Spring Boot 第二章

课程介绍:

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

一，整合 Servlet

1，通过注解扫描完成 Servlet 组件的注册

1.1 编写 servlet

```
/**
 *SpringBoot 整合 Servlet 方式一
 *
 *<servlet>
 *  <servlet-name>FirstServlet</servlet-name>
 *  <servlet-class>com.bjsxt.servlet.FirstServlet</servlet-class>
 *</servlet>
 *
 *<servlet-mapping>
 * <servlet-name>FirstServlet</servlet-name>
 * <url-pattern>/first</url-pattern>
 *</servlet-mapping>
 *
 */
```

```java
@WebServlet(name="FirstServlet",urlPatterns="/first")
public class FirstServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // TODO Auto-generated method stub
        super.doGet(req, resp);
    }
}
```

1.2 编写启动类

```java
/**
 * SpringBoot 整合 Servlet 方式一
 *
 *
 */
@SpringBootApplication
@ServletComponentScan //在 springBoot 启动时会扫描@WebServlet，并将该类实例化
public class App {

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

}
```

2，通过方法完成 Servlet 组件的注册

2.1 编写 servlet

```java
/**
 *SpringBoot 整合 Servlet 方式二
 *
 */
```

```
public class SecondServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("SecondServlet.........");
    }

}
```

2.2 编写启动类

```
/**
 * SpringBoot 整合 Servlet 方式二
 *
 *
 */
@SpringBootApplication
public class App2 {

    public static void main(String[] args) {
        SpringApplication.run(App2.class, args);
    }

    @Bean
    public ServletRegistrationBean getServletRegistrationBean(){
        ServletRegistrationBean bean = new ServletRegistrationBean(new
SecondServlet());
        bean.addUrlMappings("/second");
        return bean;
    }
}
```

二，整合 Filter

1，通过注解扫描完成 Filter 组件的注册

1.1 编写 Filter

```
/**
 *SpringBoot 整合 Filter 方式一
 *<filter>
 *  <filter-name>FirstFilter</filter-name>
 *  <filter-class>com.bjsxt.filter.FirstFilter</filter-class>
 *</filter>
 *<filter-mapping>
 *  <filter-name>FirstFilter</filter-name>
 *  <url-pattern>/first</url-pattern>
 *</filter-mapping>
 */
//@WebFilter(filterName="FirstFilter",urlPatterns={"*.do","*.jsp"})
@WebFilter(filterName="FirstFilter",urlPatterns="/first")
public class FirstFilter implements Filter {

    @Override
    public void destroy() {
        // TODO Auto-generated method stub

    }
    @Override
    public void doFilter(ServletRequest arg0, ServletResponse arg1,
FilterChain arg2)
            throws IOException, ServletException {
        System.out.println("进入 Filter");
        arg2.doFilter(arg0, arg1);
        System.out.println("离开 Filter");
    }

    @Override
    public void init(FilterConfig arg0) throws ServletException {
        // TODO Auto-generated method stub
    }
}
```

1.2 编写启动类

```
/**
 *SpringBoot 整合 Filter 方式一
 *
 */
@SpringBootApplication
@ServletComponentScan
public class App {

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

    }

}
```

2，通过方法完成 Filter 组件的注册

2.1 编写 Filter

```
/**
 *
 *SpringBoot 整合 Filter 方式二
 *
 */
public class SecondFilter implements Filter {
    @Override
    public void destroy() {
        // TODO Auto-generated method stub
    }
    @Override
    public void doFilter(ServletRequest arg0, ServletResponse arg1,
FilterChain arg2)
            throws IOException, ServletException {
        System.out.println("进入 SecondFilter");
        arg2.doFilter(arg0, arg1);
        System.out.println("离开 SecondFilter");
    }
```

```
    @Override
    public void init(FilterConfig arg0) throws ServletException {
        // TODO Auto-generated method stub
    }
}
```

2.2 编写启动类

```
/**
 * SpringBoot 整合 Filter 方式二
 *
 *
 */
@SpringBootApplication
public class App2 {

    public static void main(String[] args) {
        SpringApplication.run(App2.class, args);
    }

    /**
     * 注册 Servlet
     * @return
     */
    @Bean
    public ServletRegistrationBean getServletRegistrationBean(){
        ServletRegistrationBean bean = new ServletRegistrationBean(new
SecondServlet());
        bean.addUrlMappings("/second");
        return bean;
    }

    /**
     * 注册 Filter
     */
    @Bean
    public FilterRegistrationBean getFilterRegistrationBean(){
        FilterRegistrationBean bean = new FilterRegistrationBean(new
SecondFilter());
        //bean.addUrlPatterns(new String[]{"*.do","*.jsp"});
        bean.addUrlPatterns("/second");
        return bean;
```

```
    }
}
```

三，整合 Listener

1，通过注解扫描完成 Listener 组件的注册

1.1 编写 Listener

```
/**
 * springBoot 整合 Listener
 *
 *<listener>
 *  <listener-class>com.bjsxt.listener.FirstListener</listener-class>
 *</listener>
 */
@WebListener
public class FirstListener implements ServletContextListener {

    @Override
    public void contextDestroyed(ServletContextEvent arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void contextInitialized(ServletContextEvent arg0) {
        System.out.println("Listener...init......");

    }

}
```

1.2 编写启动类

```
/**
 * springBoot 整合 Listener 方式一
 *
```

```
 *
 */
@SpringBootApplication
@ServletComponentScan
public class App {

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

}
```

