Web后端知识整理

SSH

定义

简单的说，SSH是一种网络协议，主要用于客户端与远程主机的安全链接和交互。

安全链接的过程是：
1.远程主机端收到客户端的登陆请求时先发送自己的公钥给客户端
2.客户端用拿到的公钥加密用户名和密码，然后发送给远程主机
3.远程主机用自己的密钥解密收到的用户名和密码，然后校验用户名和密码是否正确，如果正确则登陆成功。

2.客户端免密登陆远程主机
如果只是通过第一步，以后的每次登陆都需要输入登陆密码，非常麻烦。幸运的是SSH提供了公钥登陆（免密登陆）
公钥登录的流程如下：
1.客户端在自己本地生成一对公钥密钥文件，然后将公钥存储在远程主机上
2.客户端登陆时，远程主机会随机生成一串字符串发送给客户端
3.客户端用自己的密钥将收到的字符串加密，并返回给远程主机
4.远程主机利用公钥解密收到的加密字符串，如果解密成功并且与发送的一致则直接免密登陆。

1.Ajax

参考：图解Ajax工作原理 - .长卿 - 博客园

定义

Ajax指Asynchronous JavaScript and XML（异步的 JavaScript 和 XML），最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。

AJAX(Asynchronous JavaScript and XML)，它不是一门新的语言或技术，而是多种 技术的综合，包括:Javascript、XHTML、CSS、DOM、XML、XSTL、XMLHttpRequest
而实现的原理基础就是：网页DOM对象可以精确地对网页中的部分内容进行操作、XML作为单纯的数据存储载体使得客户端与服务器交换的只是网页内容的数据而没有网页样式等等的附属信息、XMLHttpRequest是与浏览器本身内置的request相互独立的与服务器交互的请求对

象。

步骤

在会话的开始，浏览器加载 Ajax 引擎请求动作通过 JavaScript 调用 Ajax 引擎来代替. 引擎负责绘制用户界面以及与服务器端通讯。Ajax 引擎采用异步交互过程–不用等待服务器的通讯。

过程

1.要使用Ajax技术，基础中的基础，就是要创建一个XMLHttpRequest对象，无它就没有异步传输的可能

2：在网页中为某些事件的响应绑定异步操作：通过上面创建的xmlhttp对象传输请求、携带数据。在发出请求前要先定义请求对象的method、要提交给服务器中哪个文件进行请求的处理、要携带哪些数据、是否异步。

3：服务器收到请求后，把附带的数据作为输入传给处理请求的文件，例如这里：把fname=Henry&lname=Ford作为输入，传给_ *try*ajax/demo_get2.php_ 这个文件。然后文件根据传入的数据做出处理，最终返回结果，通过response对象发回去。客户端根据xmlhttp对象来获取response内容，然后调用DOM对象根据response内容来局部修改网页内容。

工作原理

在会话的开始，浏览器加载 Ajax 引擎请求动作通过 JavaScript 调用 Ajax 引擎来代替. 引擎负责绘制用户界面以及与服务器端通讯。Ajax 引擎采用异步交互过程–不用等待服务器的通讯。

XMLHttpRequest

XMLHttpRequest 对象用于在后台与服务器交换数据。
XMLHttpRequest 对象能够：

- 在不重新加载页面的情况下更新网页
- 在页面已加载后从服务器请求数据
- 在页面已加载后从服务器接收数据
- 在后台向服务器发送数据

所有现代的浏览器都支持 XMLHttpRequest 对象。

XMLHttpRequest 对象以及方法

一个 JavaScript 对象。是 Ajax 的核心。该对象的方法和属性。

open():建立到服务器的新请求。 send():向服务器发送请求。 abort():退出当前请求。
readyState:提供当前 HTML 的就绪状态。 responseText:服务器返回的请求响应文本。

同步与异步

普通 B/S 模式(同步)AJAX 技术(异步) 同步:提交请求->等待服务器处理->处理完毕返回 这个期间客户端浏览器不能干任何事， 而异步则是 请求通过事件触发->服务器处理->处理完毕 同步是阻塞模式，异步是非阻塞模式。

同步的概念应该是来自于OS中关于同步的概念:不同进程为协同完成某项工作而在先后次序上调整(通过阻塞,唤醒等 方式).同步强调的是顺序性.谁先谁后.

异步则不存在这种顺序性.

同步(发送方发出数据后，等接收方发回) 浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新 内容,进行下一步操作。

异步(发送方发出数据后，不等接收方发回响应) 浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用 户看到新内容。

ajax框架

jQuery

websocket和ajax轮询

Websocket是HTML5中提出的新的协议，注意，这里是协议，可以实现客户端与服务器端的通信，实现服务器的推送功能。 其优点就是，只要建立一次连接，就可以连续不断的得到服务器推送的消息，节省带宽和服务器端的压力。

ajax轮询模拟长连接就是每个一段时间(0.5s)就向服务器发起ajax请求，查询服务器端是否有数据更新。其缺点显而易见，每次都要建立HTTP连接，即使需要传输的数据非常少，所以这样很浪费带宽

2.Bean

什么是Bean - 林妹妹的后花园 - CSDN博客

定义

一、Bean

1、Java面向对象，对象有方法和属性，那么就需要**对象实例来调用方法和属性（即实例化）；**

2、凡是有方法或属性的类都需要实例化，这样才能具象化去使用这些方法和属性；

3、规律：**凡是子类及带有方法或属性的类都要加上注册Bean到Spring IoC的注解；**
（@Component , @Repository , @ Controller , @Service , @Configration**）**

4、把Bean理解为类的代理或代言人（实际上确实是通过反射、代理来实现的），这样它就能代表类拥有该拥有的东西了

5、我们都在微博上@过某某，对方会优先看到这条信息，并给你反馈，那么在Spring中，**你标识一个@符号，那么Spring就会来看看，并且从这里拿到一个Bean（注册）或者给出一个Bean（使用）**

二、注解分为两类：
1、一类是使用Bean，即是把已经在xml文件中配置好的Bean拿来用，完成属性、方法的组装；比如@Autowired , @Resource，可以通过byTYPE（@Autowired）、byNAME（@Resource）的方式获取Bean；

