# Spring笔记

## 1.Spring Framework

用于构建企业一站式开发解决方案

## 2.SpringMVC和SpringBoot区别

Spring 框架就像一个家族，有众多衍生产品例如 boot、security、jpa等等。但他们的基础都是Spring 的 ioc和 aop ioc 提供了依赖注入的容器 aop ，解决了面向横切面的编程，然后在此两者的基础上实现了其他延伸产品的高级功能。Spring MVC是基于 Servlet 的一个 MVC 框架 主要解决 WEB 开发的问题，因为 Spring 的配置非常复杂，各种XML、 JavaConfig、hin处理起来比较繁琐。于是为了简化开发者的使用，从而创造性地推出了Spring boot，约定优于配置，简化了spring的配置流程。

说得更简便一些：Spring 最初利用“工厂模式”（DI）和“代理模式”（AOP）解耦应用组件。大家觉得挺好用，于是按照这种模式搞了一个 MVC框架（一些用Spring 解耦的组件），用开发 web 应用（ SpringMVC ）。然后有发现每次开发都写很多样板代码，为了简化工作流程，于是开发出了一些“懒人整合包”（starter），这套就是 Spring Boot。

## 3.SpringMVC功能

Spring MVC提供了一种轻度耦合的方式来开发web应用。

Spring MVC是Spring的一个模块，式一个web框架。通过Dispatcher Servlet, ModelAndView 和 View Resolver，开发web应用变得很容易。解决的问题领域是网站应用程序或者服务开发——URL路由、Session、模板引擎、静态Web资源等等。

## 4.SpringBoot功能

Spring Boot实现了自动配置，降低了项目搭建的复杂度。

众所周知Spring框架需要进行大量的配置，Spring Boot引入自动配置的概念，让项目设置变得很容易。Spring Boot本身并不提供Spring框架的核心特性以及扩展功能，只是用于快速、敏捷地开发新一代基于Spring框架的应用程序。也就是说，它并不是用来替代Spring的解决方案，而是和Spring框架紧密结合用于提升Spring开发者体验的工具。同时它集成了大量常用的第三方库配置(例如Jackson, JDBC, Mongo, Redis, Mail等等)，Spring Boot应用中这些第三方库几乎可以零配置的开箱即用(out-of-the-box)，大部分的Spring Boot应用都只需要非常少量的配置代码，开发者能够更加专注于业务逻辑。

Spring Boot只是承载者，辅助你简化项目搭建过程的。如果承载的是WEB项目，使用Spring MVC作为MVC框架，那么工作流程和你上面描述的是完全一样的，因为这部分工作是Spring MVC做的而不是Spring Boot。

对使用者来说，换用Spring Boot以后，项目初始化方法变了，配置文件变了，另外就是不需要单独安装Tomcat这类容器服务器了，maven打出jar包直接跑起来就是个网站，但你最核心的业务逻辑实现与业务流程实现没有任何变化。

所以，用最简练的语言概括就是：

Spring 是一个“引擎”；

Spring MVC 是基于Spring的一个 MVC 框架 ；

Spring Boot 是基于Spring4的条件注册的一套快速开发整合包。

## 5. SpringBoot jdbc

## 6. SpringBoot H2

## 7. Spring事务

|  |  |
| --- | --- |
| 数据访问技术 | 实现类 |
| JDBC | DataSourceTransactionManager |
| JPA | JpaTransactionManager |
| Hibernate | HibernateTransactionManager |
| JDO | JdoTransactionManager |
| 分布式事务 | JtaTransactionManager |

## 8. Spring 事务传播特性

|  |  |  |
| --- | --- | --- |
| 1 | PROPAGATION\_REQUIRED | 如果存在一个事务，则支持当前事务。如果没有事务则开启 |
| 2 | PROPAGATION\_SUPPORTS | 如果存在一个事务，支持当前事务。如果没有事务，则非事务的执行 |
| 3 | PROPAGATION\_MANDATORY | 如果已经存在一个事务，支持当前事务。如果没有一个活动的事务，则抛出异常。 |
| 4 | PROPAGATION\_REQUIRES\_NEW | 总是开启一个新的事务。如果一个事务已经存在，则将这个存在的事务挂起。 |
| 5 | PROPAGATION\_NOT\_SUPPORTED | 总是非事务地执行，并挂起任何存在的事务。 |
| 6 | PROPAGATION\_NEVER | 总是非事务地执行，如果存在一个活动事务，则抛出异常 |
| 7 | PROPAGATION\_NESTED | 如果一个活动的事务存在，则运行在一个嵌套的事务中. 如果没有活动事务,  则按TransactionDefinition.PROPAGATION\_REQUIRED 属性执行 |

## 9.Spring 事务隔离级别

|  |  |  |
| --- | --- | --- |
| 1 | ISOLATION\_DEFAULT | 这是一个PlatfromTransactionManager默认的隔离级别，使用数据库默认的事务隔离级别. 另外四个与JDBC的隔离级别相对应 |
| 2 | ISOLATION\_READ\_UNCOMMITTED | 这是事务最低的隔离级别，它充许令外一个事务可以看到这个事务未提交的数据。这种隔离级别会产生脏读，不可重复读和幻像读。 |
| 3 | ISOLATION\_READ\_COMMITTED | 保证一个事务修改的数据提交后才能被另外一个事务读取。另外一个事务不能读取该事务未提交的数据 |
| 4 | ISOLATION\_REPEATABLE\_READ | 这种事务隔离级别可以防止脏读，不可重复读。但是可能出现幻像读。它除了保证一个事务不能读取另一个事务未提交的数据外，还保证了避免下面的情况产生(不可重复读)。 |
| 5 | ISOLATION\_SERIALIZABLE | 这是花费最高代价但是最可靠的事务隔离级别。事务被处理为顺序执行。 了防止脏读，不可重复读外，还避免了幻像读。 |

## 10.Spring事务基于注解的配置

[1.@EnableTransactionManagement](mailto:1.@EnableTransactionManagement) 基于注解的配置

2.<tx:annotation-driven>

## 11.Spring jdbc 异常抽象（自定义异常返回码）

## 12.Spring常用注解

1. Configuration

标明java类是一个配置类

2. ImportResource

注入配置文件以为的xml配置文件

3.ComponentScan

配置扫描page下的bean

4.Bean

通过bean将配置的方法配置到spring bean 容器中

5.ConfigurationProperties

获取spring配置

6.Qualifier

配置通过名字来进行注入

## 13.SpringBoot Actuator 常用的功能

|  |  |
| --- | --- |
| /actuator/health | 健康检查 |
| /actuator/beans | 查看容器中所有的beans |
| /actuator/mapping | 查看web的url映射 |
| /actuator/env | 查看幻境信息 |

## 14.多数据库、分库分表、读写分离的关系

## 15.事务的本质

Spring事务实际上是通过aop来增强了类的功能

SpringAOP本质就是为类做一个代理类，看似在调用自己写的类实际增强后的的代理类

16.事务-REQUIRES\_NEW & NESTED

REQUIRES\_NEW ：始终启动一个新的事务，两个事务没有关联

NESTED：在原事务内新建一个内嵌事务，两个事务之间有关联， 外部事务回滚内嵌事务也会回滚

16.Druid 慢日志