spring学习笔记--spring概览

|  |  |
| --- | --- |
|  | 🢂 内容概览 |
|  | Why：此文档用来做什么？它存在的意义是什么？为解决什么问题？   |  | | --- | |  |   What：当前包含了那些内容？   |  | | --- | |  |   How：此文档应如何参考？   |  | | --- | |  |   Who：此文档适用于那些人员阅读参考？   |  | | --- | |  | |

目录

[1 Spring概览 3](#_Toc450838049)

[1.1 什么是spring？ 3](#_Toc450838050)

[1.2 依赖注入&控制反转 3](#_Toc450838051)

[1.3 Spring模块介绍 3](#_Toc450838052)

[1.3.1 核心容器（Core Container） 4](#_Toc450838053)

[1.3.2 AOP和Instrumentation 4](#_Toc450838054)

[1.3.3 消息 4](#_Toc450838055)

[1.3.4 数据接入 4](#_Toc450838056)

[1.3.5 Web 5](#_Toc450838057)

[1.3.6 Test 5](#_Toc450838058)

[1.4 使用场景 5](#_Toc450838059)

[1.5 版本包管理 7](#_Toc450838060)

[1.5.1 使用maven管理 7](#_Toc450838061)

[1.5.1.1 添加spring maven仓库 8](#_Toc450838062)

[1.5.1.2 确保spring版本的一致 8](#_Toc450838063)

[1.5.2 手工管理 8](#_Toc450838064)

[1.6 日志 8](#_Toc450838065)

[1.6.1 排除对commons-logging的依赖 8](#_Toc450838066)

[1.6.2 使用log4j 9](#_Toc450838067)

[1.6.3 使用slf4j 9](#_Toc450838068)

[1.6.3.1 使用log4j 9](#_Toc450838069)

[1.6.3.2 使用logback 10](#_Toc450838070)

[2 spring4新特性 10](#_Toc450838071)

# Spring概览

## 什么是spring？

1. 轻量的、一站式的准企业应用解决方案;
2. 通过模块化构建，允许开发人员仅使用部分模块，而不必把spring全部引入到项目中；
3. 低侵入式设计，业务代码通常不需要强依赖于框架代码；

## 依赖注入&控制反转

java开发中通常有一系列松散的约定，将应用分为多层，每层的对象之间会存在各种依赖。各个对象之间的依赖关系如何组织和构建，通常是由架构师或者开发人员来完成的，期间可以引入注入模式等最佳实践。然而，这些最佳实践仅仅作为一种范式来使用，并没有作为一个代码框架用来强制约束开发人员使用，开发人员往往需要自己去实现。

Spring的IOC框架，目的就是以规范化的方式将不同的组件集成到一个应用当中，将设计模式作为第一公民，避免开发者重复造轮子，减小开发难度。

## Spring模块介绍

Spring按照功能划分大约20个模块，这些模块被归类到核心容器、数据接入、Web、AOP、Instrumentation、消息、测试等；

|  |
| --- |
| spring功能架构 |

### 核心容器（Core Container）

1. spring核心容器由spring-core、spring-beans、spring-context、spring-context-support、spring-expression组成；
2. spring-core和spring-beans提供了框架的基本功能，包括IOC和依赖注入的特性；BeanFactory是一个工厂设计模式的实现；避免了以编程的方式实现单例，并且将配置和具体的程序逻辑解耦；
3. spring-context在spring-core和spring-beans的基础之上，提供了对象的访问框架，对象访问方式类似于JNDI注册；spring-context继承了spring-beans的功能，并添加了国际化、时间传播、资源加载、透明的context创建；spring-context模块提供了对J2EE的支持，比如：EJB、JMX以及基本的远程访问功能等；ApplicationContext是spring-context模块的核心接口；
4. spring-context-support提供了集成第三方组件的功能，包括：缓存（ehcache、guava、jcache）、邮件（javamail）、任务调度（common,quartz）、模板引擎（freemarker、jasperreports、velocity）；
5. spring-expression模块提供了表达式语言（spel），能够在运行时查询和操作的对象图，如：访问或者设置对象属性、方法调用、访问数组、集合、定义变量、进行算数运算、支持列表投影、选择和聚合等；spel是（unified El，jsp2.1）的扩展实现；

