<http://blog.csdn.net/yxsimple/article/details/46771123>

版权声明：本文为博主原创文章，未经博主允许不得转载。

[**spring**](http://lib.csdn.net/base/javaee) boot集成了servlet容器，当我们在pom文件中增加spring-boot-starter-web的maven依赖时，不做任何web相关的配置便能提供web服务，这还得归于spring boot 自动配置的功能（因为加了EnableAutoConfiguration的注解），帮我们创建了一堆默认的配置，以前在web.xml中配置，现在都可以通过spring bean的方式进行配置，由spring来进行生命周期的管理，大多数情况下，我们需要重载这些配置（例如修改服务的启动端口，contextpath,filter,listener,servlet,session超时时间等）

      1. servlet配置

     当应用只有默认的servlet（即DispatcherServlet）时，映射的url为/,存在其他的servlet时，映射的路径为servlet的注册的beanname(可通过@Component注解修改)，创建方式如下：

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. @Component("myServlet")
2. **public** **class** MyServlet **implements** Servlet{
3. /\*\*
4. \*
5. \* @see javax.servlet.Servlet#destroy()
6. \*/
7. @Override
8. **public** **void** destroy() {
9. System.out.println("destroy...");
10. }
11. /\*\*
12. \* @return
13. \* @see javax.servlet.Servlet#getServletConfig()
14. \*/
15. @Override
16. **public** ServletConfig getServletConfig() {
17. **return** **null**;
18. }
19. /\*\*
20. \* @return
21. \* @see javax.servlet.Servlet#getServletInfo()
22. \*/
23. @Override
24. **public** String getServletInfo() {
25. **return** **null**;
26. }
27. /\*\*
28. \* @param arg0
29. \* @throws ServletException
30. \* @see javax.servlet.Servlet#init(javax.servlet.ServletConfig)
31. \*/
32. @Override
33. **public** **void** init(ServletConfig arg0) **throws** ServletException {
34. System.out.println("servlet init...");
35. }
36. /\*\*
37. \* @param arg0
38. \* @param arg1
39. \* @throws ServletException
40. \* @throws IOException
41. \* @see javax.servlet.Servlet#service(javax.servlet.ServletRequest, javax.servlet.ServletResponse)
42. \*/
43. @Override
44. **public** **void** service(ServletRequest arg0, ServletResponse arg1) **throws** ServletException,
45. IOException {
46. System.out.println("service...");
47. }

    2. filter配置

    filter配置的映射是/\*，创建方式如下：

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. @Component("myFilter")
2. **public** **class** MyFilter **implements** Filter{
3. /\*\*
4. \*
5. \* @see javax.servlet.Filter#destroy()
6. \*/
7. @Override
8. **public** **void** destroy() {
9. System.out.println("destroy...");
10. }
11. /\*\*
12. \* @param arg0
13. \* @param arg1
14. \* @param arg2
15. \* @throws IOException
16. \* @throws ServletException
17. \* @see javax.servlet.Filter#doFilter(javax.servlet.ServletRequest, javax.servlet.ServletResponse, javax.servlet.FilterChain)
18. \*/
19. @Override
20. **public** **void** doFilter(ServletRequest arg0, ServletResponse arg1, FilterChain arg2)
21. **throws** IOException,
22. ServletException {
23. System.out.println("doFilter...");
24. arg2.doFilter(arg0, arg1);
25. }
26. /\*\*
27. \* @param arg0
28. \* @throws ServletException
29. \* @see javax.servlet.Filter#init(javax.servlet.FilterConfig)
30. \*/
31. @Override
32. **public** **void** init(FilterConfig arg0) **throws** ServletException {
33. System.out.println("filter init...");
34. }
36. }<span style="font-family: Arial, Helvetica, FreeSans, sans-serif; font-size: 13.3333330154419px; line-height: 17.3333339691162px; background-color: transparent;">   </span>