2、一类是注册Bean,@Component , @Repository , @ Controller , @Service , @Configration这些注解都是把你要实例化的对象转化成一个Bean，放在IoC容器中，等你要用的时候，它会和上面的@Autowired , @Resource配合到一起，把对象、属性、方法完美组装。

三、@Baan是啥

```
Indicates that a method produces a bean to be managed by the Spring container.

<h3>Overview</h3>

<p>The names and semantics of the attributes to this annotation are
intentionally
similar to those of the {@code <bean/>} element in the Spring XML schema. For
example:

<pre class="code">
    @Bean
    public MyBean myBean() {
        // instantiate and configure MyBean obj
        return obj;
    }</pre>
```

思是@Bean明确地指示了一种方法，什么方法呢——**产生一个bean的方法，并且交给Spring容器管理；**从这我们就明白了为啥@Bean是放在方法的注释上了，因为它很明确地告诉被注释的方法，你给我产生一个Bean，然后交给Spring容器，剩下的你就别管了。

bean的行为

基本行为

singleton
在整个Spring IoC 容器中，使用 singleton 定义的Bean将只有一个实例

prototype
原型模式,每次通过容器的getBean 方法获取prototype定义的Bean 时,都将产生一个新的Bean实例

非基本行为

request
对于每次HTTP请求，使用request定义的Bean都将产生一个新的实例，每次HTTP请求都将产生不同的Bean实例,该作用域仅在给予web的Spring ApplicationContext情形下有效

session
对于每次HTTP Session ,使用session定义的Bean都将产生一个新实例，该作用域仅在给予web的Spring ApplicationContext情形下有效

global session
每个全局得HTTP Session对应一个Bean实例，该作用域仅在给予web的Spring ApplicationContext情形下有效

Bean factory

BeanFacotry是spring中比较原始的Factory。如XMLBeanFactory就是一种典型的BeanFactory。原始的BeanFactory无法支持spring的许多插件，如AOP功能、Web应用等。

ApplicationContext接口,它由BeanFactory接口派生而来，ApplicationContext包含BeanFactory的所有功能，通常建议比BeanFactory优先*

_ BeanFactory是接口，提供了OC容器最基本的形式，给具体的IOC容器的实现提供了规范，
_ _ _ FactoryBean也是接口，为IOC容器中Bean的实现提供了更加灵活的方式，FactoryBean

在IOC容器的基础上给Bean的实现加上了一个简单工厂模式和装饰模式(如果想了解装饰模式参考： 修饰者模式(装饰者模式，Decoration) _我们可以在getObject()方法中灵活配置。其实在Spring源码中有很多FactoryBean的实现类.

区别：BeanFactory是个Factory，也就是IOC容器或对象工厂，FactoryBean是个Bean。在Spring中，**所有的Bean都是由BeanFactory(也就是IOC容器)来进行管理的**。但对FactoryBean而言，**这个Bean不是简单的Bean，而是一个能生产或者修饰对象生成的工厂Bean,它的实现与设计模式中的工厂模式和修饰器模式类似**

javabean 与springbean

什么是JavaBean:

JavaBean是一种JAVA语言写的可重用组件。JavaBean符合一定规范写的Java类，是一种规范。它的方法命名，构造以及行为必须符合特定的要求：
1.所有属性为private

2.这个类必须具有一个公共的（public）无参构造函数

3.private属性必须提供public的getter和setter来给外部访问，并且方法的命名也必须遵循一定的命名规范

4.这个类是可序列化的，要实现serializable接口

什么是SpringBean:

SpringBean是受Spring管理的对象_ 所有能受Spring容器管理的对象都可以成为SpringBean.

二者之间的区别：

用处不同：传统javabean更多地作为值传递参数，而spring中的bean用处几乎无处不在，任何组件都可以被称为bean

写法不同：传统javabean作为值对象，要求每个属性都提供getter和setter方法；但spring中的bean只需为接受设值注入的属性提供setter方法

生命周期不同：传统javabean作为值对象传递，不接受任何容器管理其生命周期；spring中的bean有spring管理其生命周期行为

<Bean>的depends-on属性是什么

Depends-on 用于当前 Bean 初始化之前显示的强制一个或多个 bean 被初始化depends-on 属性
Depends-on 用于当前 Bean 初始化之前显示的强制一个或多个 bean 被初始化

BeanWrapper类是什么

BeanWrapper 类是一个对 JavaBean 进行各种操作的工具类
BeanWrapper 本身是一个接口 BeanWrapperImpl 实现了 BeanWrapper

3.DWR

DWR(Direct Web Remoting)是一个 WEB 远程调用框架.
可以在客户端利用 JavaScript 直接调用服务端的 Java 方法并返回值给 JavaScript DWR 根据 Java 类来动态生成 JavaScrip 代码.
支持 Dom Trees,支持 Spring,支持 commons-logging

4.Hibernate

Hibernate是一种ORM框架，全称为 Object_Relative DateBase-Mapping，在Java对象与关系数据库之间**建立某种映射，以实现直接存取Java对象**!
它提供了 强大，高效的将 JAVA 对象进行持久化操作的服务 。
既然Hibernate是关于Java对象和关系数据库之间的联系的话，也就是**我们MVC中的数据持久层->在编写程序中的DAO层...**
首先，我们来回顾一下我们在DAO层写程序的历程吧：
1 在DAO层操作XML，将数据封装到XML文件上，读写XML文件数据实现CRUD
2 在DAO层使用原生JDBC连接数据库，实现CRUD
3 嫌弃JDBC的Connection\Statement\ResultSet等对象太繁琐，使用对原生JDBC的封装组件–>DbUtils组件

Hibernate对JDBC访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。
Hibernate是一个基于JDBC的主流持久化框架，是一个优秀的ORM实现，它很大程度的简化了dao层编码工作。
总结：Hibernate是企业级开发中的主流框架，映射的灵活性很出色。它支持很多关系型数据库。

操作流程

1.读取并解析配置文件

2.读取并解析映射信息，创建SessionFactory

3.打开Sesssion

4.创建事务Transation

5.持久化操作6.提交事务7.关闭Session

8.关闭SesstionFactory

优点

开源和免费的 License，我可以在需要的时候研究源代码，改写源代码，进行功能的定制。轻量级封装，避免引入过多复杂的问题，调试容易，也减轻程序员的负担。 具有可扩展性，API 开放，当本身功能不够用的时候，可以自己编码进行扩展。