### AOP和Instrumentation

1. Spring-aop模块提供了面向切面的编程能力；
2. Spring-aspects模块提供了集成AspectJ的能力；
3. Spring-instrument提供了监控和类加载器的功能；

### 消息

1. spring-message模块能够用于基于消息的应用，提供了一系列注解，类似于spring-mvc的基于注解的编程模型；

### 数据接入

数据接入层由JDBC，ORM，OXM，JMS以及事务模型组成；

1. spring-jdbc模块提供了JDBC抽象层，避免用户编写冗余代码，处理具体错误码；
2. spring-tx模块同时支持编程式事务和声明式事务；
3. spring-orm模块提供了ORM映射的API，能够支持JPA、JDO、Hibernate等；
4. spring-oxm模块提供了对象与xml映射的支持，包括：JAXB,Castor,XMLBeans,JiBx,XStream等；
5. spring-jms模块包括了生产/消费消息的能力；在spring4.1中和spring-messaging模块集成；

### Web

1. Web层由spring-web，spring-webmvc，spring-websocket，spring-webmvc-portlet组成；
2. Spring-web模块提供了基本的面向web的特性，比如：多文件上传，通过Servlet监听初始化IOC容器，以及针对web实现的context对象等；也包括HTTP客户端以及针对Web的远程服务等；
3. Spring-webmvc模块提供了基于spring的mvc框架，以及基于REST协议的webservice框架；
4. Spring-webmvc-portlet模块提供portlet的环境支持；

### Test

Spring-test模块提供了单元测试和集成测试的组件，能够使用junit或者testNg进行测试，并提供了mock对象，能够在相对隔离的环境中测试代码；

## spring使用场景

### 提供完整web解决方案的使用场景；

|  |
| --- |
|  |

### 作为中间件的使用场景；

|  |
| --- |
|  |

### 远程调用的使用场景；

|  |
| --- |
|  |

### 对于EJB的兼容；

|  |
| --- |
|  |

## spring版本包管理

### 使用maven管理

#### 添加spring maven仓库

|  |
| --- |
| **<repositories>**  **<repository>**  **<id>**io.spring.repo.maven.release**</id> <url>**http://repo.spring.io/release/**</url> <snapshots><enabled>**false**</enabled></snapshots>**  **</repository>**  **</repositories>** |

#### 确保spring版本的一致

在使用spring时，有可能存在不同版本的jar包，导致错误。可以通过spring-framework-bom组件来确保所有spring组件都保持相同版本；需要在maven中添加如下配置，添加后，将不用分别为每个组件添加version标签；

|  |
| --- |
| **<dependencyManagement>**  **<dependencies>**  **<dependency> <groupId>**org.springframework**</groupId> <artifactId>**spring-framework-bom**</artifactId> <version>**4.2.3.RELEASE**</version> <type>**pom**</type> <scope>**import**</scope>**  **</dependency>**  **</dependencies>**  **</dependencyManagement>** |

### 手工管理

下载spring版本包，添加至类路径：http://repo.spring.io/release/org/springframework/ spring

## 日志配置

spring日志依赖于JCL（Jakarta Commons Logging）的api，故需要使用一种实现了JCL的日志组件，默认使用commons-logging作为日志工具；commons-logging能够在运行时自动发现系统中引入的日志框架，避免了用户手工操作，但也因此降低了效率；如果想要更换日志组件，则需要两步操作：

1. 排除spring对commons-logging的依赖；
2. 引入实现了JCL的日志组件进行替换；通常使用更简洁、高效的slf4j（Simple Logging Façade for Java）来替换；

