[spring boot 知识点整理](http://blog.csdn.net/luqiang81191293/article/details/54629873)

2017-01-20 11:08 2528人阅读 [评论](http://blog.csdn.net/luqiang81191293/article/details/54629873#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/luqiang81191293/article/details/54629873#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

spring boot（8） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，转载请注明出处。

最近新的项目架构启用spring boot cloud，SO现在先坐下简单的技术梳理，后边的博客会把spring的技术细节，boot的技术细节重新梳理一遍

**1、下面是根据条件初始化bean**

****

**2、读取配置信息操作**

加载配置可以用@PropertySource("classpath:com/ecej/test2/test.properties") 记得要用 private Environment environment; 读取配置

**[java]** [view plain](http://blog.csdn.net/luqiang81191293/article/details/54629873) [copy](http://blog.csdn.net/luqiang81191293/article/details/54629873)

1. **package** com.ecej.test2;
3. **import** org.apache.commons.io.IOUtils;
5. **import** org.springframework.beans.factory.annotation.Autowired;
7. **import** org.springframework.beans.factory.annotation.Value;
9. **import** org.springframework.context.annotation.Bean;
11. **import** org.springframework.context.annotation.ComponentScan;
13. **import** org.springframework.context.annotation.Configuration;
15. **import** org.springframework.context.annotation.PropertySource;
17. **import** org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
19. **import** org.springframework.core.env.Environment;
21. **import** org.springframework.core.io.Resource;
23. @Configuration
25. @ComponentScan("com.ecej.test2")
27. @PropertySource("classpath:com/ecej/test2/test.properties")
29. **public** **class** ELConfig {
31. @Value("I LOVE YOU")
33. **private** String normal;
35. @Value("#{systemProperties['os.name']}")
37. **private** String osName;
39. @Value("#{T(java.lang.Math).random() \* 100.0}")
41. **private** **double** randomNumber;
43. @Value("#{demoService.another}")
45. **private** String fromAnother;
47. @Value("#classpath:com/ecej/test2/test.properties")
49. **private** Resource testFile;
51. @Value("http://www.baidu.com")
53. **private** Resource testUrl;
55. @Value("${book.name}")
57. **private** String bookName;
59. @Autowired
61. **private** Environment environment;
63. @Bean
65. **public** **static** PropertySourcesPlaceholderConfigurer propertyConfigure() {
67. **return** **new** PropertySourcesPlaceholderConfigurer();
69. }
71. **public** **void** outputResource() {
73. **try** {
75. System.out.println(normal);
77. System.out.println(osName);
79. System.out.println(randomNumber);
81. System.out.println(fromAnother);
83. System.out.println(IOUtils.toString(testUrl.getInputStream()));
85. System.out.println(bookName);
87. System.out.println(environment.getProperty("book.author"));
89. } **catch** (Exception e) {
91. // TODO: handle exception
93. }
95. }
97. }

**3、spring event**

事件

1 、首先我们要实现ApplicationListener 实现我们自己的监听

2、 定义我们自己的事件 通过集成ApplicationEvent实现

3、 定义config启动

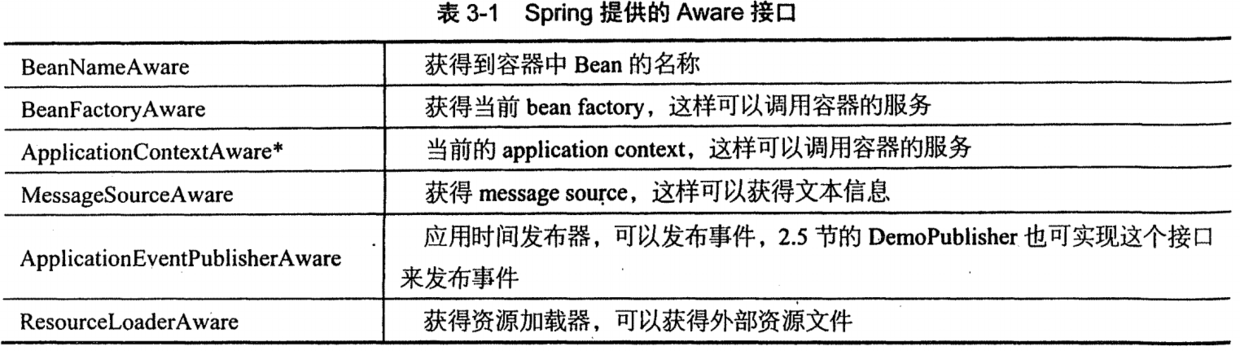
通过applicationContext 发布事件

**[java]** [view plain](http://blog.csdn.net/luqiang81191293/article/details/54629873) [copy](http://blog.csdn.net/luqiang81191293/article/details/54629873)

