JDBC提出背景?

1)JDBC解决的是:java连接数据库的问题？

2)程序猿自己实现JAVA链接数据库 XXXX

3)SUN公司实现JAVA链接数据库 XXX

4)数据库厂商实现JAVA链接各自的数据，SUN公司制定一套标准(接口集java.sql.\*)，厂商实现sun公司的这套接口。我们学习标准，使用driver

JDBC流程？

a)加载驱动

Class.forName(“java.lang.String”);

Class是一个描述类的类，其中静态方法forName是根据类的名字加载指定的类

b)获得与数据库连接

Connection con = DriverManager.getConnection(url,user,pwd);

c)创建执行SQL的Statement对象

PreparedStatement pst = con.prepareStatement(sql);

d)执行SQL，有结果返回

ResultSet rs= pst.executeQuery()

e)处理结果 rs.next stu.setXXX(rs.getXXX)

f)关闭资源 close

元数据（MetaData）:关于数据的数据

a)DatabaseMetaData:数据库的元数据 java.sql.\*:

b)ResultSetMetaData:结果集的元数据(重点)

Javax.sql.DataSource(数据源)

a)数据源是用来获得链接的工厂，数据源最核心的作用:获得链接x

b)DriverManager工具的替代者。

c)数据源是java官方提供的一个获得链接标准。从1.4版本开始

d)数据库连接池：是一种可以用来实现数据源的解决方案。

e)Commons-DBCP:是apache-commons提供的采用了连接池方案的javax.sql.DataSource标准的具体实现组件（类库）。

f)DBCP基本用法？

public class TestDBCP{

public static void main(String artgs[])throws Exception{

//1.构造对象

BasicDataSource ds = new BasicDataSource();

//2.给属性赋值 必选项

ds.setDriverClassName("com.mysql.jdbc.Driver");

ds.setUrl("jdbc:mysql:///test");

ds.setUsername("root");

ds.setPassword("etoak");

//可选项

ds.setMaxActive(10);//最大可用连接数

ds.setMaxWait(5000);//最大的等待时间 以毫秒为单位

ds.setMaxIdle(5);//最大空闲链接数

g)数据库连接池的工作原理？

当服务器启动时，初始化一些链接放在连接池中，等待客户端请求，客户端请求链接，则首先判断连接池中是否有可用的链接，如果有则返回，如果没有则，判断当前链接数是否超过了最大可用链接数，如果没有超过，则创建新的链接返回，如果超过则抛出无可用链接的异常，当客户端使用完链接之后，再将链接放回到连接池中，从而实现链接的重用。

h)常用的连接池组件？DRUID/C3P0/DBCP

============================================================================================

JAVA解析EXCEL？（JXL+POI）

使用JXL读写EXCEl文件

读：

//1.获得工作簿对象

Workbook wb = Workbook.getWorkbook(new File("etoak.xls"));

System.out.println(wb);

//2.获得第一个Sheet

Sheet sheet = wb.getSheet(0);

//3.获得列数

int columns = sheet.getColumns();

for(int i=0;i<columns;i++){

//获得某一列中所有的单元格

Cell[] cells = sheet.getColumn(i);

for(Cell cell:cells){

String value = cell.getContents();

System.out.print(value+"\t");

}

写：

//1.创建Workbook对象

WritableWorkbook wwb = Workbook.createWorkbook(new File("emp.xls"));

//2.创建Sheet

WritableSheet sheet = wwb.createSheet("employee",0);

//3.创建单元格

Label id = new Label(0,0,"id");

Label name = new Label(1,0,"name");

Label sal = new Label(2,0,"sal");

Label vid = new Label(0,1,"et001");

Label vname = new Label(1,1,"张三");

Label vsal = new Label(2,1,"1234.45");

//4.组装

sheet.addCell(id);sheet.addCell(name);sheet.addCell(sal);

sheet.addCell(vid);sheet.addCell(vname);sheet.addCell(vsal);

//5.写出到文件中

wwb.write();

wwb.close();

