**Hibernate是什么**

面向java环境的对象/关系数据库映射工具。

1.开源的持久层框架.

2.ORM(Object/Relational Mapping)映射工具，建立面向对象的域模型和关系数据模型之间的映射.

3.连接java应用和数据库的中间件.

4.对JDBC进行封装，负责java对象的持久化.

5.在分层结构中处于持久化层，封装对数据库的访问细节，

使业务逻辑层更专注于实现业务逻辑

**为什么要用Hibernate**

1、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。   
 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，它很大程度的简化了dao层编码工作。   
 3、Hibernate使用java的反射机制，而不是字节码增强程序类实现

透明性   
 4、Hibernate的性能非常好，因为它是一个轻量级框架。映射的灵

活性很出色。它支持很多关系型数据库，从一对一到多对多的各

种复杂关系。

**拦截器和过滤器的区别：**

\* 相同点：都是起拦截的作用

\* 不同点：

\* 过滤器是J2EE的标准，任何的一个程序都可以使用

\* 拦截器是struts2框架的标准，不能离开struts2框架，必须依赖struts2框架

\* 过滤器拦截的是web.xml配置文件

\* 其他的拦截工作就都交给了拦截器

\* 执行的顺序：过滤器 ---> 拦截器

## 多线程

**创建线程的四种方法**

继承thread类

class MyThread extends Thread{

public void run() {

}

}

实现runnable接口

class MyTask implements Runnable{

public void run(){

}

}

使用匿名内部类

new Thread(){

public void run(){

...

}

}

使用lambda表达式（java8以上）

new Thread(()->{...}).start();

**多线程同步**

1. 同时运行的线程需要共享数据就必须考虑其他线程的状态与行为，这时就需要实现同步。
2. Java引入了对象互斥锁的概念，来保证共享数据操作的完整性关键字synchronized用来与对象的互斥锁联系（可以保证代码和方法的互斥性和原子性）。

. synchronized{\*\*\*代码块\*\*\*}

. synchronized A ()//方法

1. 使用wait（）方法可以释放对象锁
2. 使用notify（）或者notifyAll（）可以让等待的一个或所有线程进入就绪状态

生产者，消费者模式

class CubbyHole{

private int index=0;

private int [] data=new int[3];

public synchronized void put(int value){

while(index==data.length){

try{

this.wait()

}catch(e){}

}

data[index]=value;

index++

this.notify();

}

public synchronized void get(){

while(index<=0){

try{

this.wait()

}catch(e){}

}

index--;

int val=data[index];

this.notify();

return val;

}

}

**线程启动时的start（）和run（）方法的区别**

Runnable接口只有一个方法，就是run()，当通过实现Runnable接口来创建线程时，启动线程会使得run()方法在那个独立执行的线程中被调用。

start()方法是Thread的类方法，start()方法会使得该线程开始执行；java虚拟机会去调用该线程的run()方法。

用start()来启动线程，实现了真正意义上的启动线程，此时会出现**异步执行**的效果，即在[线程的创建和启动](http://blog.csdn.net/gxx_csdn/article/details/78980380)中所述的随机性。‘   
而如果使用run()来启动线程，就不是**异步执行**了，而是同步执行，实则是由main方法调用run，中规中矩，运行结果确定，不会达到使用线程的意义。

## IO流

**输入输出流**

**InputStream类**

Read() public int read(); 读入一个字节，-1表示无

Public int read(byte b[]); 返回读入的字节数

Public int read(byte b[],int off,int len);

**OutputStream类**

Write() public void write(int b);将参数b的低位字节写入到输出流

public void write(byte b[]);将字节数组b[]中的全部字节顺序写人到输出流

public void write(byte b[],int off,int len);将字节数组b[]中从off开始的len个字节

flush() public void flush();刷新缓存，实际写入到文件，网络

close() public void close();关闭流

**Reader类**

**Writer类**

处理流的构造方法总是要带一个其他的流对象作为参数，一个流对象经过其他流的多次包装，称为刘的链接。

BufferReader in=new BuffferReader(

new InputReamReader(

new FileputStream(file),”utf-8”));

s=in.reaLin();