1. @Component
3. **public** **class** DemoPublisher {
5. @Autowired
7. **private** ApplicationContext applicationContext;
9. **public** **void** publish(String msg) {
11. applicationContext.publishEvent(**new** DemoEvent(**this**, msg));
13. }
15. }

**4、spring Aware**

讲解：bean 和spring是无耦合的，但是如果想用到spring容器的功能资源，就要你的bean知道spring的存在，这就是spring aware



**5、多线程**

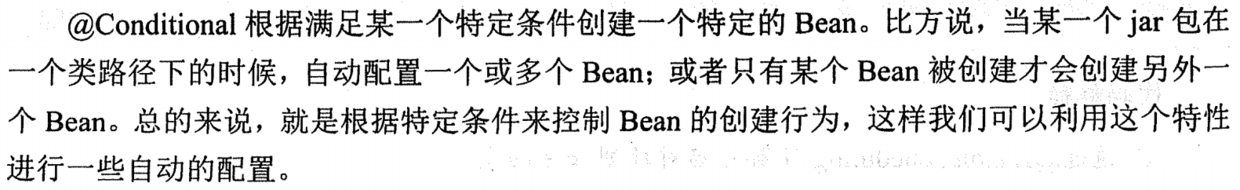
spring通过 TaskExecutor来实现多线程并发编程。使用ThreadPoolExecutor可实现基于线程池的TaskExecutor,使用@EnableAsync开启对异步任务的支持，并通过在实际执行bean方法中使用@Async注解来声明一个异步任务

**6、计划任务**

通过配置注解@EnableScheduline来开启对计划任务的支持，然后再要执行的任务上加注解@Scheduled

spring通过@Scheduled支持多种类型计划任务，包含cron,fixDelay、fixRate(固定时间执行)

