**2019/10/19**

**Spring 概述**

1. **Spring** 是一个**容器**，通过**反转控制(** IoC )和**依赖注入**( DI )来实现**高内聚、低耦合**的应用。除此之外它可以**整合很多第三方框架**，它还**提供面向切面编程**（ AOP ）的能力，**对数据库事务的管理尤其方便。**

Spring 是一个**开源**的**轻量级** Java SE（ Java 标准版本）/Java EE（ Java 企业版本）**应用开发框架**，其目的是用于**简化企业级应用程序开发**。在传统应用程序开发中，一个完整的应用是由一组**相互协作的对象**组成的。所以开发一个应用除了要开发业务逻辑之外，最多的是关注使这些对象协作来完成所需功能的同时，实现低耦合、高内聚。所以，**业务逻辑开发**是不可避免的。**如果有个框架可以帮我们来创建对象及管理这些对象之间的依赖关系，能通过配置方式来创建对象，管理对象之间依赖关系，我们不需要通过工厂和生成器来创建及管理对象之间的依赖关系，这样我们必然会减少许多工作量，加快开发**。Spring 框架问世之初主要就是来完成这个功能。

Spring 框架除了帮我们**管理对象及其依赖关系**，还提供像通用**日志记录、性能统计、安全控制、异常处理等面向切面的能力**，可以帮我们管理最头疼的**数据库事务**，同时，它本身提供了一套简单的 **JDBC**访问实现，能与第三方数据库访问框架集成（如 Hibernate、JPA ），与各种 Java EE 技术整合（如 Java Mail、任务调度等等），提供一套自己的 web 层框架 Spring MVC 、而且还能非常简单的与第三方 web 框架集成。从这里我们可以认为 **Spring 是一个超级粘合平台**，除了自己提供功能外，还提供粘合其他技术和框架的能力，从而使我们可以更自由的选择到底使用什么技术进行开发。而且不管是 JAVA SE（ C/S 架构）应用程序还是 JAVA EE（ B/S 架构）应用程序都可以使用这个平台进行开发

**注：**

1. Hibernate: **开源**的**对象关系映射框架**，它对**JDBC进行了非常轻量级的对象封装**，它将**POJO(简单java对象)**与**数据库表建立映射关系**， 是一个全自动的orm框架，**hibernate可以自动生成SQL语句**，自动执行，使得Java程序员可以随心所欲的**使用对象编程思维来操纵数据 库**。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，

JSP全名为Java Server Pages，中文名叫java服务器页面，其根本是一个简化的[Servlet](https://baike.so.com/doc/2354665-2490021.html)设计，它 是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。

2. **JDBC**: (java数据库连接)是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接 口组成。)

3. ORM(Object Relational Mapping)框架采用[**元数据**](https://baike.so.com/doc/169527-179094.html)来**描述对象一关系映射细节**，元数据一般采用XML格式，并且存放在专门的对象一映 射文件中。

4. JPA是Java Persistence API的简称，中文名**Java持久层API**，是JDK 5.0注解或XML描述对象-关系表的映射关系，并将运行期的实体**[对 象持久化](https://baike.so.com/doc/7883735-8157830.html" \t "_blank)**到数据库中。

**2.2.1. Spring 能帮我们根据配置文件创建及组装对象之间的依赖关系**

**传统程序开发**，创建对象及组装对象间依赖关系由我们在**程序内部进行控制**，这样会加大各个对象间的耦合，如果我们**要修改对象间的依赖关系就必须修改源代码，重新编译、部署**；而如果采用 Spring，则由 **Spring 根据配置文件来进行创建及组装对象间依赖关系，只需要改配置文件即可，无需重新编译。**

**2.2.2 Spring 面向切面编程能帮助我们无耦合的实现日志记录、性能统计、安全控制**

当我们要进行一些**日志记录、权限控制、性能统计等工作时**，在**传统应用程序**当中我们可能在**需要的对象或方法中进行**，而且比如权限控制、性能统计**大部分是重复的**，这样代码中就存在大量重复代码，即使把通用部分提取出来，仍旧存在重复调用，像**性能统计**我们可能只是在**必要时才进行，在诊断完毕后要删除**这些代码；还有日志记录，比如记录一些方法访问日志、数据访问日志等等，这些都会渗透到各个访问方法中；还有**权限控制，必须在方法执行时开始进行审核**，可见这些工作急需优化。如果**采用 Spring** ，这些日志记录、权限控制、性能统计从**业务逻辑中分离出来**，通过 Spring 支持的**面向切面编程，动态添加这些功能**，**无需渗透到各个需要的方法或对象中**。反观其他的解决方案，通过使用**“代理设计模式”**或**“包装器设计模式”**，仍需通过编程方式来创建代理对象，仍需耦合这些代理对象，而采用 Spring 面向切面编程能提供一种更好的方式来完成上述功能。通过配置方式，而不需要在现有代码中添加任何额外代码，使现有代码专注业务逻辑。