**读取网页文件内容的步骤如下**

1. 创建一个url类型的对象。URL url=new URL(“http//:www.datang.com”);
2. 利用URL类的openstream（），获得对应的Inputstream类的对象。InputStream stream=url.openStream();
3. 通过inputstream或inputstreamReader来读取内容。

## Xml解析

**XML 解析：** 其实就是获取元素里面的字符数据或者属性数据。

XML解析方式(面试常问)

有很多种，但是常用的有两种。

\* DOM：会解析整个xml文档到内存，形成一个树形结构，整个结构成为一个对象（document）。如果xml特别大，那么将会造成内存溢出。可以对文档进行增删。

\* SAX：基于事件驱动，读一行，解析一行。不会造成内存溢出，但不能进行增删。

针对这两种解析方式的API，一些组织或者公司， 针对以上两种解析方式， 给出的解决方案有哪些？

jaxp sun公司。 比较繁琐

jdom

dom4j 使用比较广泛

###Dom4j 基本用法

1. 创建SaxReader对象

2. 指定解析的xml

3. 获取根元素。

4. 根据根元素获取子元素或者下面的子孙元素

try {

//1. 创建sax读取对象

SAXReader reader = new SAXReader(); //jdbc -- classloader

//2. 指定解析的xml源

Document document = reader.read(new File("src/xml/stus.xml"));

//3. 得到元素、

//得到根元素

Element rootElement= document.getRootElement();

//获取根元素下面的子元素 age

//rootElement.element("age")

//System.out.println(rootElement.element("stu").element("age").getText());

//获取根元素下面的所有子元素 。 stu元素

List<Element> elements = rootElement.elements();

//遍历所有的stu元素

for (Element element : elements) {

//获取stu元素下面的name元素

String name = element.element("name").getText();

String age = element.element("age").getText();

String address = element.element("address").getText();

System.out.println("name="+name+"==age+"+age+"==address="+address);

}

} catch (Exception e) {

e.printStackTrace();

}

###Dom4j dom4j里面支持Xpath的写法。 xpath其实是xml的路径语言，支持我们在解析xml的时候，能够快速的定位到具体的某一个元素。

1. 添加jar包依赖 jaxen-1.1-beta-6.jar

2. 在查找指定节点的时候，根据XPath语法规则来查找

3. 后续的代码与以前的解析代码一样。

# Servlet

**Get 和 Post请求区别**

\* post

1. 数据是以流的方式写过去，不会在地址栏上面显示。 现在一般提交数据到服务器使用的都是POST

2. 以流的方式写数据，所以数据没有大小限制。

3.request.setcharacterencoding(“”);response.setcontenttype(“text/html;charset=””)解决post的中文乱码问题

\* get

1. 会在地址栏后面拼接数据，所以有安全隐患。 一般从服务器获取数据，并且客户端也不用提交上面数据的时候，可以使用GET

2. 能够带的数据有限， 1kb大小

3.配置tomcat server.xml URIEncoding=””或者new string(xxx.getbytes(ios-8859-1”),”utf-8”)；解决get的中文乱码问题。

