第 2 章：SpringBoot 整合 Web

1.1 本章大纲

- 整合 Servlet
- 整合 Filter
- 整合 Listener
- 访问静态资源
- 文件上传

1.2 整合 Servlet

1.2.1 方法一：通过注解扫描完成 Servlet 组件注册

1.2.1.1 编写 servlet

```
@WebServlet("/index")
public class IndexServlet extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        // TODO Auto-generated method stub
        System.out.println("执行 doGet()方法");
    }


}
```

1.2.1.2 编写启动类

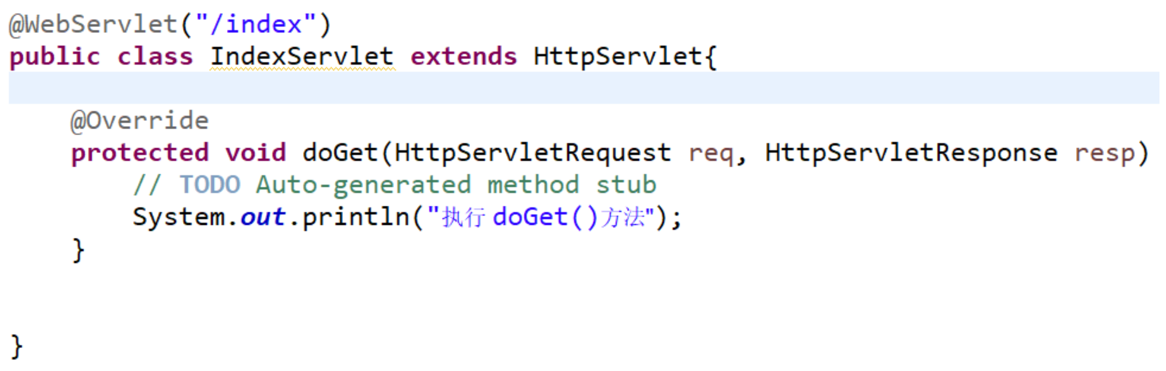
```
@SpringBootApplication
@ServletComponentScan//扫描servlet
public class Springboot02ServletApplication {

    public static void main(String[] args) {
        SpringApplication.run(Springboot02ServletApplication.class, args);
    }

}
```

1.2.2 方法二：通过方法完成 Servlet 组件的注册

1.2.2.1 编写 servlet

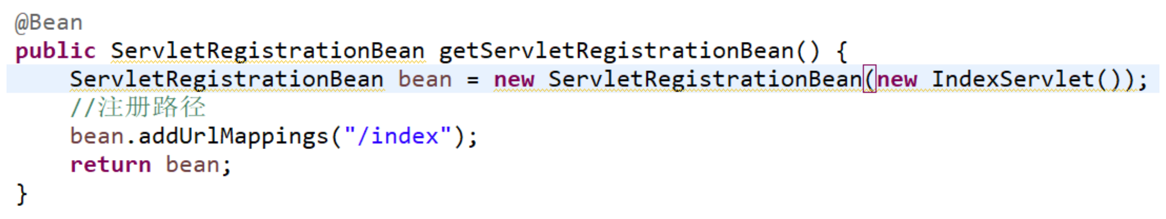


```
@WebServlet("/index")
public class IndexServlet extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        // TODO Auto-generated method stub
        System.out.println("执行 doGet()方法");
    }


}
```

1.2.2.2 编写启动类

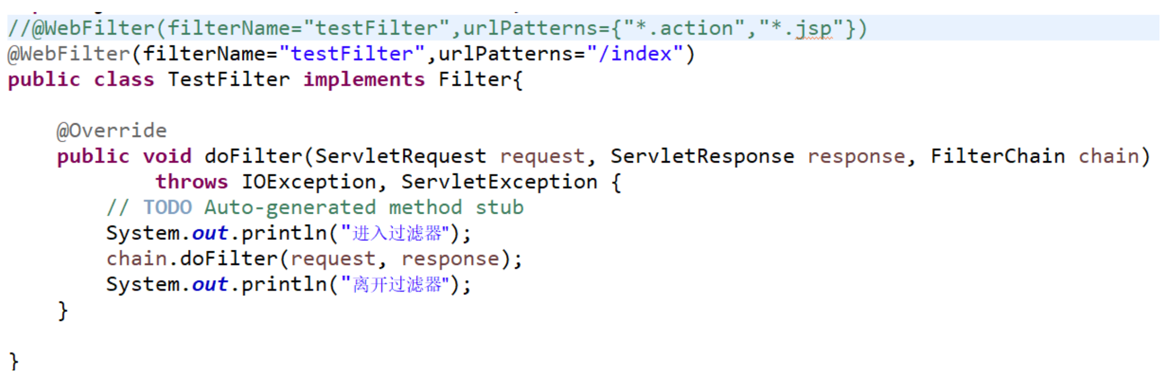


```
@Bean
public ServletRegistrationBean getServletRegistrationBean() {
    ServletRegistrationBean bean = new ServletRegistrationBean(new IndexServlet());
    //注册路径
    bean.addUrlMappings("/index");
    return bean;
}
```