**2.2.3 Spring 能非常简单的帮我们管理数据库事务**

在传统应用程序当中，我们如何来完成数据库事务管理？需要一系列**“获取连接，执行 SQL，提交或回滚事务，关闭连接”，**而且还要保证在最后一定要关闭连接，多么可怕的事情，而且也很无聊；如果采用 Spring，我们只需获取连接，执行 SQL，其他的都交给 Spring 来管理，简单方便。

**除以上之外，Spring 还可以：**

与**第三方数据库访问框架（如 Hibernate、JPA ）无缝集成，而且自己也提供了一套 JDBC 访问模板，来方便数据库访问。**

与第三方 Web（如 Struts、JSF ）框架无缝集成，而且自己也提供了一套 Spring MVC 框架，来方便 web 层搭建。

**注：****Struts2以WebWork优秀的设计思想为核心，吸收了 Struts框架的部分优点，提供了一个更加整洁的MVC设计模式实现的Web 应用程序框架,**

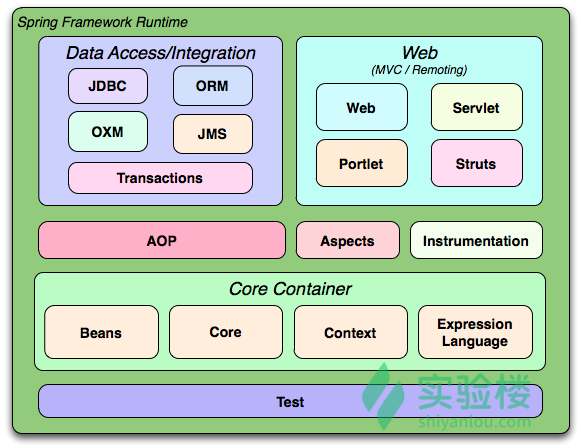
**JSF 是一种用于构建Java Web 应用程序的标准框架(是Java Community Process 规定的JSR-127标准)。它提供了一种以组件为中心的**[**用户**](https://baike.so.com/doc/5926095-6139018.html)**界面(UI)构建方法，从而简化了Java服务器端应用程序的开发。**

与 Java EE（如 Java Mail 、任务调度）整合，与更多技术整合（比如缓存框架）。

**概念总结：**

* **轻量级与重量级：**轻量级是相对于重量级而言的，轻量级一般就是**非入侵性的、所依赖的东西非常少、资源占用非常少、部署简单**等等，其实就是**比较容易使用**，而重量级正好相反。
* **框架：**能完成一定功能的**半成品**，比如我们可以使用框架进行购物网站开发；框架做一部分功能，我们自己做一部分功能，辅助高效工作。而且**框架规定了你在开发应用程序时的整体架构，提供了一些基础功能**，**还规定了类和对象的如何创建、如何协作**等，从而简化我们的代码编写，让我们专注于业务逻辑开发。
* **容器：**在日常生活中容器就是一种盛放东西的器具，从程序设计角度看就是**装对象的的对象**，因为存在**放入、拿出等操作**，所以容器还要**管理对象的生命周期。**
* **控制反转：**即 Inversion of Control ，缩写为 IoC ，控制反转还有一个名字叫做依赖注入（ Dependency Injection ），就是**由容器控制程序之间的关系**，而非传统实现中，由程序代码直接操控。
* **AOP：** AOP 是 Aspect Oriented Programming 的缩写，意思是面向切面编程。从另一个角度来考虑程序结构以完善面向对象编程（ OOP ），**即可以通过在编译期间、装载期间或运行期间实现在不修改源代码的情况下给程序动态添加功能的一种技术**。通俗点说就是**把可重用的功能提取**出来，然后将这些通用功能在合适的时候织入到应用程序中；比如安全，日志记录，这些都是通用的功能，我们可以把它们提取出来，然后在程序执行的合适地方植入这些代码并执行它们，从而完成**需要的功能并复用了这些功能**。