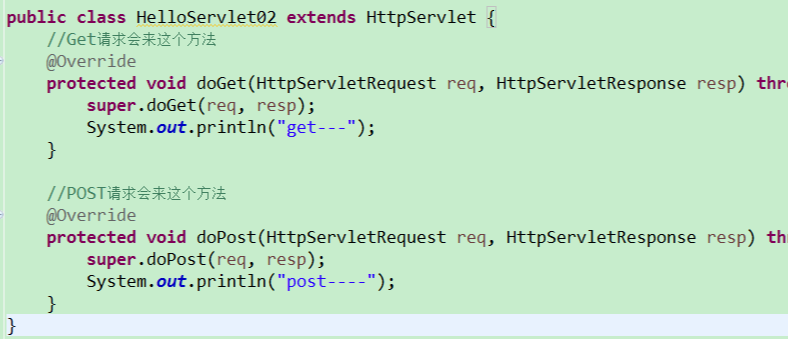
**servlet是什么?**

> 其实就是一个java程序，运行在我们的web服务器上，用于接收和响应客户端的http请求。

> 更多的是配合动态资源来做。 当然静态资源也需要使用到servlet，只不过是Tomcat里面已经定义好了一个 DefaultServlet

**Servlet执行过程**

1. 定义一个类，继承HttpServlet（或者是filter）复写doGet 和 doPost



生命周期

\* init（）

在创建该servlet的实例时，就执行该方法。

一个servlet只会初始化一次， init方法只会执行一次

默认情况下是 ： 初次访问该servlet，才会创建实例。

\* service（）

只要客户端来了一个请求，那么就执行这个方法了。

该方法可以被执行很多次。 一次请求，对应一次service方法的调用

\* destroy）

servlet销毁的时候，就会执行该方法

1. 该项目从tomcat的里面移除。

2. 正常关闭tomcat就会执行 shutdown.bat

**Servlet的配置中<url-pattean>的三种配置**

1. 完全路径匹配，/a/b（不包含\*和.）
2. 目录匹配，/a/\*（包含\*和/）
3. 后缀名匹配，\*a（绝对不能有/）

三种匹配模式的优先级为1》2》3

**ServletContxt的应用**

1. 获取web应用（全局）的初始化参数
2. 通过servletcontext对象实现数据共享
3. 实现servlet的转发
4. 利用servletcontext对象读取资源文件

ServletConfig当前servlet的参数初始化

ServletContxt全局参数的标签

**转发和重定向的区别？**

**转发**：如果服务器端处理客户端请求时，如果需要多个服务器程序同时进行处理，需要采用转发操作。（不用转回客户端）产生一次请求一次响应；转发对客户端不可见。（context.getrequestDispatcher(path); dispatcher.foword(request,response)）

**重定向**：服务器进行处理后，需要通知客户端访问下一个目标程序，继续处理。产生两次请求，两次响应；对客户端可见。(response(302,location);

**禁止缓存**

在web开发中，存在很多动态web资源，经常需要更新，为了保证客户端第一时间更新，禁止动态web资源缓存。禁止缓存三个头信息

Respose.setHeader(“Cache-Control”,”no-cache”);response.setDateHeader(“Expires”,-1);response.setHeader（“pragma”,”no-Cache”）;

Expires对于一些不经常改变的资源，是比tomcat缓存机制更优的解决方案

**Java语言不仅是一门编程语言，更是也个网络操作系统？**

1. Java语言不依赖于操作系统，它可以运行到计算机硬件或cpu上
2. Java语言可以自己管理内存
3. Java语言支持多线程运行，可对处理器资源进行分配和管理
4. Java可以对硬件设备进行管理，如鼠标，键盘等
5. Java语言支持网络文件管理，他的文件概念已经拓展到整个internet网络

简单的，面向对象的，分布式的，健壮的，安全的，平台无关的，可移植的，解释性的，高性能的，多线程的动态的程序设计语言。

**Filter**

Filter过滤器实现filter接口，并实现init（），dofilter（），destory（）三个方法。要在web.xml中配置过滤器名字和mapping，拦截目标资源后，如果想执行目标资源，则要调用chain.dofilter(res,resp);

Filterconfig和servletconfig类似---为过滤器初始化提供参数

String getInitparameter（name）获得初始化信息

Servletcontext getServletcontext（）得到servletcontext对象，保存全局数据，读取web资源文件

**Java虚拟机**由方法区，堆，java栈，寄存器，指令集组成，这5部分从逻辑上对虚拟机进行划分，java线程分为守护线程和非守护线程。守护线程是应用于后台为其他线程提供服务的线程。

**Java中使用垃圾回收器（GC）**来监视java程序的运行，当对象应用不在使用时，就会自动释放对象所占有的内存。Java用一张对象表来完成软指针和对象引用的映射。Java垃圾收集器的管理线程可以单独运行于后台，并检测每一个对象，通过更改对象的状态，垃圾回收器可以标记，移除和移动对象。

