# 概念

## Spring

Spring的两大特性IOC（DI）、Aop。

## 数据库连接池

### dbcp

### c3p0

### druid

## 事务

事务是指作为单个逻辑工作单元执行的一系列[操作](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C/33052)，要么完全地执行，要么完全地不执行。

### 事务的特性

1. 原子性（Atomicity）：事务是一个原子操作，由一系列动作组成。事务的原子性确保动作要么全部完成，要么完全不起作用。
2. 一致性（Consistency）：一旦事务完成（不管成功还是失败），系统必须确保它所建模的业务处于一致的状态，而不会是部分完成部分失败。在现实中的数据不应该被破坏。
3. 隔离性（Isolation）：可能有许多事务会同时处理相同的数据，因此每个事务都应该与其他事务隔离开来，防止数据损坏。
4. 持久性（Durability）：一旦事务完成，无论发生什么系统错误，它的结果都不应该受到影响，这样就能从任何系统崩溃中恢复过来。通常情况下，事务的结果被写到持久化存储器中。

### 事务与spring结合

Spring不负责管理事务，但提供了各种事务管理器，为jdbc和heibernate等不同平台提供了对应的事务管理器。Spring中事务的配置方式有如下两种：

#### 编程式事务

当系统需要明确的，细粒度的控制各个事务的边界，应选择编程式事务。

#### 声明式事务

当系统对于事务的控制粒度较粗时，应该选择申明式事务，通过<tx>标签和<aop>切面形式在xml中进行配置。可通过spring提供的jdbc或heibernate事务管理器进行管理。

##### Jdbc事务

如果应用程序中直接使用JDBC来进行持久化，DataSourceTransactionManager会为你处理事务边界。为了使用DataSourceTransactionManager，你需要使用如下的XML将其装配到应用程序的上下文定义中：

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

##### Hibernate事务

如果应用程序的持久化是通过Hibernate实习的，那么你需要使用HibernateTransactionManager。对于Hibernate3，需要在Spring上下文定义中添加如下的<bean>声明：

<bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory" ref="sessionFactory" />

</bean>

##### Spring中事务的传播行为

传播行为有七种最常见为以下两种其它请自行百度：https://blog.csdn.net/mini\_jike/article/details/46275079

PROPAGATION\_REQUIRED：支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。如：增删改

PROPAGATION\_SUPPORTS：支持当前事务，如果当前没有事务，就以非事务方式执行。如：查看

##### Spring中事务的隔离级别：

https://blog.csdn.net/zht741322694/article/details/78676964

isolation\_default使用数据库默认的事务隔离级别

isolation\_read\_uncommitted允许读取尚未提交的修改，可能导致脏读、幻读和不可重复读

isolation\_read\_committed 允许从已经提交的事务读取，可防止脏读、但幻读，不可重复读仍然有可能发生

isolation\_repeatable\_read 对相同字段的多次读取的结果是一致的，除非数据被当前事务自生修改。可防止脏读和不可重复读，但幻读仍有可能发生

isolation\_serializable完全服从acid隔离原则，确保不发生脏读、不可重复读、和幻读，但执行效率最低。

##### Spring中<tx:method/>配置解析

| **属性** | **是否需要？** | **默认值** | **描述** |
| --- | --- | --- | --- |
| name | 是 |  | 与事务属性关联的方法名。通配符（\*）可以用来指定一批关联到相同的事务属性的方法。如：'get\*'、'handle\*'、'on\*Event'等等。 |
| propagation | 不 | REQUIRED | 事务传播行为 |
| isolation | 不 | DEFAULT | 事务隔离级别 |
| timeout | 不 | -1 | 事务超时的时间（以秒为单位） |
| read-only | 不 | false | 事务是否只读？ |
| rollback-for | 不 |  | 将被触发进行回滚的 Exception(s)；以逗号分开。 如：'com.foo.MyBusinessException,ServletException' |
| no-rollback-for | 不 |  | *不* 被触发进行回滚的 Exception(s)；以逗号分开。 如：'com.foo.MyBusinessException,ServletException' |