1.3 整合 Filter

1.3.1 方法一：通过注解扫描完成 Filter 组件注册

1.3.1.1 编写 filter



```
//@WebFilter(filterName="testFilter",urlPatterns={"*.action","*.jsp"})
@WebFilter(filterName="testFilter",urlPatterns="/index")
public class TestFilter implements Filter{

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        // TODO Auto-generated method stub
        System.out.println("进入过滤器");
        chain.doFilter(request, response);
        System.out.println("离开过滤器");
    }

}
```

1.3.1.2 编写启动类

```java
@SpringBootApplication
@ServletComponentScan//扫描servlet
public class Springboot02ServletApplication {

    public static void main(String[] args) {
        SpringApplication.run(Springboot02ServletApplication.class, args);
    }

}
```

1.3.2 方法二：通过方法完成 Filter 组件的注册

1.3.2.1 编写 filter

```java
public class TestFilter implements Filter{

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        // TODO Auto-generated method stub
        System.out.println("进入过滤器");
        chain.doFilter(request, response);
        System.out.println("离开过滤器");
    }

}
```

1.3.2.2 编写启动类

```java
@Bean
public FilterRegistrationBean getFilterRegistrationBean() {
    FilterRegistrationBean bean = new FilterRegistrationBean(new TestFilter());
    //过滤过个路径
    //bean.addUrlPatterns(new String[] {"*.do,*.jsp"});
    //过滤单个路径
    bean.addUrlPatterns("/index");
    return bean;
}
```

1.4 整合 Listener

1.4.1 方法一：通过注解扫描完成 Listener 组件注册

1.4.1.1 编写 Listener

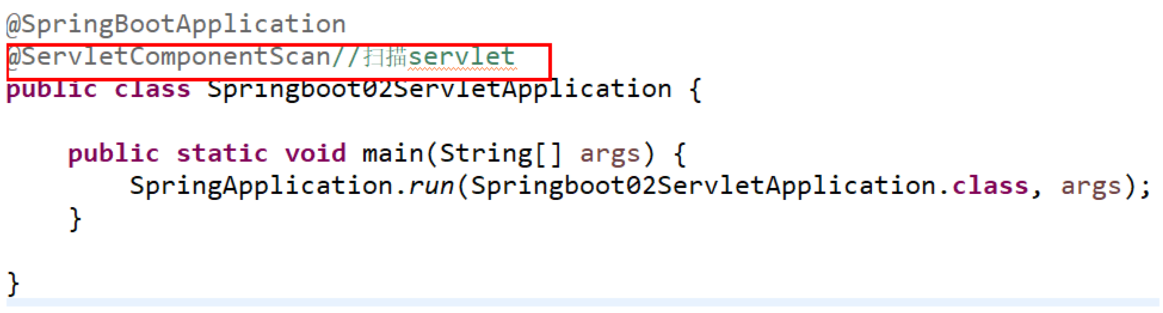
```java
@WebListener
public class TestListener implements ServletContextListener{

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        // TODO Auto-generated method stub
        System.out.println("监听器初始化.....");
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        // TODO Auto-generated method stub
        System.out.println("监听器已销毁.....");
    }


}
```

1.4.1.2 编写启动类



```java
@SpringBootApplication
@ServletComponentScan//扫描servlet
public class Springboot02ServletApplication {

    public static void main(String[] args) {
        SpringApplication.run(Springboot02ServletApplication.class, args);
    }

}
```

1.4.2 方法二：通过方法完成 Listener 组件的注册

1.4.2.1 编写 Listener

```java
public class TestListener implements ServletContextListener{

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        // TODO Auto-generated method stub
        System.out.println("监听器初始化.....");
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        // TODO Auto-generated method stub
        System.out.println("监听器已销毁.....");
    }


}
```

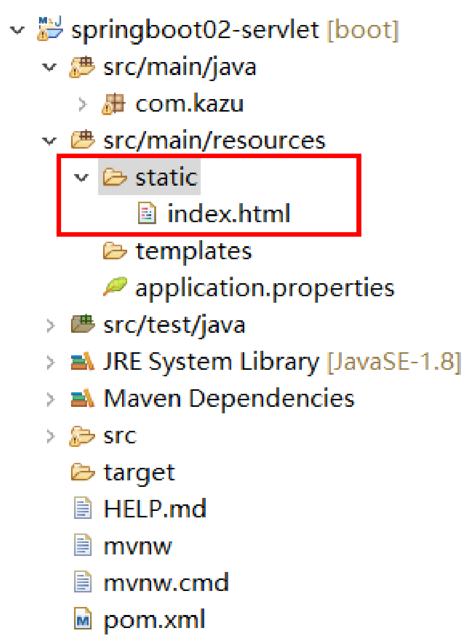
1.4.2.2 编写启动类

```java
@Bean
public ServletListenerRegistrationBean<TestListener> getServletListenerRegistrationBean(){
    ServletListenerRegistrationBean<TestListener> bean = new ServletListenerRegistrationBean<>(new TestListener());
    return bean;
}
```

