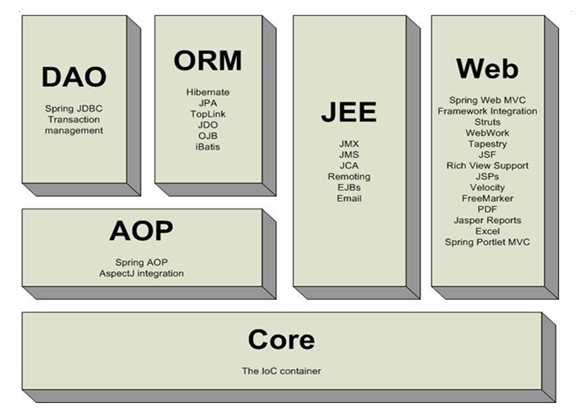
1、Spring框架组成

简单来说，Spring是一个轻量级的控制反转和面向切面的容器框架。

Spring的基本框架主要包含六大模块:DAO、ORM、AOP、JEE、WEB、CORE



**Spring DAO**:Spring提供了对JDBC的操作支持;如JdbcTemplate模板工具类。

**Spring ORM**：Spring可以与ORM框架整合。例如Spring整合Hibernate框架，其中Spring还提供HibernateDaoSupport工具类，简化了Hibernate的操作。

**Spring WEB：**Spring提供了对Struts、Springmvc的支持,支持WEB开发，Spring自身也提供了基于MVC的解决方案。

**Spring AOP:**Spring提供面向切面的编程，可以给某一层提供食物管理，例如在Service添加事务控制。

**Spring JEE;**J2EE规范的支持，例如EJB

**Spring Core**:提供IOC容器对象的创建和处理依赖对象关系。

2、IOC

**IOC容器：**就是具有依赖注入功能的容器，是可以创建对象的容器，IOC容器负责实例化、定位、配置应用程序中的对象及建立这些对象间的依赖。通常new一个实例，控制权由程序员控制，而“控制反转’是指new实例工作不由程序员来做而是交给spring容器来做。Spring中的BeanFactory是IOC容器的实际代表者。

DI依赖注入：在容器创建对象后，处理对象的依赖关系。

Spring依赖注入的四种方式;

1、set注入方式

2、静态工厂注入方式

3、构造方法注入方式

4、基于注解的方式

（1）持久层DAO层注解Repository中规定了名称，在Service层中声明名称必须一致。

（2）服务层Service层注解Service中规定了名称，在控制层中声明的名称必须一致。

（3）**注解可以简化配置，提升开发效率，但是也不利于后期维护。**

**@Resource与@Autowired注解的区别（注意是默认！！不是没有）**

**@Resource默认按照名称的方式进行bean匹配**，@Resource有两个重要的属性：name和type，而Spring将@Resource注解的name属性解析为bean的名字，而type属性则解析为bean的类型。所以，如果使用name属性，则使用byName的自动注入策略，而使用type属性时则使用byType自动注入策略。如果既不制定name也不制定type属性，这时将通过反射机制使用byName自动注入策略。

**@Autowired默认按照类型方式进行bean匹配，**默认情况下它要求依赖对象必须存在，如果允许null值，可以设置它的required属性为false。如果我们想使用按照名称（byName）来装配，可以结合@Qualifier注解一起使用。