1. 对JDBC访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。

2. Hibernate是一个基于JDBC的主流持久化框架，是一个优秀的ORM实现。他很大程度的简化DAO层的编码工作

3. hibernate的性能非常好，因为它是个轻量级框架。映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各
种复杂关系。

作用

Hibernate扮演的是数据持久层 它的作用是实现持久化对象和数据库表
之间的映射，形成持久化对象和数据库表中数据的一个转换平台

JPA

JPA Java Persistence API:sun官方提出的Java持久化规范，Hibernate依据它实现。 JPA维护了一个Persistence Context， 在其中维护实体的生命周期，主要包含:
1.ORM元数据，JPA支持annotation或者xml两种形式来描述ER映射

2. 实体操作API，实现对实体对象的CRUD操作3. 查询语言。约定了面向对象的查询语言JPQL

抓取策略get 与load

抓取策略是指当应用程序需要利用关联关系进行对象获取的时候。

立即检索:当执行某行代码的时候,马上发出SQL语句进行查询（get()）

延迟检索:当执行某行代码的时候,不会马上发出SQL语句进行查询.当真正使用这个对象的时候才会发送SQL语句（load()）

如果没有匹配的数据库记录，load()方法可能抛出无法恢复的异常(unrecoverable exception)。 如果类的映射使用了代 理(proxy)，load()方法会返回一个未初始化的代理，直到你调用该代理的某方法时才会去访问数据库。 若你希望在某对象中创建 一个指向另一个对象的关联，又不想在从数据库中装载该对象时同时装载相关联的那个对象，那么这种操作方式就用得上的了。 如 果为相应类映射关系设置了batch-size， 那么使用这种操作方式允许多个对象被一批装载(因为返回的是代理，无需从数据库中抓 取所有对象的数据)。 如果你不确定是否有匹配的行存在，应该使用get()方法，它会立刻访问数据库，如果没有对应的行，会返回null。

对象的三个状态

Transient(瞬态) ， persistent(持久态)和 Detached(游历态)

1.Transient（瞬态）表示该实体对象在内存中是自由的，也就是说和数据库没有任何关系。当new 一个实体对象后，这个对象处于临时状态，即这个对象只是一个保存临时数据的内 存区域，如果没有变量引用这个对象，则会被 jre 垃圾回收机制回收。这个对象所保存的数据与数据库没有任何关系，除非通过 Session 的 save 或者 SaveOrUpdate 把临时对 象与数据库关联，并把数据插入或者更新到数据库，这个对象才转换为持久对象;

2.Persistent（持久态）是指该实体对象处于Hibernate框架所管理的状态，也就是说这个实体对象与session对象的实例相关。处于持久太的实体对象最大的特征是对其所做的任何变更操作都将被Hibernate持久化到数据库层。持久状态:持久化对象的实例在数据库中有对应的记录，并拥有一个持久化表示(ID)。对 持久化对象进行 delete 操作后，数据库中对应的记录将被删除，那么持久化对象与数据 库记录不再存在对应关系，持久化对象变成临时状态。 持久化对象被修改变更后，不会马上同步到数据库，直到数据库事务提交。在同步之前，持 久化对象是脏的(Dirty)。

3.Detached(游离态)是指处于持久态的对象，当不再与他所对应的Session对象向关联时，这个对象就变成游离态了。游离状态:当 Session 进行了 Close、Clear 或者 evict 后，持久化对象虽然拥有持 久化标识符和与数据库对应记录一致的值，但是因为会话已经消失，对象不在持久化管理之 内，所以处于游离状态(也 叫:脱管状态)。游离状态的对象与临时状态对象是十

