Spring MVC : Configuration avec les classes (**Java Config**)

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

spring

Plan

Spring MVC 5

Rappel : **Spring** propose deux modes de configuration

- **XML Config**
- **Java Config**

Spring MVC 5

Rappel : **Spring** propose deux modes de configuration

- **XML Config**
- **Java Config**

Objectif de ce chapitre

Reconfigurer le projet du chapitre précédent avec des classes **Java**.

Spring MVC 5

Étapes

- Remplacer `servlet-context.xml` par une nouvelle classe (que nous appellerons, `ApplicationConfig`)
- Remplacer le contenu du `web.xml` par une classe qui étend la classe `AbstractAnnotationConfigDispatcherServletInitializer`
- Ne plus référencer le répertoire `spring` situé dans `WEB-INF`
- Supprimer le contenu du `web.xml` (garder seulement la balise `web-app`)

Créons la classe `ApplicationConfig` dans `org.eclipse.firstspringmvc.configuration`

```
@Configuration
@ComponentScan("org.eclipse.firstspringmvc.controller")
@EnableJpaRepositories("org.eclipse.firstspringmvc.dao")
@EnableTransactionManagement
public class ApplicationConfig {
  @Bean
  public InternalResourceViewResolver setup() {
    InternalResourceViewResolver viewResolver = new
       InternalResourceViewResolver();
    viewResolver.setPrefix("/WEB-INF/views/");
    viewResolver.setSuffix(".jsp");
    return viewResolver;
  }
  @Bean
  public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
    dataSource.setUrl("jdbc:mysql://localhost:3306/mybase?
       serverTimezone=UTC&characterEncoding=UTF-8&useUnicode=yes");
    dataSource.setUsername("root");
    dataSource.setPassword("root");
    return dataSource;
  }
```

La classe `ApplicationConfig` (suite)

```
private Properties getHibernateProperties() {
  Properties properties = new Properties();
  properties.put("hibernate.show_sql", "true");
  properties.put("hibernate.dialect", "org.hibernate.dialect.
    MySQL8Dialect");
  properties.put("hibernate.hbm2ddl.auto", "update");
  return properties;
}

@Bean
public LocalContainerEntityManagerFactoryBean entityManagerFactory()
   {
  HibernateJpaVendorAdapter vendorAdapter = new
    HibernateJpaVendorAdapter();
  LocalContainerEntityManagerFactoryBean factory = new
    LocalContainerEntityManagerFactoryBean();
  factory.setJpaVendorAdapter(vendorAdapter);
  factory.setPackagesToScan("org.eclipse.firstspringmvc.model");
  factory.setDataSource(dataSource());
  factory.setJpaProperties(getHibernateProperties());
  return factory;
}
```

Spring MVC 5

La classe `ApplicationConfig` **(suite)**

```
  @Bean
  public PlatformTransactionManager transactionManager(
     EntityManagerFactory emf) {
    JpaTransactionManager transactionManager = new
       JpaTransactionManager();
    transactionManager.setEntityManagerFactory(emf);
    return transactionManager;
  }

  public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/resources/**")
            .addResourceLocations("/resources/");
  }
}
```

Spring MVC 5

La liste des imports pour la classe `ApplicationConfig`

```
import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.
  EnableJpaRepositories;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.
  LocalContainerEntityManagerFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.annotation.
  EnableTransactionManagement;
import org.springframework.web.servlet.config.annotation.
  ResourceHandlerRegistry;
import org.springframework.web.servlet.view.
  InternalResourceViewResolver;
```

Spring MVC 5

Remarque

Pour que notre nouvelle classe (que nous appellerons `MyWebInitializer`) hérite de la classe `AbstractAnnotationConfigDispatcherServletInitializer`, on peut le préciser au moment de la création en cliquant sur `Browse` du champ `Superclass:`

La classe `MyWebInitializer`

```
package org.eclipse.firstspringmvc.configuration;

import org.springframework.web.servlet.support.
   AbstractAnnotationConfigDispatcherServletInitializer;

public class MyWebInitializer extends
   AbstractAnnotationConfigDispatcherServletInitializer {

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

        @Override
        protected Class<?>[] getServletConfigClasses() {
                return new Class [] { ApplicationConfig.class };
        }

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

Spring MVC 5

Le nouveau contenu du fichier `web.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
     instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/
     javaee http://java.sun.com/xml/ns/javaee/web-
     app_2_5.xsd">
</web-app>
```