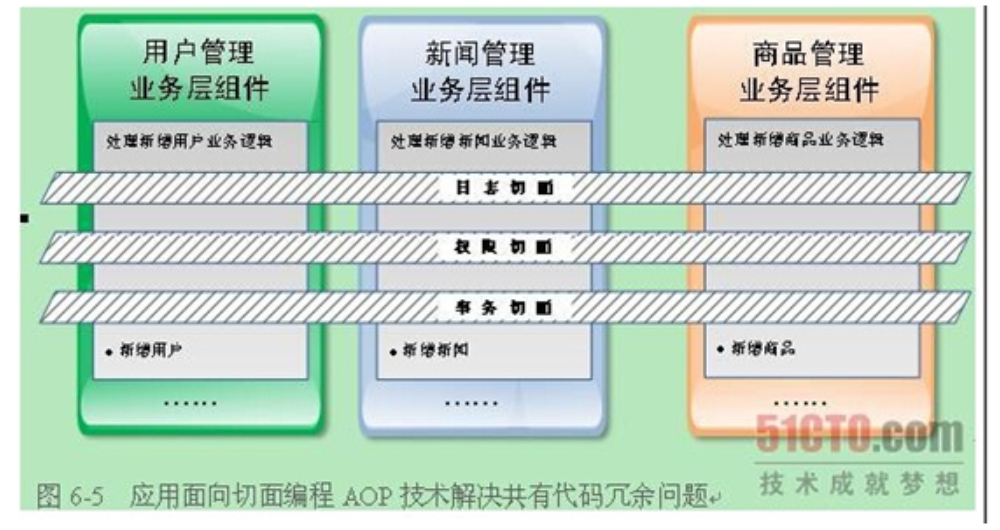
**@Resource**(javax.annotation.Resource)是J2EE的注解

**@Autowired**(org.springframework.beans.factory.annotation.Autowired)是spring的注解

Spring属于第三方的，J2EE是java自己的东西，使用@Resource可以减少代码和Spring之间的耦合。

3、AOP面向切面编程

AOP就是纵向的编程，如业务1和业务2都需要一个共同的操作，与其往来的每个业务都添加同样的代码，不如写一遍代码，让两个业务共同使用这段代码。在日常有订单管理、商品管理、资金管理、库存管理等业务，都会需要到类似日志记录、事务控制、权限控制、性能统计、异常处理及事务处理等。AOP把所有共有的代码全部抽取出来，放置到某个地方集中管理，然后在具体运行时，再由容器动态织入这些共有代码。



**切面（Aspect）：**其实就是共有功能的实现。

**通知（Advice）:**是切面的具体实现，以目标方法为参照点，根据放置的地方不同，可分为前置通知、后置通知、异常通知、最终通知、环绕通知，在实际应用中通常是切面类中的一个方法，具体通知属于哪类通知，同样是在配置中指定的。

**连接点（joinpoint）：**就是程序在运行过程中能够插入切面的地点，但Spring只支持方法级的连接点。

**切入点（Pointcut）：**用于定义通知应该切入到哪些连接点上。不同的通知通常需要切入到不同的连接点上，这种精准的匹配是由切入点的正则表达式来定义的。

**目标对象（Target）：**就是那些即将切入切面的对象，也就是那些被通知的对象。这些对象中已经只剩下干干净净的核心业务逻辑代码了，所有的共有功能代码等待AOP容器的切入。

**代理对象（Proxy）:**将通知应用到目标对象之后被动态创建的对象，可以简单的理解为，代理对象的功能等于目标对象的核心业务逻辑功能加上共有功能，代理对象对于使用者而言是透明的，是程序运行过程中的产物。

**织入（Weaving）:** 将切面应用到目标对象从而创建一个新的代理对象的过程。这个过程可以发生在编译期、类装载期及运行期，当然不同的发生点有着不同的前提条件。譬如发生在编译期的话，就要求有一个支持这种AOP实现的特殊编译器；发生在类装载期，就要求有一个支持AOP实现的特殊类装载器；只有发生在运行期，则可直接通过Java语言的反射机制与动态代理机制来动态实现。

4、Spring事务管理配置

Spring使用AOP配置事务管理由三个部分组成，分别是DataSource、TransactionManager和代理机制这三部分，无论哪种配置方式，一般变化的只是代理机制这部分。