分相似 的，只是它还含有持久化标识。

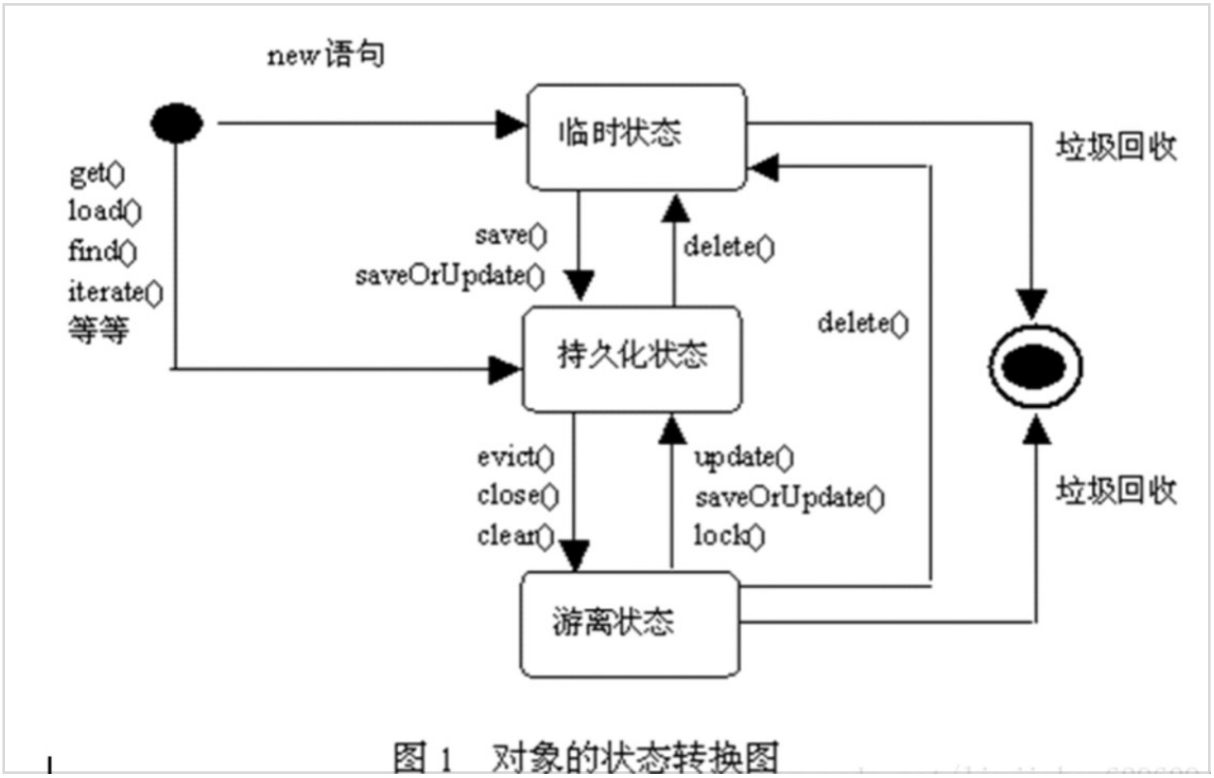

图 1　对象的状态转换图

hibernate缓存

Hibernate的一级缓存是由Session提供的，默认只支持一级缓存，因此它只存在于Session的生命周期中，当程序调用 save(),update(),saveOrUpdate()等方法 及调用查询接口 list,filter,iterate时，如Session缓存中还不存 在相应的对象，Hibernate会把该对象加入到一级缓存中，当Session关闭的时候缓存也会消失。

Hibernate的一级缓存是Session所内置的，不能被卸载，也不能进行任何配置一级缓存采用的是key-value 的Map方式来实现的，在缓存实体对象时，对象的主关键字ID是Map的key，实体对象就是对应的值。

Hibernate二级缓存:把获得的所有数据对象根据ID放入到第二级缓存中。Hibernate二级缓存策略，是针 对于ID查询的缓存策略，删除、更新、增加数据的时候，同时更新缓存。 二级缓存独立于session，默认不开启;

Hibernate实体之间的关联关系的三种形式

一对一关联

一对多关联

多对多关联

Hibernate具用的三种检索方式

HQL 检索方式： HQL 是面向对象的查询语言，它可以查询以对象形式存在的数据。 和sql 语句还是非常类似的，就是将sql语句中的表改成实体名，字段改为属性名即可，其它 基本都很相似的。主要用的是Query这个对象。返回值可以使用list，iterate(多 值)，uniqueResult(单值)等属性进行取值。

QBC 检索方式； 是通过利用Criteria对象来进行查询 的，更适合做不定参数查询的情况，

SQL 检索方式

级联cascade属性

!2019SE-NOTE/hibernate_cascade描述.png at master · gogowhy/2019SE-NOTE · GitHub

Cascade 属性的作用是描述关联对象进行操作时的级联特性，因此只有涉及到关系的元素才有 cascade 属性

inverse属性

"Inverse"-直译过来就是"反转，使颠倒"的意思，书面化的解释为"是否将关系维护的权力交给对方"

Hibernate中的"inverse"属性只有两个值"true"和"false"。"true"表示将关系维护的权力交给对方，"false"表示不交出维护权力（默认值）。

inverse 属性只存在于集合标记的元素中集合元素包括 <set>,<*map*>,<list>,<*array*>,<bag/>

Inverse 属性的作用是是否将对集合对象的修改反射到数据库中 inverse="**false**"时 修改反射到数据库中 inverse="**true**" 时修改不反射到数据库中

如果不设置inverse，两个都会操作，不会出问题，但是影响性能

延迟加载（lazy懒加载）

Hibernate从数据库获取某一个对象数据、获取某一个对象的集合属性时，获取某一个对象所关联的另一个对象时，由于没有使用该对象的数据，hibernate并不从数据库加载真正的数据。而是为该对象创建一个代理对象来代表这个对象，这个对象上的所有属性都为默认值，只有在真正的需要该对象的数据时才创建这个真实的对象，真正的从数据库中加载数据。

因为在类加载和集合加载的时候都会遇到懒加载的问题，当我们在查询的时候不需要懒

加载，因为通过懒加载不能查到是情况下，可以在class类标签或者是set集合的标签中设置lazy=false，表示立即加载，而true表示懒加载

hibernate优化

1.使用双向一对多关联，不使用单向一对多

2.不用一对一，用多对一取代

3.配置对象缓存，不使用集合缓存

Integer与Int

Integer是对象. code = null; 对象可以为空.

Int 是普通类型, 不可能 = null. 根据你的数据库code是可以空的, 故应该映射成Integer. 你没理由hbm.xml里写Integer, 类里却写int

锁机制

1，悲观锁:对于一些数据我们是不能同时去修改的，否则就会出现数据的 错误，在数据库中我们可以通过行级锁select……for update进行对数据的锁定，避 免的。而在框架中就是通过悲观锁的。为什么叫它悲观锁呢?因为发生这种同时修改数据的几率是非常非常小，而此种锁却一直加上了，所以它是一种悲观者的身份来做 的事。所以称为悲观锁。看一下怎么使用吧!设置LockMode.UPGRADE参数，那么只有当前事务提交后，另外的事务 才能够查询这个数据。这种悲观锁的性能比较底。

```
Account account =(Account)session.get(Account.class, 1 ,
LockMode.UPGRADE);
```

2，乐观锁:其实就是以乐观人的态度来解决这种数据同时修改的问题。解 决原理是，当事务不同时发生时，没有锁，如果事务同步发生了，它的锁就起作用 了。所以来说，它只是在这种几率很小的情况发生时才会加锁，所以叫乐观锁。它的 性能大大的提高了。

A，实现方式:时间戳和版本号(Hibernate框架实现的)
b，在我们需要加乐观锁对应的数据上，添加ptimistic- lock=”version”属性:

5.Session

session与cookie

session在服务器端，cookie在客户端
Cookie的安全性一般，他人可通过分析存放在本地的Cookie并进行Cookie欺骗。在安全性第一的前提下，选择Session更优。重要交互信息比如权限等就要放在Session中，一般的信息记

录放Cookie就好了。

3、单个Cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个Cookie。

4、Session 可以放在 文件、数据库或内存中，比如在使用Node时将Session保存在redis中。由于一定时间内它是保存在服务器上的，当访问增多时，会较大地占用服务器的性能。考虑到减轻服务器性能方面，应当适时使用Cookie。

5、Session 的运行依赖Session ID，而 Session ID 是存在 Cookie 中的，也就是说，如果浏览器禁用了 Cookie，Session 也会失效（但是可以通过其它方式实现，比如在 url 中传递 Session ID）。