使用POI对EXCEL文件进行解析：

读：

//1.构造workbook

Workbook wb = new XSSFWorkbook(new File("etoak.xlsx"));

//2.找Sheet

Sheet sheet = wb.getSheetAt(0);

//3.获得行范围

int firstRow= sheet.getFirstRowNum();

int lastRow = sheet.getLastRowNum();

//4.遍历每一行

for(int i=firstRow;i<=lastRow;i++){

// row===xssfrow

Row row = sheet.getRow(i);

//获得列范围

int firstCell = row.getFirstCellNum();

int lastCell = row.getLastCellNum();

System.out.println(firstCell+"\t"+lastCell);

for(int k=firstCell;k<lastCell;k++){

//5.获得某一列

Cell cell = row.getCell(k);

if(cell.getCellType()==Cell.CELL\_TYPE\_STRING){

System.out.println(cell.getStringCellValue());

}else{

System.out.println(cell.getNumericCellValue());

}

}

}

写：

//1.构造workbook

Workbook wb = new XSSFWorkbook();

//2.添加Sheet

Sheet sheet = wb.createSheet("employee");

//3.创建行

Row row = sheet.createRow(0);

//4.创建列

row.createCell(0).setCellValue("id");

row.createCell(1).setCellValue("name");

row.createCell(2).setCellValue("sal");

Row row1 = sheet.createRow(1);

//4.创建列

row1.createCell(0).setCellValue("et001");

row1.createCell(1).setCellValue("zhangsan ");

row1.createCell(2).setCellValue(122222);

//5.写出

OutputStream os = new FileOutputStream(new File("emp.xlsx"));

wb.write(os);

wb.close();

POI和JXL的区别？

1)JXL只能解析EXCEL文件，POI可以解析其他office文件

2)JXL只能解析xls文档，POI可以解析xlsx文档

===========================================================================================

一、两种架构：

a)C/S:Client Server QQ 360 大型网络游戏

b)B/S:Browser Server 网站 应用 页游

二、网络中计算机进行通信，需要遵循一定的规范，称之为协议。

三、在JAVAWEB中，客户端和服务器通信遵循的是HTTP协议。

四、HTTP协议是建立在TCP协议基础之上的，基于请求、响应模式的，无状态的应用层协议。

五：请求:从客户端发送到服务器的信息（字符串）

请求行 请求方法 请求资源 协议版本 GET/POST /index.html HTTP/1.1

请求头 Host: Accept: Accept-Language Cookie

消息体 key=value&key1=value2

第六、响应:从服务器回送给浏览的信息(字符串)

状态行 HTTP/1.1 404/500/200 ok

响应头 ....

响应正文<html></html>

GET /index.html HTTP/1.1

Host: www.baidu.com:443

Accept-Encoding: gzip, deflate, sdch, br

Accept-Language: zh-CN,zh;q=0.8

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36

HTTP/1.1 200 OK

Bdpagetype: 1

Bdqid: 0x80129bca0001cbb3

Bduserid: 0

Cache-Control: private

Connection: Keep-Alive

Content-Encoding: gzip

Content-Type: text/html; charset=utf-8

Cxy\_all: baidu+8f0b6ad2cae390c1633d8beb13a38c48

Date: Tue, 05 Dec 2017 03:30:09 GMT

Expires: Tue, 05 Dec 2017 03:29:23 GMT

Server: BWS/1.1

Set-Cookie: BDSVRTM=0; path=/

Set-Cookie: BD\_HOME=0; path=/

Set-Cookie: H\_PS\_PSSID=1427\_24866\_18195\_21078\_17001\_25178\_22158; path=/; domain=.baidu.com