Spring事务配置的五种方式：

**每个Bean都有一个代理**

**所有Bean共享一个代理基类**

**使用拦截器**

**使用tx标签配置的拦截器**



**全注解**

5、BeanFactory

Spring的本质是一个Bean工厂（BeanFactory）或者说bean容器，它按照我们的要求，生产我们需要的各种各样的bean,只是在生产bean过程中,需要解决bean之间的依赖问题，于是引入了DI（依赖注入）技术，也就是说依赖注入是BeanFactory生产bean时为了解决bean之间的依赖的一种技术而已。

6、Hibernate和MyBatis

Hibernate是建立在若干POJO通过xml映射文件（或注解）提供的规则映射到数据库表上 的。我们可以通过POJO直接操作数据库的数据，它提供的是一种全表映射模型。

**Hibernate的好处**：

1，消除了代码的映射规则，它全部分离到了xml或者注解里面去配置。

2，无需在管理数据库连接，它也配置到xml里面了。

3，一个会话中不需要操作多个对象，只需要操作Session对象。

4，关闭资源只需要关闭一个Session便可。

**Hibernate的缺点：**

1、全表映射带来不便，比如更新时需要发送所有字段。

2、无法根据不同的条件组装不同的SQL

3、对多表关联和复杂的sql查询支持较差，需要自己写sql,返回后，需要自己封装为pojo。

4、不能有效的支持存储过程。

5、HQL的性能较差，大型互联网系统往往需要优化sql,而hibernate做不到。

**Mybatis：**

为了解决Hibernate的不足，Mybatis出现了，Mybatis是半自动的框架，是因为它需要手工匹配提供POJO，sql和映射关系。

Mybatis需要提供的映射文件包含了一下三个部分：sql，映射规则，pojo。

1、需要自己动手编写sql,虽然配置多，但可以配置动态sql,解决了hibernate表名根据时间变化、不同条件下列不一样的问题。

2、可以对sql进行优化，通过配置决定你的sql映射规则

3、支持存储过程，对于一些复杂和需要优化性能的sql查询它就很方便

**什么时候使用Hibernate，Mybatis？**

Hibernate作为留下的Java orm框架，它确实编程简易，需要我们提供映射的规则，完全可以通过IDE生成，同时无需编写sql确实开发效率优于Mybatis。此外Hibernate还提供了缓存，日志，级联等强大的功能，但是Hibernate的缺陷也是十分明显，**多表关联复杂sql，数据系统权限限制，根据条件变化的sql，存储过程等场景使用Hibernate十分不方便**，而性能又难以通过sql优化，所以注定了Hibernate只适用于在场景不太复杂，要求性能不太苛刻的时候使用。

如果你需要一个灵活的，可以动态生成映射关系的框架，那么Mybatis确实是一个最好的选择。它几乎可以替代jdbc，拥有动态列，动态表名，存储过程支持，同时提供了简易的缓存，日志，级联。但是它的缺陷是需要你提供映射规则和sql，所以开发工作量比hibernate要大些。

**jdbc,mybatis,hibernate的区别？**

1、JDBC是较底层的持久层操作方式，而Hibernate和MyBatis都是在JDBC的基础上进行了封装使其更加方便程序员对持久层的操作。

2、，JDBC就是简单的建立数据库连接，然后创建statement，将sql语句传给statement去执行，如果是有返回结果的查询语句，会将查询结果放到ResultSet对象中，通过对ResultSet对象的遍历操作来获取数据；Hibernate是将数据库中的数据表映射为持久层的Java对象，对sql语句进行修改和优化比较困难；MyBatis是将sql语句中的输入参数和输出参数映射为java对象，sql修改和优化比较方便.

