La DispatcherServlet

Cosa imparerai

- Cos’è la DispatcherServlet e come lavora





Il front controller pattern

Prima di entrare nel dettaglio di **Spring MVC**, è necessario conoscere un design pattern molto utilizzato nella realizzazione di framework per applicazioni web:

Front Controller Pattern

Questo pattern prevede un punto di ingresso centralizzato per la gestione delle richieste.

Il punto d'ingresso può essere una Servlet Java, una pagina PHP ecc…

https://it.wikipedia.org/wiki/Front_Controller_pattern

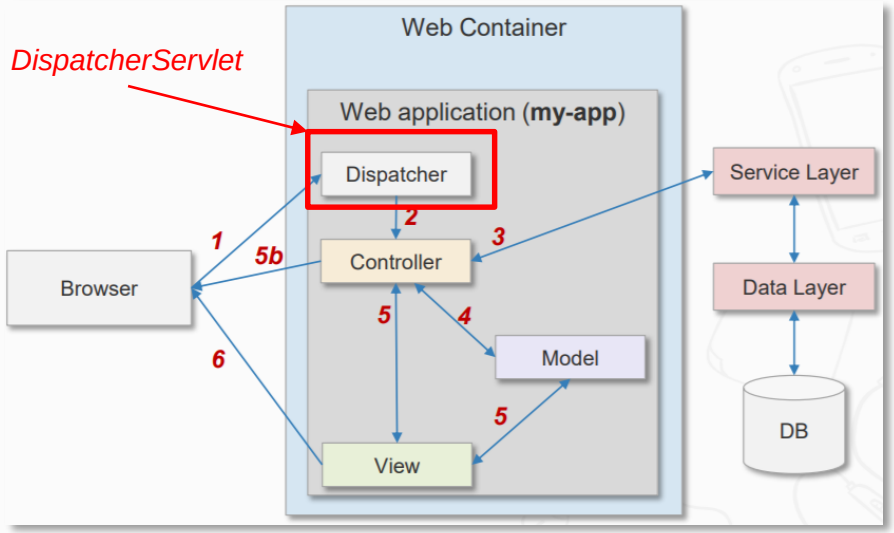
paolopreite

La DispatcherServlet

In **Spring MVC**, il «*front controller pattern*» è implementato attraverso una servlet chiamata **DispatcherServlet**.

La **DispatcherServlet** è una servlet messa a disposizione dal framework Spring MVC ed implementa un algoritmo per l'elaborazione delle richieste e si occupa di inoltrare la richiesta al controller di riferimento.

Il lavoro effettivo viene eseguito dal controller, dalle view, eccetera…



paolopreite

La DispatcherServlet

Per utilizzare la **DispatcherServlet**, come qualsiasi servlet, dobbiamo dichiararla e mapparla.

Possiamo utilizzare la configurazione Java o via web.xml.

La DispatcherServlet utilizza la configurazione Spring per conoscere quali sono i controller a cui delegare le richieste, le view per l'output della response ecc…

```java
public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class<?>[] { WebConfig.class };
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }

    @Override
    protected FrameworkServlet createDispatcherServlet(WebApplicationContext servletAppContext) {
        DispatcherServlet ds = new DispatcherServlet(servletAppContext);
        ds.setThrowExceptionIfNoHandlerFound(true);

        return ds;
    }

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return null;
    }
}
```

paolopreite

La DispatcherServlet

La DispatcherServlet utilizza un oggetto di tipo **WebApplicationContext** (un'estensione di ApplicationContext) per la propria configurazione.

```java
public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class<?>[] { WebConfig.class };
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }

    @Override
    protected FrameworkServlet createDispatcherServlet(WebApplicationContext servletAppContext) {
        DispatcherServlet ds = new DispatcherServlet(servletAppContext);
        ds.setThrowExceptionIfNoHandlerFound(true);

        return ds;
    }

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return null;
    }
}
```

Generalmente è sufficiente definire un unico WebApplicationContext.

Tuttavia è possibile avere una gerarchia di context in cui un WebApplicationContext radice è condiviso tra più DispatcherServlet (o altre Servlet).

paolopreite

La DispatcherServlet

Il DispatcherServlet delega a dei «**bean speciali**» l'elaborazione delle richieste e restituire la risposta al client.

Per «**bean speciali**» si intendono oggetti che implementano determinate funzioni. Alcuni «**bean speciali**» rilevati dalla DispatcherServlet sono:

- **HandlerExceptionResolver**: consente di gestire le eccezioni, la visualizzazione degli errori HTML, ecc…
- **ViewResolver**: permette di identificare una determinata View sulla base di una stringa
- **LocaleResolver, LocaleContextResolver**: gestisce le impostazioni sulla localizzazione utilizzate dal client
- **ThemeResolver**: gestisce i temi definiti nell'applicazione web per offrire layout personalizzati
- **MultipartResolver**: gestisce il parsing di una request di tipo multi-part (ad es. upload di file)

paolopreite

Di cosa abbiamo parlato in questa lezione

- Cos’è la DispatcherServlet e come lavora