**7、条件注解@Conditional**

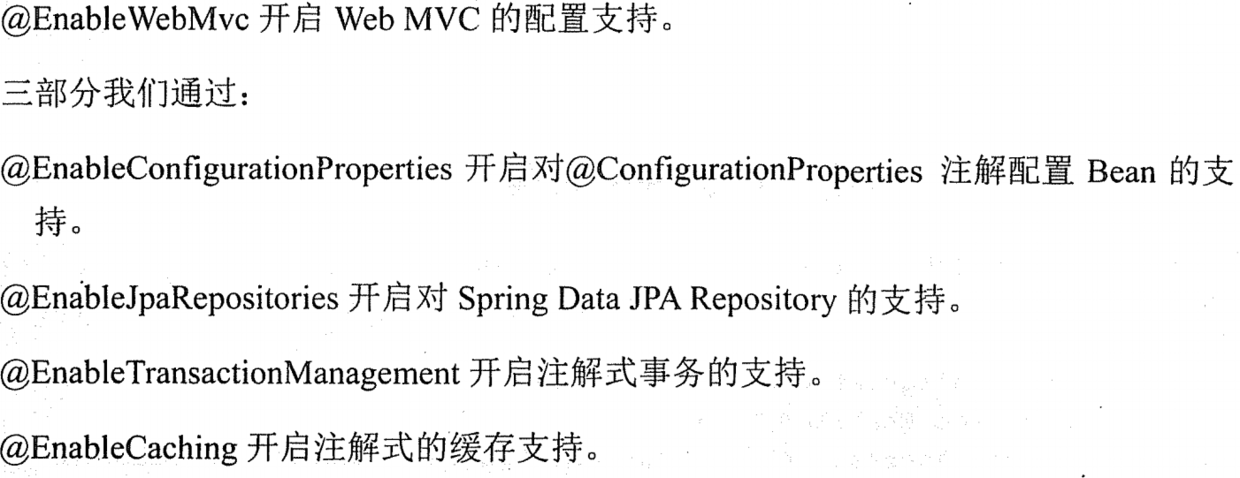


**8、组合注解与源注解**

源注解：就是可以注解到别的注解上的注解，被注解的注解称之为组合注解。组合注解有源注解的功能。

@Configuration  @ComponentScan 组合出 Wiselyconfiguration

**9、@Enable\*注解工作原理**



**10、Spring MVC**

下面所讲解的都是配置在MyMVCConfig

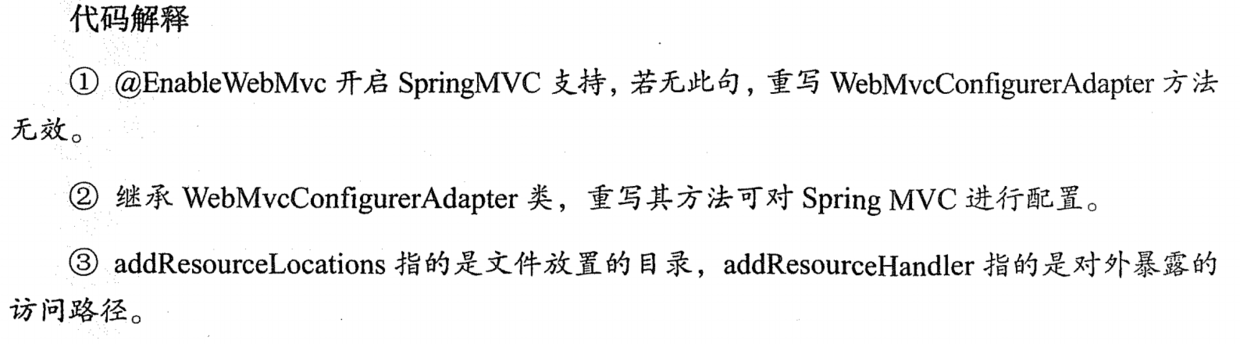
通过实现WebApplicationInitializer 等同于web.xml配置

基本配置

spring MVC的定制配置需要我们配置集成一个WebMvcConfigurerAdapter,并在此使用@EnableWebMvc注解，来开启对spring MVC配置支持

**静态资源**

重写addResourceHandlers方法实现



**拦截器配置**

**类似servlet的Filter**

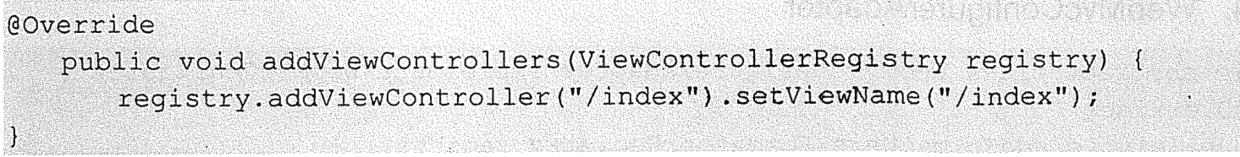
可以让普通bean 实现HanlderIntercepetor接口或者继承HandlerInterceptorAdapter类来实现自定义拦截器

在boot中通过重写WebMvcConfigurerAdapter 的 addInterceptors方法来注册自定义拦截器

**@ControllerAdvice**

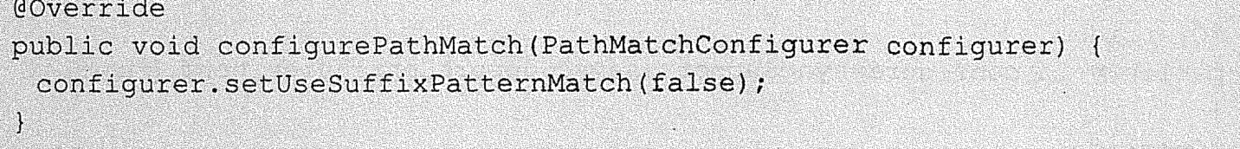
@ExceptionHandler定义全局处理 ，通过value属性可以设置拦截过滤条件

在开发中经常会遇到跳转页面的事情，我们还要单独写一个方法很麻烦，现在可以这样



**路径参数配置**

在spring mvc中路径参数如果带点“.” ,那点后面的值将被忽略。通过重写configurePathMatch(PathMatchConfigurer) 可不忽略 点后参数