    3. listener配置

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. @Component("myListener")
2. **public** **class** MyListener **implements** ServletContextListener{
3. /\*\*
4. \* @param arg0
5. \* @see javax.servlet.ServletContextListener#contextDestroyed(javax.servlet.ServletContextEvent)
6. \*/
7. @Override
8. **public** **void** contextDestroyed(ServletContextEvent arg0) {
9. System.out.println("contextDestroyed...");
10. }
11. /\*\*
12. \* @param arg0
13. \* @see javax.servlet.ServletContextListener#contextInitialized(javax.servlet.ServletContextEvent)
14. \*/
15. @Override
16. **public** **void** contextInitialized(ServletContextEvent arg0) {
17. System.out.println("listener contextInitialized...");
18. }
20. }

如果觉得控制力度不够灵活（例如你想修改filter的映射），spring boot还提供了 ServletRegistrationBean，FilterRegistrationBean，ServletListenerRegistrationBean这3个东西来进行配置

    修改filter的映射

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. @Configuration
2. **public** **class** WebConfig {
3. @Bean
4. **public** FilterRegistrationBean filterRegistrationBean(MyFilter myFilter){
5. FilterRegistrationBean filterRegistrationBean = **new** FilterRegistrationBean();
6. filterRegistrationBean.setFilter(myFilter);
7. filterRegistrationBean.setEnabled(**true**);
8. filterRegistrationBean.addUrlPatterns("/bb");
9. **return** filterRegistrationBean;
10. }
11. }

   4. 配置servlet 容器

    可以通过两种方式配置servlet容器，一种是通过properties文件，例如：

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. server.port=8081
2. server.address=127.0.0.1
3. server.sessionTimeout=30
4. server.contextPath=/springboot

完整的配置信息可以看这 <http://github.com/spring-projects/spring-boot/tree/master/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/ServerProperties.java>

    另一种方式是**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)**代码的形式：

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. @Component
2. **public** **class** MyCustomizationBean **implements** EmbeddedServletContainerCustomizer  {
3. /\*\*
4. \* @param container
5. \* @see org.springframework.boot.context.embedded.EmbeddedServletContainerCustomizer#customize(org.springframework.boot.context.embedded.ConfigurableEmbeddedServletContainer)
6. \*/
7. @Override
8. **public** **void** customize(ConfigurableEmbeddedServletContainer container) {
9. container.setContextPath("/sprintboot");
10. container.setPort(8081);
11. container.setSessionTimeout(30);
12. }
14. }

   5. error 处理

   spring boot 提供了/error映射来进行错误处理，它会返回一个json来对错误信息进行描述（包含了http状态和异常信息）,例如404的错误



     当然可以定制错误页面，通过实现ErrorController接口，并将它装配起来即可，如下：

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. @Controller
2. **public** **class** ErrorHandleController **implements** ErrorController {
3. /\*\*
4. \* @return
5. \* @see org.springframework.boot.autoconfigure.web.ErrorController#getErrorPath()
6. \*/
7. @Override
8. **public** String getErrorPath() {
9. **return** "/screen/error";
10. }
12. @RequestMapping
13. **public** String errorHandle(){
14. **return** getErrorPath();
15. }
16. }

    还可以这样：

**[java]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. @Component
2. **private** **class** MyCustomizer **implements** EmbeddedServletContainerCustomizer {
4. @Override
5. **public** **void** customize(ConfigurableEmbeddedServletContainer container) {
6. container.addErrorPages(**new** ErrorPage(HttpStatus.BAD\_REQUEST, "/screen/error"));
7. }
9. }

   6.模板引擎

    spring boot 会自动配置 FreeMarker,Thymeleaf,Velocity，只需要在pom中加入相应的依赖即可，例如应用Velocity

**[html]** [view plain](http://blog.csdn.net/yxsimple/article/details/46771123) [copy](http://blog.csdn.net/yxsimple/article/details/46771123)

1. **<dependency>**
2. **<groupId>**org.springframework.boot**</groupId>**
3. **<artifactId>**spring-boot-starter-velocity**</artifactId>**
4. **</dependency>**

    默认配置下spring boot会从src/main/resources/templates目录中去找模板

  7. jsp限制

    如果要在spring boot中使用jsp,得将应用打包成war,这里有配置的example <https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples/spring-boot-sample-web-jsp>