1.5 访问静态资源

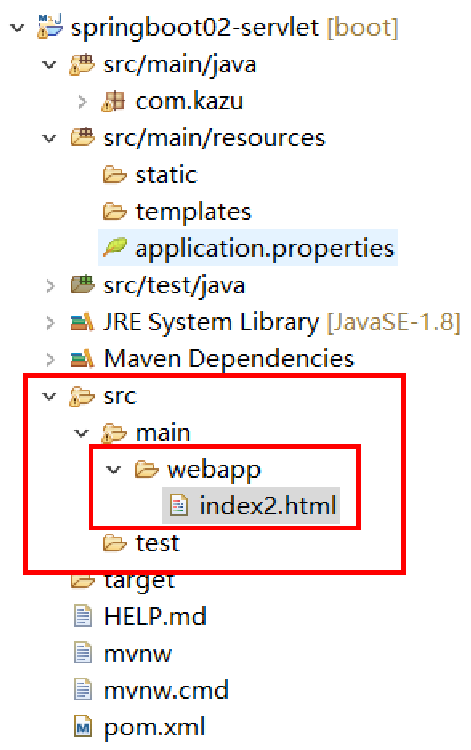
1.5.1 static 目录

注意：静态资源文件夹必须命名为 static


springboot02-servlet [boot]
src/main/java
com.kazu
src/main/resources
static
index.html
templates
application.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml

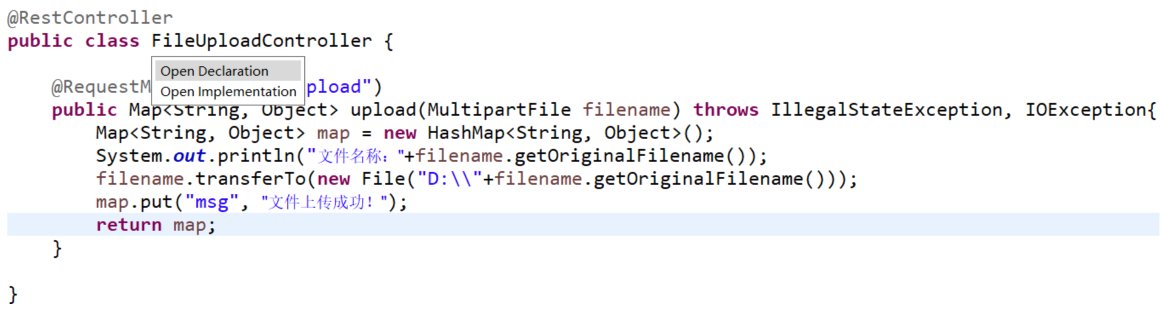
1.5.2 webapp 目录

注意：静态资源文件夹必须命名为 webapp


springboot02-servlet [boot]
src/main/java
com.kazu
src/main/resources
static
templates
application.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
main
webapp
index2.html
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml

1.6 文件上传

1.6.1 编写文件上传 Controller

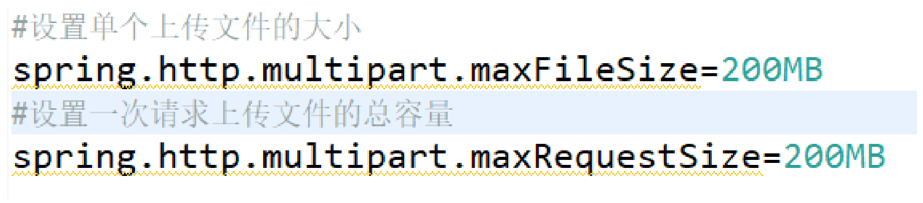


```
@RestController
public class FileUploadController {

    @RequestM[illegible]pload")
    public Map<String, Object> upload(MultipartFile filename) throws IllegalStateException, IOException{
        Map<String, Object> map = new HashMap<String, Object>();
        System.out.println("文件名称: "+filename.getOriginalFilename());
        filename.transferTo(new File("D:\\"+filename.getOriginalFilename()));
        map.put("msg", "文件上传成功! ");
        return map;
    }

}
```

1.6.2 编写 application.properties 文件



```
#设置单个上传文件的大小
spring.http.multipart.maxFileSize=200MB
#设置一次请求上传文件的总容量
spring.http.multipart.maxRequestSize=200MB
```