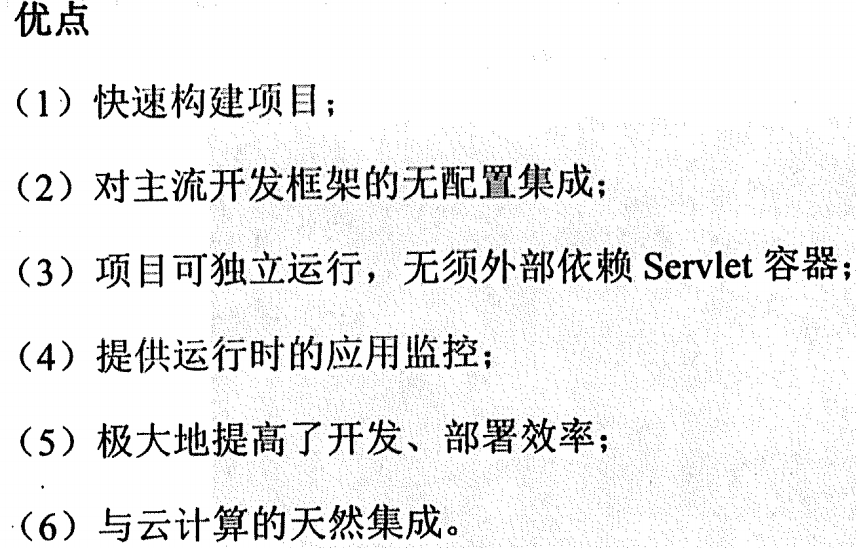


**文件上传**

demo集合



**二 、正式开始spring boot**



CLI  命令行控制工具

**1、@SpringBootApplication 和入口**

这个标签是个组合注解，包含了@Configuration @EnableAutoConfiguration @ComponentScan三个标签

@EnableAutoConfiguration 让spring boot根据类路径中的jar包依赖为当前项目进行自动配置

**在spring boot中我们可以使用**

@Value("${book.author}")直接注入属性，但是还是感觉一个个注入麻烦啊，SO，我们可以直接映射一个类，用@ConfigurationProperties(prefix="author",locations={"classpath:author.properties"})通过prefix指定前缀，通过locations指定位置

**2、spring boot 的web开发**

需要定义模版信息的话，使用ViewResolver  ,别忘了在config上加注解@EnableWebMvc

**[java]** [view plain](http://blog.csdn.net/luqiang81191293/article/details/54629873) [copy](http://blog.csdn.net/luqiang81191293/article/details/54629873)

1. @Bean
3. **public** InternalResourceViewResolver viewResolver() {
5. InternalResourceViewResolver resolver = **new** InternalResourceViewResolver();
7. resolver.setPrefix("WEB-INF/classes/views/");
9. resolver.setSuffix(".jsp");
11. resolver.setViewClass(JstlView.**class**);
13. **return** resolver;
15. }

静态资源默认放在src/main.resources/static 下面

**3、静态首页的支持**

**4、接管spring boot 的web配置**

如果boot 提供的配置不是我们需要的，可以通过配置类修改，

注解来实现自己完全控制

@Configuration

@ComponentScan("com.ecej.test.mvc")

@EnableWebMvc

如果我们想即用默认配置，又增加自定义配置，可以集成WebMvcConfigurerAdapter，无需使用@EnableWebMvc注解

**5、注册servlet Filter Listener(重点来了)**

可以注册ServletRegistrationBean  FilterRegistrationBean  ServletListenerRegistrationBean 的bean来实现

**6、Tomcat配置**

其实就是servlet容器配置，因为BOOT内置的是tomcat，所以也就叫tomcat配置了

配置都在[**org**](http://blog.csdn.net/luqiang81191293/article/details/54629873)**.**[**springframework**](http://blog.csdn.net/luqiang81191293/article/details/54629873)**.**[**boot**](http://blog.csdn.net/luqiang81191293/article/details/54629873)**.**[**autoconfigure**](http://blog.csdn.net/luqiang81191293/article/details/54629873)**.**[**web**](http://blog.csdn.net/luqiang81191293/article/details/54629873)**.ServerProperties 中（其实大部分都有这么个配置）**

SO，我们只需要在application.properties中配置就好了（如果你想拥别的名字，只需要配置下就行咯，在上边有提到过）。通用的配置都以server作为前缀