2. 通过方法完成 Listener 组件注册

2.1 编写 Listener

```
/**
 * springBoot 整合 Listener 方式二。
 *
 *
 */
public class SecondListener implements ServletContextListener {

    @Override
    public void contextDestroyed(ServletContextEvent arg0) {
        // TODO Auto-generated method stub
    }

    @Override
    public void contextInitialized(ServletContextEvent arg0) {
        System.out.println("SecondListener..init.....");
    }

}
```

2.2 编写启动类

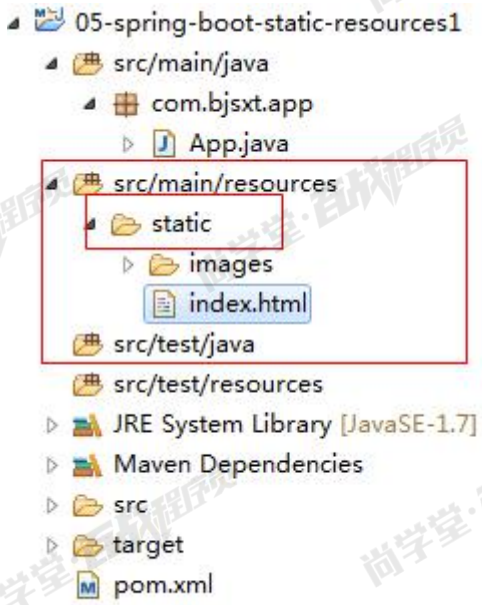
```
/**
 * SpringBoot 整合 Listener 方式二
```

```
 *
 *
 */
@SpringBootApplication
public class App2 {

    public static void main(String[] args) {
        SpringApplication.run(App2.class, args);

    }
    /**
     * 注册 listener
     */
    @Bean
    public ServletListenerRegistrationBean<SecondListener>
getServletListenerRegistrationBean(){
        ServletListenerRegistrationBean<SecondListener> bean= new
ServletListenerRegistrationBean<SecondListener>(new SecondListener());
        return bean;
    }
}
```

四，访问静态资源

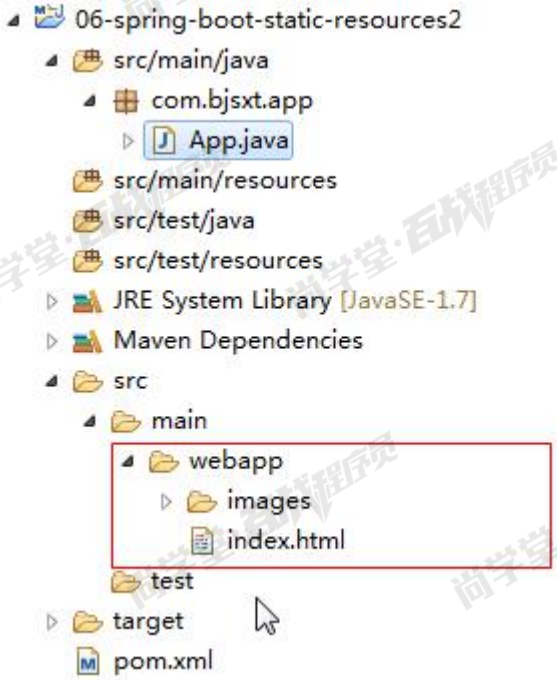
1. SpringBoot 从 classpath/static 的目录

注意目录名称必须是 static



2. ServletContext 根目录下

在 src/main/webapp　目录名称必须要 webapp



五，文件上传

1. 编写 Controller

```
/**
 * SpringBoot 文件上传
 *
 *
 */
//@Controller
@RestController //表示该类下的方法的返回值会自动做 json 格式的转换
public class FileUploadController {

    /*
     * 处理文件上传
     */
    @RequestMapping("/fileUploadController")
    public Map<String, Object> fileUpload(MultipartFile filename)throws
Exception{
        System.out.println(filename.getOriginalFilename());
        filename.transferTo(new
File("e:/"+filename.getOriginalFilename()));
        Map<String, Object> map = new HashMap<>();
        map.put("msg", "ok");
        return map;
    }
}
```

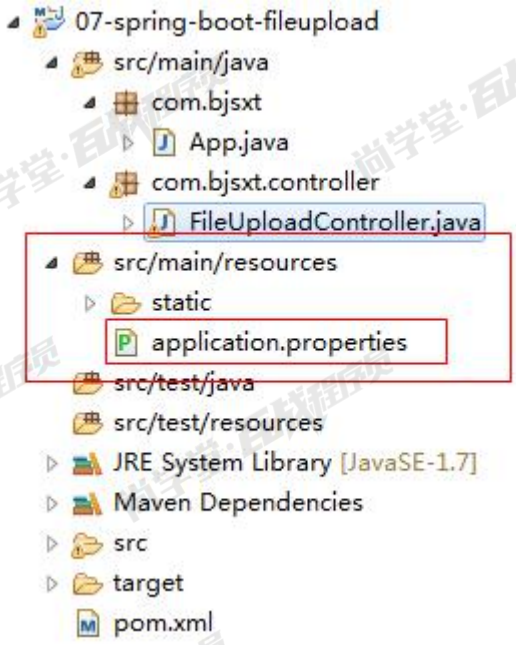
2. 编写启动类

```
/**
 * SpringBoot 文件上传
 *
 *
 */
@SpringBootApplication
public class App {

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

```
    }
}
```

3. 设置上传文件大小的默认值

需要添加一个 springBoot 的配置文件
application.properties



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
```