##### 利用aop将配置的事务进行切入

<!--1.声明式 配置事务管理器:注意这里的dataSource和SqlSessionFactoryBean的dataSource要一致，不然事务就没有作用了 -->

<!-- 配置事务的传播特性 -->

<tx:advice id=*"userTxAdvice"* transaction-manager=*"transactionManager"*>

<tx:attributes>

<tx:method name=*"delete\*"* propagation=*"REQUIRED"* read-only=*"false"*

rollback-for=*"java.lang.Exception"* no-rollback-for=*"java.lang.RuntimeException"* />

<tx:method name=*"insert\*"* propagation=*"REQUIRED"* read-only=*"false"*

rollback-for=*"java.lang.RuntimeException"* />

<tx:method name=*"update\*"* propagation=*"REQUIRED"* read-only=*"false"*

rollback-for=*"java.lang.Exception"* />

<tx:method name=*"find\*"* propagation=*"SUPPORTS"* />

<tx:method name=*"get\*"* propagation=*"SUPPORTS"* />

<tx:method name=*"select\*"* propagation=*"SUPPORTS"* />

<tx:method name=*"batch\*"* propagation=*"REQUIRED"* read-only=*"false"* />

</tx:attributes>

</tx:advice>

<!-- 切面配置 -->

<aop:config>

<aop:pointcut id=*"transactionPointcut"*

expression=*"execution(\* com.future.service.\*.impl.\*.\*(..))"* />

<aop:advisor pointcut-ref=*"transactionPointcut"*

advice-ref=*"userTxAdvice"* />

</aop:config>

<!--2.注解式事务配置,即哪些方法需要事务就在那个方法上写一个@Transactional注解）-->

<!-- 开启事务注解，标注@Transactional的类和方法将具有事务性 -->

<tx:annotation-driven transaction-manager=*"transactionManager"* />

## Shiro

<https://www.w3cschool.cn/shiro/>

<https://www.cnblogs.com/tohxyblog/p/6870261.html>

https://blog.csdn.net/oppoppoppo/article/details/53432202

shiro缓存和会话的区别： **shiro每个授权都会通过realm获取权限信息，为了提高访问速度需要添加缓存，第一次从realm中读取权限数据，之后不再读取。**

缓存失效逻辑：**如果用户正常退出 缓存清空。如果用户非正常退出比如关闭浏览器 缓存清空。如果session到期，缓存清空，设置一般在缓存的配置文件中。如果修改了用户的权限，而用户不退出系统，修改的权限不会生效，需要手动调用realm的clearCache方法清除。**

### 缓存

shiro的缓存和会话session的区别：<http://syq891015.iteye.com/blog/2357769>

#### shiro缓存的实现

[Shiro缓存使用Redis、Ehcache、自带的MpCache实现的三种方式实例](https://www.cnblogs.com/zfding/p/8536480.html)：https://www.cnblogs.com/zfding/p/8536480.html

##### ehcache

将数据存放在磁盘或内存中，使用时需要配置xml文件

##### redis

将数据存放在redis数据库，在利用IOC创建对象时，需指定数据库的连接信息

##### mpcache

此缓存为shiro内置缓存方式，数据存放在内容中，使用非常简单。

<!-- 配置shiro自带的缓存管理器 -->

<bean id = "shiroSpringCacheManager"class="org.apache.shiro.cache.MemoryConstrainedCacheManager"/>

### 会话session

## Dubbo

## Webservice

### 概念

webService主要用于向其他系统提供接口以便调用，系统间可能开发语言等完全不同，根据约定的接口规范，调用者传递相关参数进行接口调用，服务方根据传入的条件进行业务处理并进行结果返回。

<http://www.cnblogs.com/xiaocai0923/archive/2011/12/30/2308154.html>

https://www.cnblogs.com/gissuifeng/p/3726900.html

### 服务器端的开发

<https://www.cnblogs.com/xiaochangwei/p/4969448.html>

<https://www.cnblogs.com/frankliiu-java/articles/1641949.html>

[https://blog.csdn.net/zl834205311/article/details/51612207 4](https://blog.csdn.net/zl834205311/article/details/51612207%204)种不同的外放方式