6、用户验证这种场合一般会用 Session。因此，维持一个会话的核心就是客户端的唯一标识，即Session ID

session的方法

Save()方法
1：save方法使临时对象–––>变成持久化对象。
2：为对象分配ID。
3：flush缓存时发送insert语句。

persist()：
和save一样。只有一个区别：在persist()方法前设置ID会报错。如果对象有ID。不能执行insert。而是会抛出异常

3：get()和load()方法的区别：
1）： get会立即加载对象。返回的是类本身。load不会。只是去使用的时候才会执行查询语句。返回的是一个代理对象。代理对象就是别人让你做一件事。先答应下来。然后他需要的时候去帮他做。get立即检索。load延迟检索。

1记录不存在时 get()方法会返回空(**null**)，而load()方法将会抛出一个 HibernateException 异常

2 load()方法查询数据时会先找 Hibernate 的内部缓存和二级缓 存中的现有数据，get()方法在内部缓存中没有打到相对应的数据时装直接执行 SQL 语句 进行查询

session什么时候是游离态

session.close();方法后

怎样构建SessionFactory

Hibernate 的 SessionFactory 接口提供 Session 类的实例，Session 类用于完成 对数据库的操作。由于 SessionFactory 实例是线程安全的(而 Session 实例不是线程安全的)，所以每个 操作都可以共用同一个 SessionFactory 来获取 Session。Hibernate 配置文件分为两种 格式，一种是 xml 格式的配置文件，另一种是 Java 属性文件格式的配置文件

28.:从XML文件读取配置信息构建SessionFactory的具体步骤如下。
(1)创建一个 Configuration 对象，并通过该对象的 configura()方法加载 Hibernate 配置文件

(2)完成配置文件和映射文件的加载后，将得到一个包括所有 Hibernate 运行期参数的 Configuration 实例，通过 Configuration 实例buildSessionFactory()方法可以构 建一个惟一的 SessionFactory

6.ORM

在介绍Hibernate的时候，说了**Hibernate是一种ORM的框架**。那什么是ORM呢？**ORM是一种思想**

- O代表的是Objcet
- R代表的是Relative
- M代表的是Mapping

ORM->对象关系映射....ORM关注是**对象与数据库中的列的关系**

HQL查询参数的处理方法

直接将参数拼写为 HQL 语句
通过参数名称来标识参数

HQL 是面向对象的查询语言，它可以查询以对象形式存在的数据。

一对一、多对一等关系

1、一对多：比如说一个班级有很多学生，可是这个班级只有一个班主任。在这个班级中随便找一个人，就会知道他们的班主任是谁；知道了这个班主任就会知道有哪几个学生。这里班主任和学生的关系就是一对多。

2、多对一：比如说一个班级有很多学生，可是这个班级只有一个班主任。在这个班级中随便找一个人，就会知道他们的班主任是谁；知道了这个班主任就会知道有哪几个学生。这里学生和班主任的关系就是多对一。

3、一对一：比如说一个班级有很多学生，他们分别有不同的学号。一个学生对应一个学号，一个学号对应一个学生；通过学号能找到学生，通过学生也能得到学号，不会重复。这里学生和学号的关系就是一对一。

4、多对多：比如说一个班级有很多学生，他们有语文课、数学课、英语课等很多课。一门课有很多人上，一个人上很多门课。这里学生和课程的关系就是多对多。

1、一对一的使用方法
（1）使用嵌套结果映射来处理重复的联合结果的子集 。
（2）通过执行另外一个SQL映射语句来返回预期的复杂类型。

2、一对多的实现方法
在多的一方的表里面，添加外键。

3、多对多的实现方法
多对多，必须要通过单独的一张表来表示。

7.spring容器

定义什么是spring

Spring 是一个集成了许多第三方框架的大杂烩，其核心技术是 IOC (控制反转，也称 依赖注入)和 AOP(面向切面编程)

MVC的dao层、service层和controller层

1、dao层
Dao层主要做数据持久层的工作，负责与数据库进行联络的一些任务都封装在此，dao层的设计首先是设计dao层的接口，然后在Spring的配置文件中定义此接口的实现类，然后就可以再模块中调用此接口来进行数据业务的处理，而不用关心此接口的具体实现类是哪个类，显得结构非常清晰，dao层的数据源配置，以及有关数据库连接参数都在Spring配置文件中进行配置。

2、service层
Service层主要负责业务模块的应用逻辑应用设计。同样是首先设计接口，再设计其实现类，接着再Spring的配置文件中配置其实现的关联。这样我们就可以在应用中调用service接口来进行业务处理。service层的业务实，具体要调用已经定义的dao层接口，封装service层业务逻辑有利于通用的业务逻辑的独立性和重复利用性。程序显得非常简洁。

3、controller层

Controller层负责具体的业务模块流程的控制，在此层要调用service层的接口来控制业务流程，控制的配置也同样是在Spring的配置文件里进行，针对具体的业务流程，会有不同的控制器。我们具体的设计过程可以将流程进行抽象归纳，设计出可以重复利用的子单元流程模块。这样不仅使程序结构变得清晰，也大大减少了代码量。

4、view层

View层与控制层结合比较紧密，需要二者结合起来协同开发。view层主要负责前台jsp页面的显示。

5、它们之间的关系：

Service层是建立在DAO层之上的，建立了DAO层后才可以建立Service层，而Service层又是在Controller层之下的，因而Service层应该既调用DAO层的接口，又要提供接口给Controller层的类来进行调用，它刚好处于一个中间层的位置。每个模型都有一个Service接口，每个接口分别封装各自的业务处理方法。

什么是容器

容器就是符合某种规范的能够提供一系列服务的管理器。

spring实现了什么模式

工厂模式 和 单例模式

Spring 用到了哪些设计模式

spring源码也读了两遍了，但对于里面描述的关系还是不太清楚，以后再多读几遍吧，但总还有些收获的，下面就说说spring用到的设计模式主要有哪些：

1.工厂模式，这个很明显，在各种BeanFactory以及ApplicationContext创建中都用到了；

2.模版模式，这个也很明显，在各种BeanFactory以及ApplicationContext实现中也都用到了；

3.代理模式，在Aop实现中用到了JDK的动态代理；

4.策略模式，第一个地方，加载资源文件的方式，使用了不同的方法，比如：ClassPathResourece，FileSystemResource，ServletContextResource，UrlResource但他们都有共同的借口Resource；第二个地方就是在Aop的实现中，采用了两种不同的方式，JDK动态代理和CGLIB代理；