Strict-Transport-Security: max-age=172800

Transfer-Encoding: chunked

Vary: Accept-Encoding

X-Powered-By: HPHP

X-Ua-Compatible: IE=Edge,chrome=1

第七、服务器:能够接收标准的HTTP请求，返回标准的HTTP响应的软件（一堆别人写的代码）

第八、TOMCAT是一款免费的、开源的、纯JAVA实现的WEB服务器。

第九、SERVLET容器:提供了Servlet的运行环境，管理Servlet的生命周期的组件，一般容器存在与服务器中

第十、浏览器:能够发送标准的HTTP请求，接收标准的HTTP相应的软件()

第十一、Servlet核心的API

a)API:javax.servlet.Servlet:

i. 该接口定义了所有的Servlet必须实现的方法。作为Servlet必须实现该接口

ii. 方法列表

1.init(ServletConfig config){setServletConfig()+初始化}

2.service(ServletRequest,ServletResponse)

3.destroy()

4.getServletConfig()

5.getServletInfo()

iii.Servlet是一个运行在WEB服务器端的小的JAVA程序。Servlet接收和响应来自客户端的请求。通常是基于HTTP协议。

iv. 实现该接口，可以继承javax.servlet.GenericServlet或者是继承javax.servlet.http.HttpServlet.

v. 该接口定义了初始化Servlet、服务器请求、从服务器中删除Servlet的方法。这些方法被称作生命周期方法，并且以如下顺序调用:

1. Servlet被构造，然后init()方法初始化

2. 所有来自客户端的请求，都交给service方法处理。

3. 服务器完毕之后，调用destroy方法销毁资源，等待GC回收。

vi. 除了生命周期方法之外，Servlet中提供了一个getServletConfig方法获得ServletConfig对象，从而通过该对象获得一些启动是的信息。

b)API2：javax.servlet.ServletConfig:

i. 每一个Servlet都有一个与之对应的ServletConfig对象。

ii. 容器使用ServletConfig对象传递信息给Servlet在初始化阶段以init()方法参数的形式。

iii. 方法列表:

1. getInitParameter(String name)

2. getInitParameterNames()

3. getServletContext()

4. getServletName()

c)API3:javax.servlet.ServletContext:

i. 该接口定义了一堆与容器通信的方法。简称上下文环境。

ii. getRealPath setAttribute getAttrigute() getContextPath(): /工程名字

iii. ServletContext代表容器多个Servlet对应一个容器

d)API4:javax.servlet.GenericServlet: implements Servlet,ServletConfig

Private ServletConfig config;

1. init(ServletConfig config){this.config = config;init()}

2. abstract service(ServletRequest,ServletResponse)

3. destroy()

4. getServletConfig()

5. getServletInfo()

6. getInitParameter(String name)

7. getInitParameterNames()

8. getServletContext()

9. getServletName()

10. Init(){}

11. Log\*2

e)API5: javax.servlet.http.HttpServlet: extends GenericServlet

1. init(ServletConfig config){this.config = config;init()}

2. public service(ServletRequest req,ServletResponse resp){

HttpServletRequest request = (HttpServletRequest)req;

HttpServletResponse response = (HttpServletResponse)resp;

this.service(request,response);

}

3. Protected service(HttpServletRequest,HttpServletResponse){

String method = Request.getMethod();//获得请求行的第一部分

If(“get”.equals(method)){doGet()}

Esle if(“post”.equals(method)){doPost()}

}

4. doXX(HttpServletRequest,HttpServletResponse){等待你来}

5. destroy()

6. getServletConfig()

7. getServletInfo()

8. getInitParameter(String name)

9. getInitParameterNames()

10. getServletContext()

11. getServletName()

12. Init(){}

13. Log\*2

f)API6:HttpServletRequest/HttpServletResponse

i.request与客户端浏览器发送到服务器的一堆字符串对应

第十二、API 构造者(new) 实现者({..}) 调用者(.)