N'oublions pas de supprimer le répertoire `spring` situé dans `WEB-INF`

Spring MVC 5

Le nouveau contenu du fichier `web.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
     instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/
     javaee http://java.sun.com/xml/ns/javaee/web-
     app_2_5.xsd">
</web-app>
```

N'oublions pas de supprimer le répertoire `spring` situé dans `WEB-INF`

Relancer le projet

Spring MVC 5

Question

Comment alléger `ApplicationConfig` ?

Spring MVC 5

Question

Comment alléger `ApplicationConfig` ?

La classe `ApplicationConfig` ?

- Mettre tout ce qui concerne les vues et les contrôleurs dans une classe `MvcConfig`
- Annoter cette dernière par `@EnableWebMvc`
- Mettre à jour le fichier `MyWebInitializer`

Spring MVC 5

La classe `MvcConfig`

```
package org.eclipse.firstspringmvc.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.
   ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.
   WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.
   InternalResourceViewResolver;
@Configuration
public class MvcConfig extends WebMvcConfigurerAdapter{
  @Bean
  public InternalResourceViewResolver setup() {
    InternalResourceViewResolver viewResolver = new
       InternalResourceViewResolver();
    viewResolver.setPrefix("/WEB-INF/views/");
    viewResolver.setSuffix(".jsp");
    return viewResolver;
  }
}
```

Depuis Java 8, `WebMvcConfigurerAdapter` est dépréciée et a été remplacée par

```
package org.eclipse.firstspringmvc.configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.
   ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.
   ViewResolverRegistry;
import org.springframework.web.servlet.config.annotation.
   WebMvcConfigurer;

@EnableWebMvc
@Configuration
public class MvcConfig implements WebMvcConfigurer {
  @Override
  public void configureViewResolvers(ViewResolverRegistry registry) {
    registry.jsp("/WEB-INF/views/", ".jsp");
  }
  @Override
  public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/resources/**")
        .addResourceLocations("/resources/");
  }
}
```

La classe `ApplicationConfig` devient ainsi :

```
@Configuration
@ComponentScan(basePackages = "org.eclipse.firstspringmvc.controller")
@EnableJpaRepositories("org.eclipse.firstspringmvc.dao")
@EnableTransactionManagement
public class ApplicationConfig {
  @Bean
  public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
    dataSource.setUrl("jdbc:mysql://localhost:3306/myBase?
       serverTimezone=UTC&characterEncoding=UTF-8&useUnicode=yes");
    dataSource.setUsername("root");
    dataSource.setPassword("root");
    return dataSource;
  }
    private Properties getHibernateProperties() {
    Properties properties = new Properties();
    properties.put("hibernate.show_sql", "true");
    properties.put("hibernate.dialect", "org.hibernate.dialect.
       MySQL8Dialect");
    properties.put("hibernate.hbm2ddl.auto", "create");
    return properties;
  }
```

La classe `ApplicationConfig` (suite)

```
  @Bean
  public LocalContainerEntityManagerFactoryBean entityManagerFactory()
     {
    HibernateJpaVendorAdapter vendorAdapter = new
       HibernateJpaVendorAdapter();
    LocalContainerEntityManagerFactoryBean factory = new
       LocalContainerEntityManagerFactoryBean();
    factory.setJpaVendorAdapter(vendorAdapter);
    factory.setPackagesToScan("org.eclipse.firstspringmvc.model");
    factory.setDataSource(dataSource());
    factory.setJpaProperties(getHibernateProperties());
    return factory;
  }

  @Bean
  public PlatformTransactionManager transactionManager(
     EntityManagerFactory emf){
    JpaTransactionManager transactionManager = new
       JpaTransactionManager();
    transactionManager.setEntityManagerFactory(emf);
    return transactionManager;
  }
}
```

La classe `MyWebInitializer`

```
package org.eclipse.firstspringmvc.configuration;

import org.springframework.web.servlet.support.
   AbstractAnnotationConfigDispatcherServletInitializer;

public class MyWebInitializer extends
   AbstractAnnotationConfigDispatcherServletInitializer {
  @Override
  protected Class<?>[] getRootConfigClasses() {
    return null;
  }

  @Override
  protected Class<?>[] getServletConfigClasses() {
    return new Class [] { ApplicationConfig.class, MvcConfig.class };
  }

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