5.单例模式，这个比如在创建bean的时候。

依赖注入的三种方式

分别是属性(Setter方法)注入，构造注入和接口注入。

在Spring中，那些组成应用的主体及由Spring IOC容器所管理的对象被称之为Bean。 Spring的IOC容器通过反射的机制实例化Bean并建立Bean之间的依赖关系。 简单地讲，Bean就是由Spring IOC容器初始化、装配及被管理的对象。 获取Bean对象的过程，首先通过Resource加载配置文件并启动IOC容器，然后通过getBean方法获取 bean对象，就可以调用他的方法。

依赖注入的好处

程序可扩展性更强;利于并行开发;

什么是IOC控制反转

不创建对象，但是描述创建它们的方式。在代码中不直接与对象和服务连接，但在配置文件中描述哪一个组件需要哪一项服务。容器(在 Spring 框架中是 IOC 容器) 负责将这些 联系在一起。 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，控制权由应用代 码中转到了外部容器，控制权的转移，就是所谓的反转。

(Ioc)模式将应用程 序的配置和依赖性规范与实际的应用代码程序分开

依赖注入和控制反转是对同一件事情的不同描述，从某个方面讲，就是它们描述的角度不同。依赖注入是从 应用程序的角度在描述，可以把依赖注入描述完整点:应用程序依赖容器创建并注入它所需要的外部资源;而控制反 转是从容器的角度在描述，描述完整点:容器控制应用程序，由容器反向的向应用程序注入应用程序所需要的外部资 源。
其实IoC*DI*对编程带来的最大改变不是从代码上，而是从思想上，发生了"主从换位"的变化。应用程序 原本是老大，要获取什么资源都是主动出击，但是在*IoC*DI思想中，应用程序就变成被动的了，被动的等待IoC/DI 容器来创建并注入它所需要的资源了。
这么小小的一个改变其实是编程思想的一个大进步，这样就有效的分离了对象和它所需要的外部资源，使 得它们松散耦合，有利于功能复用，更重要的是使得程序的整个体系结构变得非常灵活

ODBC与JDBC

ODBC（Open Database Connectivity）odbc对数据库进行了封装，对应用程序提供一致的接口。odbc是驱动管理器，针对不同的数据库产品，有对 应的驱动程序。可以这样认为，针对不同的数据库产品，需要相应的适配器，这个适配器就是驱动，而 odbc就 是适配器的管理器。odbc中管理了一组适配器，添加数据源的时候，需要指定Ip地址，数据库，登录名，密码，以及适配器，为这个数据源取个名称。

是一组对数据库访问的标准API，这些API通过SQL来完成大部分任务，而且它本身也支持SQL语言，支持用户发来的SQL。ODBC定义了访问数据库API的一组规范，这些API独立于形色各异的DBMS和编程语言。 也就是说，一个基于ODBC的应用程序，对数据库的操作不依赖任何DBMS，不直接与DBMS打交道，所有的数据库操作由对应的DBMS的ODBC驱动程序完成。不论是SQL Server、Access还是Oracle数据库，均可用ODBC API进行访问。由此可见，ODBC的最大优点是能以统一的方式处理所有的数据库。

JDBC（JavaDatabase Connectivity）是Java与数据库的接口规范，JDBC定义了一个支持标准SQL功能的通用低层API，它由Java 语言编写的类和接口组成，旨在让各数据库开发商为Java程序员提供标准的数据库API。JDBC API定义了若干Java中的类，表示数据库连接、SQL指令、结果集、数据库元数据等。它允许Java程序员发送SQL指令并处理结果。

相同点：*JDBC*与*ODBC*都是基于*X*Open的SQL调用级接口；从结构上来讲，JDBC的总体结构类似于ODBC,都/有四个组件：应用程序、驱动程序管理器、驱动程序和数据源，工作原理亦大体相同；
在内容交互方面，JDBC保持了ODBC的基本特性,也独立于特定数据库. 而且都不是/直接与数据库交互，而是通过驱动程序管理器。

区别：我们知道，ODBC几乎能在所有平台上连接几乎所有的数据库。为什么 Java 不使用 ODBC?

答案是：Java 可以使用 ODBC，但最好是以JDBC-ODBC桥的形式使用（Java连接总体分为Java直连和JDBC-ODBC桥两种形式）。

那为什么还需要 JDBC？因为ODBC 不适合直接在 Java 中使用，因为它使用 C 语言接口。从Java 调用本地 C代码在安全性、实现、坚固性和程序的自动移植性方面都有许多缺点。从 ODBC C API 到 Java API 的字面翻译是不可取的。例如，Java 没有指针，而 ODBC 却对指针用得很广泛（包括很容易出错的指针"void *"）。另外，ODBC 比较复杂，而JDBC 尽量保证简单功能的简便性，同时在必要时允许使用高级功能。如果使用ODBC，就必须手动地将 ODBC 驱动程序管理器和驱动程序安装在每台客户机上。如果完全用 Java 编写 JDBC 驱动程序则 JDBC代码在所有 Java 平台上（从网络计算机到大型机）都可以自 动安装、移植并保证安全性。

总之，JDBC 在很大程度上是借鉴了ODBC的，从他的基础上发展而来。JDBC 保留了 ODBC 的基本设计特征，因此，熟悉 ODBC 的程序员将发现 JDBC 很容易使用。它们之间最大的区别在于：JDBC 以 Java 风格与优点为基础并进行优化，因此更加易于使用。

@annotation注释

@Override

它是伪代码,表示重写(当然不写也可以)，不过用它有以下好处:
1、可以当注释用,方便阅读；

2、编译器可以给你验证@Override下面的方法名是否是你父类中所有的，如果没有则报错。例如，你如果没写@Override，而你下面的方法名又写错了，这时你的编译器是可以编译通过