Servlet 容器 我们 容器

ServletConfig 容器 容器 我们

ServletContext 容器 容器 我们

Http..Request 容器 容器 我们

Http..Response 同上

RequestDispatcher

Cookie

HttpSession

Filter

Listener

第十三、客户端请求一个地址到能够看到信息整个过程？

a)服务器启动阶段:1.解析tomcat自己的配置文件（conf目录），有错则无法启动，没错，则读取webapps下所有的目录中的web.xml文件，如果有问题，容器报错，能过够启动。

b)浏览器地址栏输入地址：http://localhost:8080/xx/servlet/test

c)浏览器将地址打包成标准的HTTP请求，通过网络发送到服务器

d)服务器（Tomcat）接收标准的HTTP请求字符串，然后创建HttpServletRequest对象存放解析的字符串。HttpServletResponse

e)根据容器实现读取web.xml中的配置，根据url-pattern找到对应的Servlet-class.

f)判断内存中是否有对应的Servlet对象，如果有，则使用原来的，如果没有则创建信息的Servlet对象（使用反射） 【Servlet是但实例的】

g)构造servlet对象之后，容器调用init(ServletConfig){thisconfig = config ;init()}完成初始化

h)================初始化完毕=========

i)容器调用service()方法服务器请求，在调用service方法时会将事先创建好的request和response对象传入到service(ServletRequest,ServletResponse)中

j)Protected service()--->doXXxxxxx

k)我们在doXXX方法中，通过response对象想客户端产生响应。

l)Tomcat根据我们组装response对象，生成标准的HTTP响应字符串，通过网络回送给客户端浏览器

m)浏览器解析响应将正文显示。

===========================================================================================

获得ServletContext的方式？

\* ServletConfig.getServletContext();

\* HttpSession.getServletContext();

\* FilterConfig.getServletContext();

\* ServletContextEvent.getServletContext();

从服务器向客户端响应内容

getWriter

getOutputStream---->下载 中文文件名字

3.两种跳转

请求转发: RequestDispatcher

RequestDispatcher:负责转发的转发器对象，由容器创建对象，容器实现方法，我们调用

1.HttpServletRequest.getRequestDispatcher(path).forward(req,resp)

path:

相对路径

绝对路径 /：表示 服务器上工程的根

2.ServletContext.getRequestDispatcher(path).forward(req,resp)

只能使用绝对路径

重定向:

绝对路径 /表示福服务器的根 必须手动加工程名字

response.sendRedirect(request.getContextPath()+"/servlet1/a/b/c/d/other");

相对路径

response.sendRedirect("../servlet1/a/b/c/d/other");

getOutputStream可以输出任意数据，字符 字节都可以

浏览器原则是能打开就打开，下载原理就是让浏览器能打开也不打开，将响应内容以附件形式发送给浏览器，下载需要修改响应方式--》修改响应头：

"Content-Disposition":attachment;filename=xxx.jpg

===========================================================================================

保持会话的技术:cookie+session

1.cookie机制

1.案例

网页换肤

记住我

上次访问时间

最近浏览的商品

2.javax.servlet.http.Cookie:

1.Cookie是服务器（Servlet）发送给浏览器的一小撮儿信息。保存在浏览器上，稍后回送给服务器，因为cookie的值可以唯一定位一个客户端，所以cookie通常被用来会话管理

2.cookie由名字和值以及一些可选项构成。

3.Servlet发送cookie给浏览器通过使用response.addCookie()方法【底层通过添加HTTP响应头的方式将cookie发送给客户端的】

4.客户端回送cookie给服务器通过添加HTTP请求头的方式实现的。

===========================================================================================

1.bootstrap:twitter 推特工程师写的一套融合JS/HTML/CSS的前端框架，Bootstrap是最受欢迎的HTML、CSS和JS框架，用于开发响应式布局、移动设备优先的 WEB项目

2.bootstrap:响应式布局 兼容移动设备

3.响应式布局？

相同的内容，根据显示设备的不同，显示不同的效果。

===========================================================================================1.文件上传

客户端:浏览器

<form method="post" enctype="multipart/form-data">

enctype:application/x-www-form-urlencoded

参数以key=value&key1=value1的形式传送到服务器。

enctype:multipart/form-data

参数按照分隔符分割:

----adfadfaf12223sxvss

name="name"

adfadf

----adfadfaf12223sxvss

name="price"

12.3

----adfadfaf12223sxvss

name="author"

asdss

----adfadfaf12223sxvss

...文件

以二进制形式发送到服务器。

enctype对应请求头:

普通表单:Content-Type:application/x-www-form-urlencoded

文件上传:Content-Type:multipart/form-data;boundary=----adfadfaf12223sxvss

服务器:

普通表单:服务器（Tomcat）解析请求中的消息体，获得参数信息，setParameter() setHeader()

我们:request.getParameter() getHeader(String)

文件上传:服务器接收request信息，但是不解析参数，会setHeader，不会setparameter()，也就意味着我们不能request.getParameter()，服务器只是setInputStream()，我们可以使用request.getInputStream()方法获得输入流，本来我们应该这样处理:getHeader()没问题，不能用用getParameter，而是通过getInputStream获得输入流，读取并按照请求头的分隔符，自己手动解析请求中每一部分包括普通参数和文件信息。{我们不愿意做这工作}

第三方文件上传插件:负责解析请求，将文件信息和参数信息打包到自己肚子里。

实际: 构造第三方组件，交由第三方组件接管请求，我们只负责从第三方插件中

获得各种信息（普通表单数据+文件数据）

===========================================================================================

HIBERNATE:

框架：

a)体积：类----》组件-----》框架

b)功能：框架关注的是非核心的必不可少的逻辑单元

c)完整度: 框架是半成品。

d)种类: 前端：bs、easyui；持久层:(DB)、Hibernate、MyBATIS ..； 业务:Spring

Hibernate是一个生存于持久层的ORM框架的实现，是对JDBC的封装。

a)持久层【DAO层】:在系统中专注于数据持久化的部分。与数据库交互的部分

b)Hibernate作用与 JDBC一样，也是与数据库交互完成数据的CRUD操作。

c)ORM:Object Relational Mapping 对象关系映射 ORM表示一种技术，主要目的是将面向对象开发中的对象映射的基于SQL的关系型数据库中。

JDBC :addStudent(Student stu){ Hibernate addStudent(Student stu){

Connection con save(stu);-->参考映射文件。

PrepareStatement, } .xml

String sql=”insert into xx values(???)”

pst.setXXX(1,stu.getXX());

Pst.execute()

核心API:

a)org.hibernate.cfg.Configuration:

1.读取配置文件（属性文件和映射文件）

2.创建SessionFactory buildSessionFactory

b)org.hibernate.SessionFactory

1.创建Session openSession()

c)Org.hibernate.Session

1.save(Object)

2.update(Object)

3.delete(Object)

4.Object get()

===========================================================================================

1.Hibernate中的主键增长策略？

1. identity:表示使用数据库的自动增长语句，(适用于mysql，SQLSERVER 支持自动增长的数据)不适用于oracle，手动设置主键不再起作用

2.increment:先查询最大的id，然后最大ID+1作为新的ID添加到数据库中。mysql +oracle

3.sequence:适用于oracle，hibernate默认使用名字为hibernate\_sequence的序列生成主键。也可以使用指定的序列

<generator class="sequence">

<param name="sequence">xxx\_sequence</param>

</generator>

4.native:自动根据底层使用的数据库选择，相应的策略，如果底层使用的是mysql数据库，则native相当于identity,如果底层使用的是oracle数据库，则native相当于sequence

5.foreign

2.Session中get方法和load方法的区别?