3、如果进行底层编程，而且对性能要求极高的话，应该采用JDBC的方式；

如果要对数据库进行完整性控制的话建议使用Hibernate；

如果要灵活使用sql语句的话建议采用MyBatis框架。

7、Redis

Redis 是一个基于内存的高性能key-value数据库。很像memcached,整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库操作flush到硬盘上进行保存。

Redis的出色之处不仅仅是性能，Redis最大的魅力是支持保存多种数据结构，此外单个value的最大限制是1G,而memcached只能保存1M的数据。

Redis的主要缺点是数据库容量受到物理内存的限制，不能用作海量数据的高性能读写，因此Redis适合的场景主要局限在较小数据量的高性能操作和运算上。

**使用redis有哪些好处？**

(1) 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

(2) 支持丰富数据类型，支持string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性。

**redis相比memcached有哪些优势？**

1、memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型

2、redis的速度比memcached快很多

3、redis可以持久化其数据

**Memcache与Redis的区别都有哪些？**

**1、**存储方式 Memecache把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小。 Redis有部份存在硬盘上，这样能保证数据的持久性。

2、数据支持类型 Memcache对数据类型支持相对简单。 Redis有复杂的数据类型。

3、它们之间底层实现方式 以及与客户端之间通信的应用协议不一样，Redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

**如何保证redis中的数据都是热点数据？**

redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略（回收策略）。redis 提供 6种数据淘汰策略：

1、从已设置过期时间的数据集，最少使用淘汰

2、从已设置过期时间的数据集，中挑选将要过期的数据淘汰。

3、从已设置过期时间的数据集，中任意选择数据淘汰

4、从数据集中挑选最近最少使用的数据淘汰

5、从数据集中任意选择数据淘汰

6、禁止驱逐数据

**Redis是单进程单线程的？**

redis利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销，因为在内存中加锁的同步开销太大。ZooKeeper就是多线程的

8、连接池

最常用的是C3P0连接池，

**什么是数据库连接池?**

所谓的连接池就是用来存放数据库连接的一个容器, 用于在整个程序中共享, 可以减少开关连接的次数, 提高程序执行的效率。

**为什么要使用数据库连接池?**

在开发中, 频繁的开关连接是一件非常消耗资源和时间的操作, 会导致程序执行的效率非常的低下，创建连接要花的时时间一般比对数据库的数据增删改查的时间还要长！ 因此我们可以在程序中创建容器(连接池), 在程序启动时就初始化一批连接放在连接池中, 当用户需要时,就从连接池中获取一个连接, 用完连接之后再将连接还回连接池中, 这样就可以实现连接的复用, 减少连接开关的次数, 提高程序执行的效率!!

9、Springboot

用来简化spring应用的初始搭建以及开发过程 使用特定的方式来进行配置，（properties或yml文件）

1、创建独立的spring引用程序 main方法运行

2、嵌入的Tomcat 无需部署war文件

3、简化maven配置

4、自动配置spring添加对应功能starter自动化配置

（嵌入tomcat和web开发需要servlet与jsp支持的starter、数据库支持的starter、redis数据库的支持、mybatis的支持）

**Springboot自动配置的原理？**

在spring程序main方法中 添加@SpringBootApplication或者@EnableAutoConfiguration，会自动去maven中读取每个starter中的spring.factories文件 该文件里配置了所有需要被创建spring容器中的bean

10、SpringClould

**什么是微服务？**

以前的模式是 所有的代码在同一个工程中 部署在同一个服务器中 同一个项目的不同模块不同功能互相抢占资源.

微服务 将工程根据不同的业务规则拆分成微服务 微服务部署在不同的机器上 服务之间进行相互调用Java微服务的框架有 dubbo（只能用来做微服务），spring cloud（提供了服务的发现，断路器等）

**springcloud如何实现服务的注册和发现？**

服务在发布时 指定对应的服务名（服务名包括了IP地址和端口），将服务注册到注册中心（eureka或者zookeeper），这一过程是springcloud自动实现 只需要在main方法添加@EnableDisscoveryClient 同一个服务修改端口就可以启动多个实例