的，因为编译器以为这个方法是你的子类中自己增加的方法。

@Component

它表示一个通用注释用于说明一个类是一个spring容器管理的类，即该类已经拉入到spring的管理中了。（把普通pojo实例化到spring容器中，相当于配置文件中的<bean id="" class=""/>）

@Service

@Service用于标注业务层组件_

@Controller

它是对@Component的细化的类。（注入服务）,用于标注控制层组件(如struts中的action)_

@Repository

用于标注数据访问组件，即DAO组件.它是对@Component的细化，用来给持久层的类定义一个名字，让Spring根据这个名字关联到这个类。（实现dao访问）

@Autowired

@Autowired为Spring提供的注解，只按照byType注入。它可以对类成员变量、方法及构造函数进行标注，完成自动装配的工作。 通过 @Autowired的使用来消除 set， get方法。

这个注解就是*spring*可以自动帮你把*bean*里面引用的对象的*setter*getter方法省略，它会自动帮你set*get*。通过@Autowired自动装配方式，从IoC容器中去查找到，并返回给该属性

在使用@Autowired时，首先在容器中查询对应类型的bean

如果查询结果刚好为一个，就将该bean装配给@Autowired指定的数据

如果查询的结果不止一个，那么@Autowired会根据名称来查找。

如果查询的结果为空，那么会抛出异常。解决方法时，使用required=false

@Resource

只是@AutoWried按by type自动注入，而@Resource默认按byName自动注入。@Resource有两个重要属性，分别是name和type

spring将name属性解析为bean的名字，而type属性则被解析为bean的类型。所以如果使用

name属性，则使用byName的自动注入策略，如果使用type属性则使用byType的自动注入策略。如果都没有指定，则通过反射机制使用byName自动注入策略。

@Resource依赖注入时查找bean的规则：(以用在field上为例)

annotation与xml

xml:

优点:使代码通俗易懂;扩展性好;有成熟的验证机制确保正确性; 配置与代码 分离 缺点:需要解析工具的支持;解析xml影响性能;配置文件过多导致管理困难;只 能在运行时查错;IDE无法验证配置正确性;查错困难;同时维护代码和配置，效 率低下;配置项与代码间可能因为一方修改影响到另一方

annotation: 优点:

保存在class文件中，降低维护成本;无需工具支持与解析;编译期间即可查错且 较容易查错;提升开发效率缺点:对配置修改不得不修改Java文件，重新编译打包应用;可扩展性差

Spring 框架的 7 个模块

(1) spring AOP –面象切面编程

AOP 把软件系统分为两个部分:核心关注点和横切关注点。所谓的核心关注点，是业务处理的主要流程，也就是说这个解决方案要做的事。所谓横切关注点，是与核心业务无关 的部分，它把常发生在核心关注点的多处，而各处基本相似，如日志，事务，权限等 。

应用场景：

性能检测，访问控制，日志管理，事务等。 默认的策略是如果目标类实现接口，则使用JDK动态代理技术，如果目标对象没有实现接口，则默认会采用 CGLIB代理

(2)spring DAO –数据访问对象

(3)spring ORM –对象关系影射

(4)springContect –上下文配置，向Spring框架提供上下文信息

(5)springWEB –WEB上下文模块

(6)\springWEB-MVC –实现了MVC

(7)spring CORE –核心容器提供 Spring 框架基本功能

Spring 赋值方式

(1)普通属性赋值 (2)集合属性赋值 (3)Properties 赋值 (4)Map 属性赋值

Spring mvc的运行原理

1. 客户端请求提交到DispatcherServlet
2. 由DispatcherServlet控制器查询HandlerMapping，找到并分发到指定的Controller中。
3. Controller调用业务逻辑处理后，返回ModelAndView
4. DispatcherServlet查询一个或多个ViewResoler视图解析器，找到ModelAndView指定的视图
5. 视图负责将结果显示到客户端

Spring事务配置方法:

1.切点信息，用于定位实施事物切面的业务类方法

2.控制事务行为的事务属性，这些属性包括事物隔离级别，事务传播行为，超时时间，回滚规则。

8.JSP、Servlet

四种会话追踪技术

cookie

Cookie是Web服务器发送给客户端的一小段信息，客户端请求时可以读取该信息发送到服务器端，进而进行用户的识别。对于客户端的每次请求，服务器都会将Cookie发送到客户端,在客户端可以进行保存,以便下次使用。
客户端可以采用两种方式来保存这个Cookie对象，一种方式是保存在客户端内存中，称为临时Cookie，浏览器关闭后 这个Cookie对象将消失。另外一种方式是保存在 客户机的磁盘上，称为永久Cookie。以后客户端只要访问该网站，就会将这个Cookie再次发送到服务器上，前提是 这个Cookie在有效期内。这样就实现了对客户的跟踪。
Cookie是可以被禁止的。

url 重写

URL(统一资源定位符)是Web上特定页面的地址，URL重写的技术就是在URL结尾添加一个附加数据以标识该会话,把会话ID通过URL的信息传递过去，以便在服务器端进行识别不同的用户 。

session

每一个用户都有一个不同的session，各个用户之间是不能共享的，是每个用户所独享的，在session中可以存放信息。在服务器端会创建一个session对象，产生一个sessionID来标识这个session对象，然后将这个sessionID放入到Cookie中发送到客户端，下一次访问时，sessionID会发送到服务器，在服务器端进行识别不同的用户。Session是依赖Cookie的，如果Cookie被禁用，那么session也将失效。因为Session是用Session ID来确定当前对话所对应的服务器Session，而Session ID是通过Cookie来传递的，禁用Cookie相当于失去了Session ID，也就得不到Session了。此时可以考虑URL重写和表单隐藏域。

隐藏表单域

将会话ID添加到HTML表单元素中提交到服务器,此表单元素并不在客户端显示 。

filter

Filter也称之为过滤器，它是Servlet技术中最激动人心的技术，WEB开发人员通过Filter技术，对web服务器管理的所有web资源：例如Jsp, Servlet, 静态图片文件或静态 html 文件等进行拦截，从而实现一些特殊的功能。例如实现URL级别的权限访问控制、过滤敏感词汇、压缩响应信息等一些高级功能。

Servlet API中提供了一个Filter接口，开发web应用时，如果编写的Java类实现了这个接口，则把这个java类称之为过滤器Filter。通过Filter技术，开发1.Filter的创建