开发步骤：

1. 创建webproject
2. 创建class即准备外放的提供服务的webservice接口。该类需添加注解（@webService） 表示该类被发布成了一个ws。该注解由jdk进行提供。
3. 在ws类中创建方法。方法上添加注解(@webMethod(exclude=true))表示此方法不会被发布。其次使用static修饰的静态方法也不会被发布。final方法也不会被发布。
4. 调用main方法外放对应的webservice接口。需要外放的方法创建成功以后，使用Endpoint （Endpoint – 此类为端点服务类，该类也由jdk进行提供）将ws类中的指定方法进行发布。例如：

Endpoint.publish(“url”,new WSObject());

参数1：为服务器端ip，若设置的不是接口提供者的ip，而是其他服务器ip时，接口的外放正常进行。但是使用wsdl进行校验时，则由于访问的ip并不是接口提供者的ip则会因为访问不到服务而空白无法正常显示xml格式文件。Ip必须是服务提供的ip，服务名称可以随便写。

参数2：外放的类，该类为接口的实现类

1. 发布成功以后，在浏览器输入对应网址进行校验，即外放的url后面加 ?wsdl

### 客户端调用

<https://blog.csdn.net/woshishabiaaaaa/article/details/24868323>

## Restful

### 概念

REST是一种架构风格，其核心是面向资源，REST专门针对网络应用设计和开发方式，以降低开发的复杂性，提高系统的可伸缩性。REST提出设计概念和准则为：

1. 网络上的所有事物都可以被抽象为资源(resource)
2. 每一个资源都有唯一的资源标识(resource identifier)，对资源的操作不会改变这些标识，其核心操作只有GET,PUT,POST,DELETE。
3. 所有的操作都是无状态的

由此通过资源统一标识符和操作符即可确定用户的请求。

## Soap

### 概念

基于xml的简单对象访问协议。它是轻型协议，用于分散的、分布式计算环境中交换信息。SOAP有助于以独立于平台的方式访问对象、服务和服务器。它借助于XML，提供了HTTP所需的扩展。

### SOAP协议规范

第一部分：SOAP封装（Envelop）定义了一个的框架（描述消息的内容多少、谁发送、谁应当接受、处理，以及如何处理它们）。

第二部分：SOAP编码规则（Encoding Rules）定义了可选数据编码规则，用于表示应用程序定义的数据类型和直接图表，以及一个用于序列化非语法数据模型统一标准。

第三部分：SOAP RPC表示（RPC Representation）定义一个远程调用风格（请求/响应）信息交换的模式。

第四部分：SOAP绑定（Binding）定义了SOAP和HTTP之间的绑定和使用底层协议的交换。

SOAP协议可以简单地理解为：SOAP=RPC+HTTP+XML，即采用HTTP作为通信协议，RPC（Remote Procedure Call Protocol - 远程过程调用协议）作为一致性的调用途径，XML作为数据传送的格式，从而允许服务提供者和服务客户经过防火墙在Internet上进行通信交互。

## http

### 概念

http是标准超文本传输协议。使用对参数进行编码并将参数作为键值对传递，还使用关联的请求语义。每个协议都包含一系列HTTP请求标头及其他一些信息，定义客户端向服务器请求哪些内容，服务器用一系列HTTP响应标头和所请求的数据进行响应。HTTP-GET 使用 MIME 类型application/x-www-form-urlencoded（将追加到处理请求的服务器的 URL 中）以 URL 编码文本的形式传递其参数。 URL 编码是一种字符编码形式，可确保传递的参数中包含一致性文本，例如将空格编码为 %20，其它符号转换为%XX,其中XX为该符号以16进制表示的ASCII（或ISOLatin-1）值。 追加的参数也称为查询字符串;HTTP-POST参数也是 URL 编码的，但是，键/值对是在实际的 HTTP 请求消息内部传递的，而不是作为 URL 的一部分进行传递。