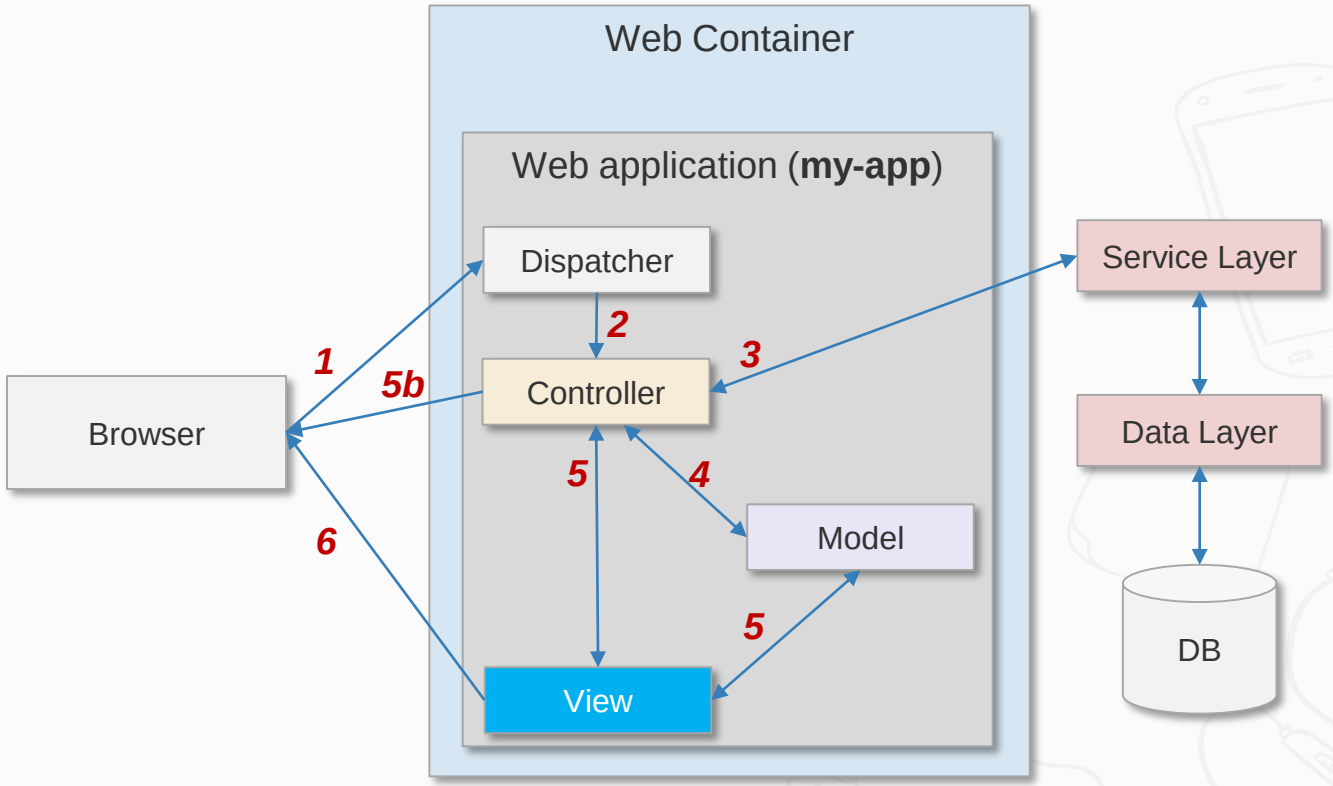
3. Il controller accede ai servizi per recuperare i dati richiesti

4. Il controller popola gli oggetti con i dati ricevuti dai servizi

5. Il controller inoltra la richiesta alla view, la quale utilizzerà i dati per creare i contenuti da inviare all'utente

6. L'output generato dalla view viene inviato all'utente

5b. Se la è asincrona (XMLHttpRequest), il controller non invoca la view ma ritorna i dati elaborati, in un formato diverso da HTML (XML, JSON,…)



Creazione e gestione delle view

Spring MVC consente di utilizzare diverse tecnologie per la realizzazione delle view:

- Freemarker
- Thymeleaf
- Groovy
- JSP
- JSTL
- …

Per utilizzare una delle tecnologie supportate, è sufficiente modificare opportunamente la configurazione della web app.

NOTA: è possibile utilizzare anche Angular come tecnologia per la realizzazione delle componenti di front-end di una web app. In questo caso, tuttavia, la web app Spring avrà solo dei componenti REST per implementare la logica di business.

Creazione e gestione delle view

Le view sono sostanzialmente dei file che contengono codice HTML ed utilizzano determinati tag o una certa sintassi per gestire i dati (tipicamente i dati di un form…).

Ogni view ha accesso a tutti i bean che si trovano nel context dell'applicazione.

Come fa Spring MVC a decidere quale view dovrà essere utilizzata per genera l'output?
Come sappiamo, un controller delega alle viste la generazione dell'output (codice HTML) da visualizzare all'utente.

Se un controller ritorna una stringa (purché che non inizi con *redirect* o *forward*) oppure un oggetto *ModelAndView*, vuol dire che l'output dovrà essere generato da una view.

Creazione e gestione delle view

Come viene idenfiticata la view che dovrà gestire la generazione dell'output?

Il controller utilizza un oggetto di tipo **ViewResolver** che identifica la view sulla base di uno dei seguenti casi:

- **il metodo ritorna String**: il valore della stringa ritornata dal metodo coincide con il nome della view
- **il metodo ritorna un oggetto di tipo View:** ed il nome della view è incapsulato nell'oggetto

In pratica…

1. **La View** identifica la risorsa che deve generare l'output da inviare all'utente (JSP, file Freemarker, ecc…)

2. **Il ViewResolver** identifica la View sulla base di una stringa.

Creazione e gestione delle view

Spring MVC non ha un ViewResolver di default. Infatti, se non specifichiamo il view resolver ed invochiamo una URL la nostra web app andrà in errore…

```java
@Controller
@RequestMapping(path = "/")
public class HelloController {
    @GetMapping(path = "/hello")
    public String hello() {
        return "redirect:/bye";
    }

    @GetMapping(path = "/helloForward")
    public String helloForward() {
        return "forward:/bye";
    }

    @GetMapping(path = "/bye")
    public String bye() {
        return "bye";
    }
}
```

Di cosa abbiamo parlato in questa lezione

- Come creare e gestire le viste (le pagine HTML) in una web app Spring MVC