Filter的创建和销毁由**WEB服务器负责**。 web 应用程序**启动时**，web 服务器将**创建Filter的实例对象**，并调用其init方法，完成对象的初始化功能，从而为后续的用户请求作好拦截的准备工作，**filter对象只会创建一次，init方法也只会执行一次**。通过init方法的参数，可获得代表当前filter配置信息的FilterConfig对象。

2.Filter的销毁

Web容器调用destroy方法销毁Filter。destroy方法在Filter的生命周期中仅执行一次。在destroy方法中，可以释放过滤器使用的资源。

Servlet与Filter的区别

整体的流程是:Filter对用户请求进行预处理，接着将请求交给Servlet进行处理并生成响应，最后Filter再对服务器响应进行后处理。

Filter有如下几个用处: Filter可以进行对特定的url请求和相应做预处理和后处理。 在HttpServletRequest到达Servlet之前，拦截客户的HttpServletRequest。 根据需要检查HttpServletRequest，也可以修改HttpServletRequest头和数据。 在HttpServletResponse到

达客户端之前，拦截HttpServletResponse。 根据需要检查HttpServletResponse，也可以修改HttpServletResponse头和数据。

实际上Filter和Servlet极其相似，区别只是Filter不能直接对用户生成响应。实际上Filter里doFilter()方法里 的代码就是从多个Servlet的service()方法里抽取的通用代码，通过使用Filter可以实现更好的复用。

Filter和Servlet的生命周期：

1.Filter在web服务器启动时初始化 2.如果某个Servlet配置了 1， 该Servlet也是在Tomcat(Servlet容器)启动时初始化。 3.如果Servlet没有配置1， 该Servlet不会在Tomcat启动时初始化，而是在请求到来时初始化。 4.每次请求， Request都会被初始化，响应请求后，请求被销毁。 5.Servlet初始化后，将不会随着请求的结束而注销。 6.关闭Tomcat时，Servlet、Filter依次被注销。

listener

Listener 用于监听java web程序中的事件，例如创建、修改、删除Session、request、context等，并触发响应的事件。

_Listener 对应观察者模式，事件发生的时候会自动触发该事件对应的Listeer。 _Listener 主要用于对_Session、request、context 进行监控。

servlet2.5 规范中共有 8 种Listener _ 。

- HttpSessionListener：监听 Session 的创建与销毁。创建 Session 时执行 sessionCreated（HttpSessionEvent se）方法。超时或者执行 session.invalidate()时执行 sessionDestroyed（HttpSessionEvent se）方法。该 Listener 可用于收集在线者信息。
- ServletContextListener：监听 context 的创建与销毁。context 代表当前的 Web 应用程序。服务器启动或者热部署 war 包时执行 contextInitialized（ServletContextEvent event）方法。服务器关闭时或者只关闭该 Web 时会执行 contextDestroyed（ServletContextEvent event）方法。该 Listener 可用于启动时获取 web.xml 里配置的初始化参数。
- ServletRequestListener：监听 request 的创建与销毁。用户每次请求 request 都会执行 requestInitialized（ServletRequestEvent event）方法。request 处理完毕自动销毁前执行 requestDestroyed（ServletRequestEvent event）方法。注意如果一个 HTML 页面内含有多个图片，则请求一次 HTML 页面可能会触发多次 request 事件。

JSP原理

一个jsp页面第一次被访问时，jsp引擎将其翻译成servlet，这个servlet是一个Java文 件，也是一个完整的Java程序;jsp引擎调用Java编译器对servlet进行编译，得到class 可执行文件;jsp引擎调用Java虚拟机来解释执行class 生成向客户端的应答，发送给客 户端

JSP的缺点

9.Java

线程池

线程池的作用

: 在程序启动的时候就创建若干线程来响应处理，它们被称为线程池，里面的线程叫工作线 程 第一:降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。 第二:提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。 第三:提高线程的可管理性。

类加载器

类加载器是负责将可能是网络上、也可能是磁盘上的class文件加载到内存中。并为其生成对应的java.lang.class对象。一旦一个类被载入JVM了，同一个类就不会被再次加载。那么怎样才算是同一个类？在JAVA中一个类用其全限定类名（包名和类名）作为其唯一标识，但是在JVM中，一个类用其全限定类名和其类加载器作为其唯一标识。也就是说，在JAVA中的同一个类，如果用不同的类加载器加载，则生成的class对象认为是不同的。

当JVM启动时，会形成由三个类加载器组成的初始类加载器层次结构

1、启动类加载器BootstrapClassLoader:

是嵌在JVM内核中的加载器，该加载器是用C++语言写的，主要负载加载JAVA_HOME/lib下的类库，启动类加载器无法被应用程序直接使用。

2、扩展类加载器Extension ClassLoader：

该加载器器是用JAVA编写，且它的父类加载器是Bootstrap，是由sun.misc.Launcher$ExtClassLoader实现的，主要加载JAVA_HOME*lib*ext目录中的类库。开

发者可以这几使用扩展类加载器。

10.mongodb

mongodb的查询只返回数据库的游标，只有你在真正要用它的时候才会取出数据mongodb支持自动分表 在本机存储耗尽，想要读取数据更快，或者保持一大部分数据在内存中时，要考虑分表。 分表的操作:选择一个key来分离数据。mongodb会自动重平衡数据分布。

Webpack

WebPack可以看做是**模块打包机**：它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其打包为合适的格式以供浏览器使用。

他会分析模块间的依赖关系，然后使用loaders处理它们，最后生成一个优化并且 合并后的静态资源。

为什么要用

今的很多网页其实可以看做是功能丰富的应用，它们拥有着复杂的JavaScript代码和一大堆依赖包。为了简化开发的复杂度，前端社区涌现出了很多好的实践方法

A:模块化，让我们可以把复杂的程序细化为小的文件;
B:类似于TypeScript这种在JavaScript基础上拓展的开发语言：使我们能够实现目前版本的JavaScript不能直接使用的特性，并且之后还能能装换为JavaScript文件使浏览器可以识别；
c:scss，less等CSS预处理器
这些改进确实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外的处理才能让浏览器识别,而手动处理又是非常繁琐的，这就为WebPack类的工具的出现提供了需求。