#### 2.5.1 核心容器：包括 Core 、 Beans 、 Context 、 EL 模块

* **Core 模块：**封装了**框架依赖的最底层部分**，包括资源访问、类型转换及一些常用工具类。
* **Beans 模块：**提供了**框架的基础部分**，包括控制反转（ IOC ）和依赖注入（ DI ）。其中 **BeanFactory 是容器核心**，本质是“工厂设计模式”的实现，而且无需编程实现“单例设计模式”，单例完全由容器控制，而且提倡面向接口编程，而非面向实现编程；所有应用程序对象及对象间关系由框架管理，从而真正从程序逻辑中把维护对象之间的依赖关系提取出来，**所有这些依赖关系都由 BeanFactory 来维护。**
* **Context 模块**：**以 Core 和 Beans 为基础**，集成 Beans 模块功能并添加**资源绑定、数据验证、国际化、 Java EE 支持、容器生命周期、事件传播**等；核心接口**ApplicationContext** 。
* **EL 模块：**提供强大的**表达式语言**支持，支持访问和修改属性值，方法调用，支持访问及修改数组、容器和索引器，命名变量，支持算数和逻辑运算，支持从 Spring 容器获取 Bean， 它也支持列表投影、选择和一般的列表聚合等。

**2.5.1 AOP 、 Aspects 模块**

* **AOP 模块：** Spring AOP 模块提供了**符合 AOP Alliance 规范**的面向切面的编程（ aspect-oriented programming ）实现，提供比如日志记录、权限控制、性能统计等**通用功能和业务逻辑分离的技术**，并且能**动态**的把这些功能添加到需要的代码中；这样各专其职，降低业务逻辑和通用功能的耦合。
* **Aspects 模块：**提供了对 AspectJ 的集成，AspectJ 提供了比 Spring ASP 更强大的功能。

**注：AspectJ是一个面向切面的框架，它扩展了Java语言。AspectJ定义了AOP语法，所以它有一个专门的**[**编译器**](https://baike.so.com/doc/6133192-6346352.html)**用来生成遵守Java字节编码规范的Class文件。**

**2.5.2数据访问/集成模块：该模块包括了 JDBC 、 ORM 、 OXM 、 JMS 和事务管理。**

* **事务模块：**该模块用于 **Spring 管理事务**，只要是 Spring 管理对象都能得到 Spring 管理事务的好处，无需在代码中进行事务控制了，而且支持编程和声明性的事务管理。
* **JDBC 模块：**提供了一个 JDBC 的样例模板，使用这些模板能消除传统冗长的 JDBC 编码还有必须的事务控制，而且能享受到 Spring 管理事务的好处。
* **ORM 模块：**提供与流行的**“对象-关系”映射框架的无缝集成**，包括 **Hibernate 、JPA 、 MyBatis** 等。而且可以使用 Spring 事务管理，无需额外控制事务。
* **OXM 模块：**提供了一个对 **Object / XML 映射实现**，将 java 对象映射成 XML 数据，或者将 XML 数据映射成 java 对象， Object / XML 映射实现包括 JAXB 、 Castor 、 XMLBeans 和 XStream 。
* **JMS 模块：**用于 JMS ( Java Messaging Service )，提供一套 “消息生产者、消息消费者”模板用于更加简单的使用 JMS ， **JMS 用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。**

**2.5.3 Web / Remoting 模块：包含了 Web 、 Web-Servlet 、 Web-Struts 、 Web-Porlet 模块。**

* **Web 模块：**提供了基础的 web 功能。例如多文件上传、集成 IoC 容器、远程过程访问（ RMI 、Hessian 、 Burlap ）以及 Web Service 支持，并提供一个 RestTemplate 类来提供方便的 Restful services 访问。
* **Web-Servlet 模块：**提供了一个 Spring MVC Web 框架实现。Spring MVC 框架提供了基于**注解的请求资源注入、更简单的数据绑定、数据验证**等及一套非常易用的 JSP 标签，完全无缝与 Spring 其他技术协作。
* **Web-Struts 模块：**提供了与 Struts 无缝集成， Struts1.x 和 Struts2.x 都支持。

**2.5.4 Test 模块： Spring 支持 Junit 和 TestNG 测试框架，而且还额外提供了一些基于 Spring 的测试功能，比如在测试 Web 框架时，模拟 Http 请求的功能。**