spring注解

springboot注解

@RestController和@RequestMapping注解

我们的Example类上使用的第一个注解是 @RestController 。这被称为一个构造型（stereotype）注解。它为阅读代码的人们提供建议。对于Spring，该类扮演了一个特殊角色。在本示例中，我们的类是一个web @Controller ，所以当处理进来的web请求时，Spring会询问它。@RequestMapping 注解提供路由信息。它告诉Spring任何来自"/"路径的HTTP请求都应该被映射到 home 方法。 @RestController 注解告诉Spring以字符串的形式渲染结果，并直接返回给调用者。

注： @RestController 和 @RequestMapping 注解是Spring MVC注解（它们不是Spring Boot的特定部分）

@EnableAutoConfiguration注解

第二个类级别的注解是 @EnableAutoConfiguration 。这个注解告诉Spring Boot根据添加的jar依赖猜测你想如何配置Spring。由于 spring-boot-starter-web 添加了Tomcat和Spring MVC，所以auto-configuration将假定你正在开发一个web应用并相应地对Spring进行设置。Starter POMs和Auto-Configuration：设计auto-configuration的目的是更好的使用"Starter POMs"，但这两个概念没有直接的联系。你可以自由地挑选starter POMs以外的jar依赖，并且Spring Boot将仍旧尽最大努力去自动配置你的应用。

你可以通过将 @EnableAutoConfiguration 或 @SpringBootApplication 注解添加到一个 @Configuration 类上来选择自动配置。

注：你只需要添加一个 @EnableAutoConfiguration 注解。我们建议你将它添加到主 @Configuration 类上。

如果发现应用了你不想要的特定自动配置类，你可以使用 @EnableAutoConfiguration 注解的排除属性来禁用它们。

```
import org.springframework.boot.autoconfigure.*;
import org.springframework.boot.autoconfigure.jdbc.*;
import org.springframework.context.annotation.*;
@Configuration
@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})
public class MyConfiguration {
}
```

@Configuration

Spring Boot提倡基于Java的配置。尽管你可以使用一个XML源来调用 SpringApplication.run() ，我们通常建议你使用 @Configuration 类作为主要源。一般定义 main 方法的类也是主要 @Configuration 的一个很好候选。你不需要将所有的 @Configuration 放进一个单独的类。 @Import 注解可以用来导入其他配置类。另外，你也可以使用 @ComponentScan 注解自动收集所有的Spring组件，包括 @Configuration 类。

如果你绝对需要使用基于XML的配置，我们建议你仍旧从一个 @Configuration 类开始。你可以使用附加的 @ImportResource 注解加载XML配置文件。

@Configuration注解该类，等价 与XML中配置beans；用@Bean标注方法等价于XML中配置bean

@ComponentScan

你可以自由地使用任何标准的Spring框架技术去定义beans和它们注入的依赖。简单起见，我们经常使用 @ComponentScan 注解搜索beans，并结合 @Autowired 构造器注入。

如果使用上面建议的结构组织代码（将应用类放到根包下），你可以添加 @ComponentScan 注解而不需要任何参数。你的所有应用程序组件（ @Component , @Service , @Repository , @Controller 等）将被自动注册为

Spring Beans。

@SpringBootApplication

很多Spring Boot开发者总是使用 @Configuration ， @EnableAutoConfiguration 和 @ComponentScan 注解他们的main类。由于这些注解被如此频繁地一块使用（特别是你遵循以上最佳实践时），Spring Boot提供一个方便的@SpringBootApplication 选择。

该 @SpringBootApplication 注解等价于以默认属性使用 @Configuration ， @EnableAutoConfiguration 和 @ComponentScan 。

```
package com.example.myproject;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication // same as @Configuration @EnableAutoConfiguration @ComponentScan
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

@ConfigurationProperties

属性注入

```
@Component
@ConfigurationProperties(prefix="connection")
public class ConnectionSettings {
    private String username;
    private InetAddress remoteAddress;
    // ... getters and setters
}
```

为了使用@ConfigurationProperties beans，你可以使用与其他任何bean相同的方式注入它们

```
@Service
public class MyService {
    @Autowired
    private ConnectionSettings connection;
    //...
    @PostConstruct
    public void openConnection() {
        Server server = new Server();
        this.connection.configure(server);
    }
}
```

正如使用@ConfigurationProperties注解一个类，你也可以在@Bean方法上使用它。当你需要绑定属性到不受你控制的第三方组件时，这种方式非常有用。

为了从Environment属性配置一个bean，将@ConfigurationProperties添加到它的bean注册过程：

```
@ConfigurationProperties(prefix = "foo")
@Bean
public FooComponent fooComponent() {
...
```

```
}
```

和上面ConnectionSettings的示例方式相同，任何以foo为前缀的属性定义都会被映射到FooComponent上。

Spring Boot将尝试校验外部的配置，默认使用JSR-303（如果在classpath路径中）。你可以轻松的为你的@ConfigurationProperties类添加JSR-303 javax.validation约束注解：

```
@Component
@ConfigurationProperties(prefix="connection")
public class ConnectionSettings {
@NotNull
private InetAddress remoteAddress;
// ... getters and setters
}
```

@EnableConfigurationProperties

当@EnableConfigurationProperties注解应用到你的@Configuration时，任何被@ConfigurationProperties注解的beans将自动被Environment属性配置

你可以通过在@EnableConfigurationProperties注解中直接简单的列出属性类来快捷的注册@ConfigurationProperties bean的定义。

```
@Configuration
@EnableConfigurationProperties(ConnectionSettings.class)
public class MyConfiguration {
}
```

@Component和@Bean

@Component被用在要被自动扫描和装配的类上。@Component类中使用方法或字段时不会使用CGLIB增强(及不使用代理类：调用任何方法，使用任何变量，拿到的是原始对象)Spring 注解@Component等效于@Service,@Controller,@Repository

@Bean主要被用在方法上，来显式声明要用生成的类;用@Configuration注解该类，等价 与XML中配置beans；用@Bean标注方法等价于XML中配置bean。

现在项目上，本工程中的类，一般都使用`@Component`来生成bean。在把通过web service取得的类，生成Bean时，使用`@Bean`和getter方法来生成bean

---

@Profiles

Spring Profiles提供了一种隔离应用程序配置的方式，并让这些配置只能在特定的环境下生效。任何@Component或@Configuration都能被@Profile标记，从而限制加载它的时机。

```
@Configuration
@Profile("production")
public class ProductionConfiguration {
// ...
}
```

以正常的Spring方式，你可以使用一个spring.profiles.active的Environment属性来指定哪个配置生效。你可以使用平常的任何方式来指定该属性，例如，可以将它包含到你的application.properties中：

```
spring.profiles.active=dev,hsqldb
```