**调用方法：**

传递服务名称通过注册中心获取所有的可用实例，通过负载均衡策略调用（ribbon和feign（[feɪn]））对应的服务

**Ribbon和feign的区别？**

Ribbon是一个基于 HTTP 和 TCP 客户端的负载均衡器, 它可以在客户端配置服务端列表，然后轮询请求以实现均衡负载。Ribbon需要自己构建http请求, 步骤相当繁琐。

Feign 是一个使用起来更加方便的 HTTP 客戶端，使用起来就像是调用自身工程的方法，而感觉不到是调用远程方法。Feign则是在Ribbon的基础上进行了一次改进，采用接口的方式，将需要调用的其他服务的方法定义成抽象方法即可，不需要自己构建http请求。

**Springclould断路器的作用？**

当一个服务调用另一个服务由于网络原因或者自身原因出现问题时，调用者就会等待被调用者的响应，当更多的服务请求到这些资源时，导致更多的请求等待，这样就会发生连锁效应，

断路器就是解决这一问题。

**断路器的三种状态：**

**全开：** 一定时间内 达到一定的次数无法调用 并且多次检测没有恢复的迹象，断路器完全打开，那么下次请求就不会请求到该服务

**半开**：短时间内，有恢复迹象，断路器会将部分请求发给该服务，当能正常调用时，断路器关闭

**关闭**：当服务一直处于正常状态，能正常调用，断路器关闭

11、Shiro（[ʃɪrɒ]）

Apache Shiro是Java的一个安全框架, 对比Spring Security，可能没有Spring Security做的功能强大，但是在实际工作时可能并不需要那么复杂的东西。

Shiro可以帮助我们完成：认证、授权、加密、会话管理、与Web集成、缓存等。

Shrio框架中，有最重要的三个组件，Subject、SecurityManager、Realm

Subject:本质上就是当前访问用户的抽象描述

SecurityManage: 是Shiro架构中最核心的组件，通过它可以协调其他组件完成用户认证和授权,可以这么理解，SecurityManager就是Shrio框架的控制器。

Realm：定义了访问数据的方式，用来连接不同的数据源

原理：

Shrio的核心是java servlet规范中的filter,通过配置拦截器，使用拦截器链来拦截请求，如果允许访问，则通过，通常情况下，系统的登录、退出会配置拦截器，登录的时候，调用调用subject.login(token),token是用户验证信息，这个时候会在Realm中进行认证。会把用户提交的验证信息与数据库中存储的认证信息进行比较，一致则允许访问，在客户端存cookies,在服务器端存储session信息，退出的时候，调用subject.Logout()会清除会话信息。

1、认证：通过调用Subject.login（token）方法开始用户认证流程

2、授权：shrio访问授权有三种方式：api调用、java注解、jsp标签

Shiro框架只能控制操作权限，不能控制数据权限。

12、Activiti

1：什么是工作流，工作流的核心对象是什么，activiti共操作数据库多少张表?

工作流就是多个参与者，按照某种预定义的规则，传递业务信息，进行审核的功能一个框架（Activiti）核心对象：processEngine，调用Service，从而操作数据库的表，23张表

2、工作流中RepositoryService、RuntimeService、TaskService、HistoryService分别表示什么操作？

RepositoryService:流程定义和部署对象

RuntimeService：执行管理，包括流程实例和执行对象（正在执行）

TaskService：执行任务相关的（正在执行）

HistoryService：历史管理

IdentityService：Activiti表的用户角色组

3：流程实例和执行对象的区别？

流程从开始到结束的最大分支，一个流程中，流程实例只有1个

执行对象，就是按照流程定义的规则执行一次的操作，一个流程中，执行对象可以有多个

4、activiti工作流中，如果一个任务完成后，存在多条连线，应该如何处理？

当一个任务完成之后，根据这几条连线的条件和设置流程变量

13、ActiveMQ

消息中间件，可以在分布式系统的不同服务之间进行消息的发送和接收

它的出现让系统解耦