**Array和Arrays的区别（connection和connections）**

Arrays是工具类，提供对数组操作的静态方法，而array是数组。两者分离是因为会经常对数组进行操作，所以就把常用的方法提取出来作为工具，可以提高代码的复用性和低耦合。

for和foreach在遍历数组时的区别

两个都是常用的遍历方法，一般情况下可以通用，但在固定长度或者长度不需要计算的时候for循环的效率高于foreach，在补确定长度或计算长度有性能损耗的时候用foreach比较方便，并且使用foreach的时候会锁定集合中的对象，期间不能修改。

## Sql

创建语句create database a; use a; create table user( ..); 创建数据库；数据表

对**表**的crud操作

select user; show user;select \*from user;显示表结构

alter table user **add** name(列名) varchar(9) unique;为表添加新的列

alter table user **modify** name varchar(9) not null; 修改列信息

alter table user **change** name namenum varchar(9) not null;改变列的名字

alter table user **drop** namenum;删除列

alter table user **rename** to users;改变表的名字

alter table users character **set** utf8;改变表的字符集

对**数据记录**的操作

Insert into users(name,sex,…)values(zs,’nan’,…);

Insert into users values(zs,’nan’…);插入一条记录

Update users set namenum=’4’ where [binary] sex=’nan’;修改记录

Delete from users where sex=’nan’;删除记录

Select name=’zs’,sex=’nv’ from users group by address having height>170;条件查询

Select distinct name from user;排重查询

Select count(name) from user;计数统计查询

**删除数据记录的delete和truncate的区别？**

后者不支持日志回滚，执行删除操作将不可恢复。

**Where和having的区别？**

Where在分组语句执行前执行，不能和group by联合使用；having在分组之后执行，可以与分组函数联合使用，having可以取代where。

**Exists和in的区别？**

in执行时,先把外层挂起,执行子表,子表执行完成后在执行外层表.exists是每次执行的时候执行子表. 那么,如果子表大的话,每次循环在执行,就选择exitsts,子表小的话,用in.

**Union和union all的区别？**

union在进行表求并集后会去掉重复的元素，所以会对所产生的结果集进行排序运算，删除重复的记录再返回结果。union all则只是简单地将两个结果集合并后就返回结果。因此，如果返回的两个结果集中有重复的数据，那么返回的结果就会包含重复的数据

## 数据库编程

**流程**

1. 连接数据库
2. 获取connection连接对象
3. 创建statement或者prestatement对象
4. 执行executeQuery（sql），updateQuery（sql），execute（sql）
5. 创建resultset对象
6. 遍历

…

Class,forname(…);

Conn=drivermanager.getconnection(url,user,password)

Statement/prestatem=conn.createstatement;

Boolean hasresultset=stat.execute(sql)

Resultset rs=stat.executeQuery(sql);需要遍历\*\*\*\*\*

**execute、executeUpdate、executeQuery三者的区别（及返回值）**

**boolean** execute(String sql)：允许执行查询语句、更新语句、DDL语句。返回值为true时，表示执行的是查询语句，可以通过getResultSet方法获取结果；返回值为false时，执行的是更新语句或DDL语句，getUpdateCount方法获取更新的记录数量。

**int** executeUpdate(String sql)：执行给定 SQL 语句，该语句可能为 INSERT、UPDATE 或 DELETE 语句，或者不返回任何内容的 SQL 语句（如 SQL DDL 语句）。返回值是更新的记录数量

**ResultSet** executeQuery(String sql)：执行给定的 SQL 语句，该语句返回单个 ResultSet 对象。