例子：

配置容器

server.port=8080

server.session-timeout=2

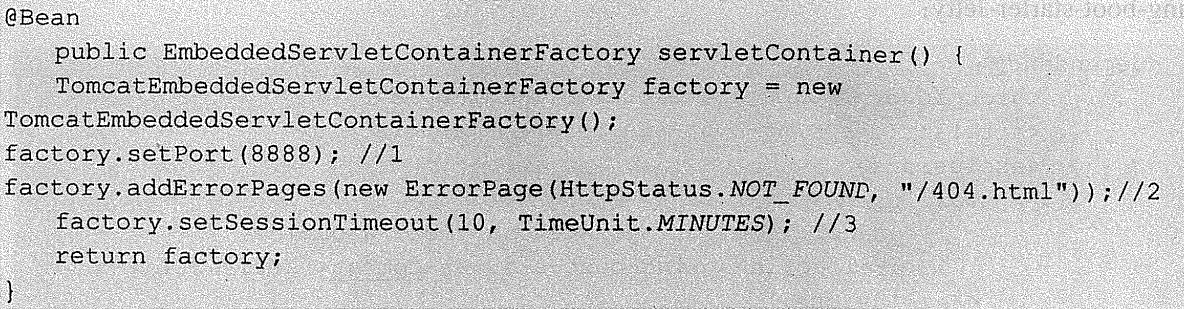
配置tomcat

server.tomcat.uri-encoding=UTF-8

上边都是配置文件配置，如果想玩玩代码也是可以的，下面介绍代码配置

想配置servlet容器可以实现一个EmbeddedServletContainerCustomizer的接口（注意声明的类要为static）

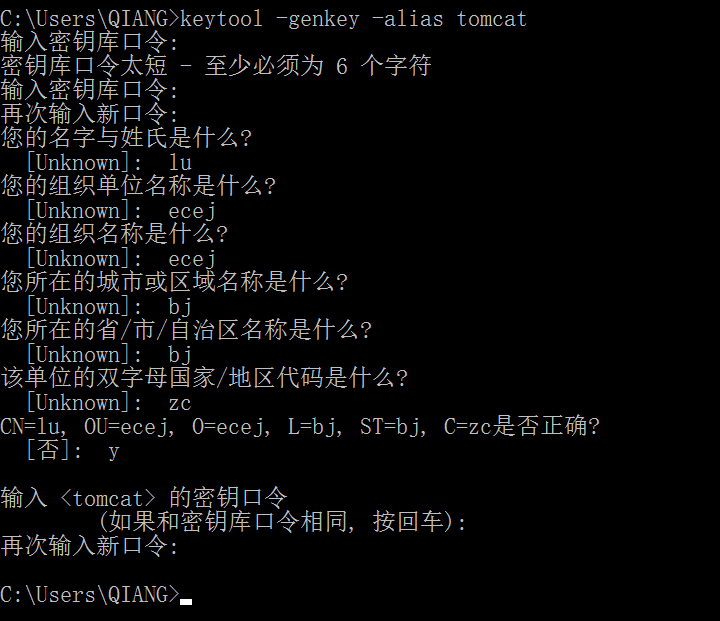
想直接配置tomcat等则可以直接定义TomcatEmbeddedServletContainerFactory等



**7、SSL配置（安全套接层）**

生成证书：

**JDK下有工具keytool ，他是一个证书管理工具，可以用来生成证书**



在你命令执行的当前目录下生成了一个.keystore的证书

配置我们的配置文件

server.port=8000

server.ssl.key-store=.keystore

server.ssl.key-store-password=123456

server.ssl.keyStoreType=JKS

server.ssl.keyAlias:tomcat

此时在启动项目就变成了https的

**8、设置自己的Favicon**

这个就非常简单了，只需要将自己的favicon.ico文件放在META-INF/resources/   resources/  static/  public/下面任意一个目录下就行了。

**9、WebSocket**

这到底是个什么鬼呢？

官方说法，就是为浏览器和服务端提供双工异步通信的功能。直接使用WebSocket会使开发非常繁琐的，所以我们使用它的子协议STOMP，它是一个更高级的协议，STOMP协议使用一个基于帧的格式来定义消息，与HTTP的request response类似。

spring boot内置了这玩意，可以看websocket包下的类

需要加入

spring-boot-starter-websocket 包

@EnableWebSocketMessageBroker注解并继承AbstractWebSocketMessageBrokerConfigurer

模式：

广播式  会将消息发送给所有连接了当前的浏览器

代码片段

**[java]** [view plain](http://blog.csdn.net/luqiang81191293/article/details/54629873) [copy](http://blog.csdn.net/luqiang81191293/article/details/54629873)

1. **package** com.ecej.demo2.websocket;
3. **import** org.springframework.context.annotation.ComponentScan;
5. **import** org.springframework.context.annotation.Configuration;
7. **import** org.springframework.messaging.simp.config.MessageBrokerRegistry;
9. **import** org.springframework.web.socket.config.annotation.AbstractWebSocketMessageBrokerConfigurer;
11. **import** org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
13. **import** org.springframework.web.socket.config.annotation.StompEndpointRegistry;
15. @Configuration
17. @ComponentScan(value = "com.ecej.demo2.websocket")
19. @EnableWebSocketMessageBroker
21. **public** **class** WebSocketConfig **extends** AbstractWebSocketMessageBrokerConfigurer {
23. @Override
25. **public** **void** registerStompEndpoints(StompEndpointRegistry register) {
27. register.addEndpoint("/endpointWisely").withSockJS();
29. }
31. @Override
33. **public** **void** configureMessageBroker(MessageBrokerRegistry register) {
35. register.enableSimpleBroker("/topic");
37. }
39. }