### 排除对commons-logging的依赖

|  |
| --- |
| **<dependencies>**  **<dependency>**  **<groupId>**org.springframework**</groupId> <artifactId>**spring-core**</artifactId> <version>**4.2.3.RELEASE**</version> <exclusions>**  **<exclusion> <groupId>**commons-logging**</groupId> <artifactId>**commons-logging**</artifactId>**  **</exclusion>**  **</exclusions>**  **</dependency>**  **</dependencies>** |

### 使用log4j

log4j使用，只需要将其jar包添加到类路径中，同时提供log4j的配置文件即可；maven配置如下：

|  |
| --- |
| **<dependencies>**  **<dependency>**  **<groupId>**org.springframework**</groupId> <artifactId>**spring-core**</artifactId>**  **<version>**4.2.3.RELEASE**</version>**  **</dependency>**  **<dependency>**  **<groupId>**log4j**</groupId> <artifactId>**log4j**</artifactId> <version>**1.2.14**</version>**  **</dependency>**  **</dependencies>** |

### 使用slf4j

slf4j相比commons-logging更加简洁高效，因为它使用了编译时静态绑定，而非在运行时自动发现日志框架；slf4j支持多种日志框架的扩展，并且具有桥接日志工具的功能；所以在spring中使用slf4j，需要使用slf4j-jcl组件，将spring的内部调用转换为slf4j的api调用；

#### 使用log4j

通常的做法是使用slf4j的api，然后绑定到log4j等日志组件，由log4j等日志组件作为具体的实现；maven配置如下：

|  |
| --- |
| **<dependencies>**  **<dependency>**  **<groupId>**org.springframework**</groupId> <artifactId>**spring-core**</artifactId> <version>**4.2.3.RELEASE**</version> <exclusions>**  **<exclusion> <groupId>**commons-logging**</groupId> <artifactId>**commons-logging**</artifactId>**  **</exclusion>**  **</exclusions>**  **</dependency>**  **<dependency>**  **<groupId>**org.slf4j**</groupId> <artifactId>**jcl-over-slf4j**</artifactId> <version>**1.5.8**</version>**  **</dependency>**  **<dependency>**  **<groupId>**org.slf4j**</groupId> <artifactId>**slf4j-api**</artifactId> <version>**1.5.8**</version>**  **</dependency>**  **<dependency>**  **<groupId>**org.slf4j**</groupId> <artifactId>**slf4j-log4j12**</artifactId> <version>**1.5.8**</version>**  **</dependency>**  **<dependency>**  **<groupId>**log4j**</groupId> <artifactId>**log4j**</artifactId> <version>**1.2.14**</version>**  **</dependency>**  **</dependencies>** |

#### 使用logback

更加通用的做法是使用logback，因为logback直接实现了slf4j-api，不需要做中转，也不需要引入slf4j-api；

|  |
| --- |
| **<dependencies>**  **<dependency>**  **<groupId>**org.springframework**</groupId> <artifactId>**spring-core**</artifactId> <version>**4.2.3.RELEASE**</version> <exclusions>**  **<exclusion> <groupId>**commons-logging**</groupId> <artifactId>**commons-logging**</artifactId>**  **</exclusion>**  **</exclusions>**  **</dependency>**  **<dependency>**  **<groupId>**org.slf4j**</groupId> <artifactId>**jcl-over-slf4j**</artifactId> <version>**1.5.8**</version>**  **</dependency>**  **<dependency>**  **<groupId>**org.slf4j**</groupId> <artifactId>**slf4j-api**</artifactId> <version>**1.5.8**</version>**  **</dependency>**  **<dependency>**  **<groupId>**org.slf4j**</groupId> <artifactId>**slf4j-log4j12**</artifactId> <version>**1.5.8**</version>**  **</dependency>**  **<dependency>**  **<groupId>**log4j**</groupId> <artifactId>**log4j**</artifactId> <version>**1.2.14**</version>**  **</dependency>**  **</dependencies>** |

# spring4新特性

spring4主要提供了对java8特性的支持；

1. 示例：https://spring.io/guides
2. spring3和spring4的api差别：http://docs.spring.io/spring-framework/docs/3.2.4.RELEASE\_to\_4.0.0.RELEASE/