解析：

1、通过@EnableWebSocketMessageBroker 注解开启使用STOMP协议传输基于代理（message broker）的消息，

这时控制器支持使用@MessageMapping，就像使用@RequestMapping一样

2、注册STOMP协议的节点，并映射的指定的URL

3、注册一个STOMP的endpoint，并指定使用SocketJS协议

4、配置消息代理（message broker）

5、广播式应配置一个/topic消息代理

**10、spring 的事物机制**

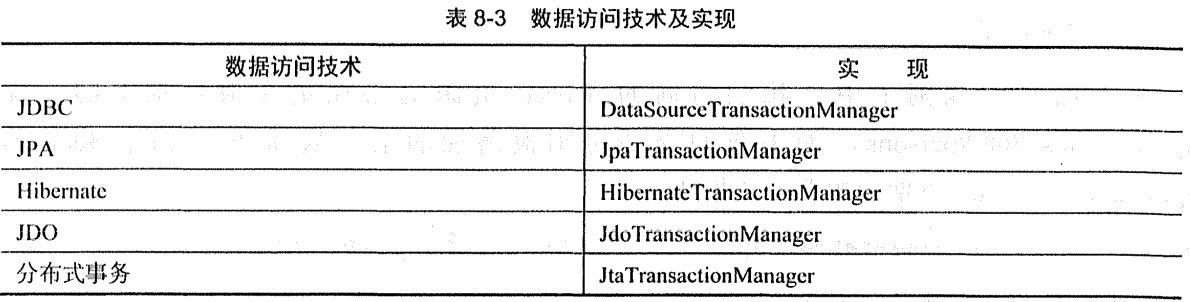
spring事物机制提供了一个PlatformTransactionManager接口，不同的数据访问技术的事物使用不同的接口实现

声明式事物

使用@Transactional注解在方法上表明该方法需要事物支持，基于AOP实现操作。

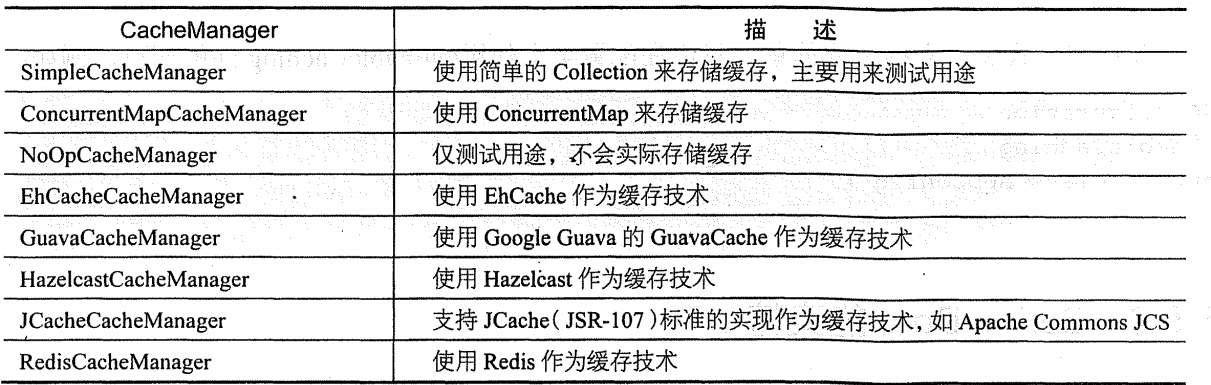
使用@EnableTransactionManagement来开启上边的注解

类级别的注解@Transactional会重载方法级别的



**11、缓存支持**

不同的缓存技术，需要不同的cacheManager



**12、异步消息**

**spring 对JMS和AMQP的支持分别来自于spring-jms 和spring-rabbit**

他们分布需要ConnectionFactory来实现连接消息代理，并分别提供了JmsTemplate、RabbitTemplate

spring为JMS 、AMQP提供了@JmsListener @RabbitListener 注解在方法上监听消息代理发布的消息。我们只需要分别通过@EnableJms @EnableRabbit开启支持