1.get:立即加载模式，load是延迟加载模式，当执行到get方法时，会立即发出SQL语句，当执行到load方法时，不会立刻发出SQL语句，只有当真正使用到时才会发出SQL语句。

2.当数据库中没有对应记录时，get返回null，load返回ObjectNotFoundException

3.Session中的close\clear\evict 方法的区别？

close:关闭session

clear:清空缓存

evict:从缓存中将指定的对象清除

===========================================================================================

HQL:Hibernate 查询语言 面向对象

SQL:结构化查询语言

HQL是面向对象的查询语言，在HQL中查询的是类、类中属性，而不是表中字段。底层Hibernate会自动将HQL翻译成SQL语句执行，我们按照面向对象的原则写SQL，除了类名字、属性名字之外其他关键字不区分大小写。

\*注意：在HQL查询列表中不能直接出现\*

Hibernate如何分页？

Hibernate使用底层的分页机制进行分页，如果底层使用的mysql数据库，则使用limit分页，如果底层使用的是oracle数据库，则使用rownum分页。Hibernate将分页的功能打包到了相应的方法中，setFirstResult():从第几条记录开始查询，从0开始；setMaxResults():最大显示几条。

\*注意：当查询某几个字段而不是所有字段时，Hibernate返回Object数组，不能直接返回对象

\*注意：给占位符赋值 下标从0开始

\*注意：当明确知道返回一个结果时，使用uniqueResult

===========================================================================================

1.数据库中表与表之间的关系？

基于主键的关联:两个表的关联关系通过从表主键实现

基于外键的关联:两个表的关联关系通过外键实现

1 vs 1：

Country

id name

1

Capital

id name cid

1000 1

例子：Person : Idcard

1 vs m：

Dept

id name

Emp

id name did

例子：Goods:pic

m vs n:多对多的关系必须添加第三张表

Student Course sc

<one-to-one name="ca" class="com.etoak.po.Capital" cascade="all"></one-to-one>

one-to-one:表示一对一

name:表示属性名字

class:表示属性类型

cascade:表示操作当前表时是否级联影响对方数据

all:所有情况下都级联

save-update:保存或者更新时级联

delete:删除时级联

none:不级联

操作主方：

主方设置从方--目的是级联保存从方

co.setCa(ca);

从方设置主方--》目的是给从表的主键赋值

ca.setCo(co);//ca.getCo().getId()

IDao<Country> dao = new DaoImpl<Country>();

注意：在基于主键的一对一的关联关系中，查询主方会立刻将从方数据一并取出

Country c = dao.queryById(Country.class, 1);

System.out.println(c.getCa().getName());

dao.add(co);

操纵从方：

注意：在一对一的关联关系中，查询从方默认不会将主方一块取出。需要通过lazy属性修改设置

one-to-many:

inverse:表示关联字段的维护权是否反转

反转:交给对方维护

false:不反转 自己维护

insert into xian values(xxx)

insert into zhen values(id,name,xianid)

xx xx null xian.id

update zhen set xianid= xian.id where id=?

true:反转 多方维护关联关系 推荐

insert into xian values(xxx)

insert into zhen values(id,name,getXian().getId())

<set name="zhens" inverse="true" cascade="save-update" >

<key column="xianid"></key>

<one-to-many class="com.etoak.po.Zhen"/>

</set>

注意：一对多的关联关系，查询一方，不会立刻查询多方数据

注意：一对多的关联关系，查询多方，不会查一方

多对多：

Student(id ,name ,age)

Course(id,name)

Sc(id,sid,cid[,score])

第一种多对多:桥表中没有其他业务字段，只有外键

第二种多对多:桥表中有其他业务字段

===========================================================================================

package:代表包，在struts中一般使用package来组织代码，一个功能或者一个模块代表一个包

name:包名字 唯一

extends:继承 struts-default:是struts提供的默认包 必须继承

action:表示请求

name:表示请求的名字【不带后缀】

class:表示处理请求的类，在struts中一般叫Action类，普通类，默认处理方法是execute

result:表示结果

name:表示结果字符串 ，通常与Action类中处理方法的返回值一值

type:跳转类型

dispatcher:默认 以请求转发的方式跳转到其他页面

redirect:以重定向的方式跳转到其他页面

chain: 以请求转发的方式跳转到其他Action

redirectAction: 以重定向的方式跳转到其他Action

<package name="etoak" extends="struts-default">

<action name="hello" class="com.etoak.action.HelloAction">

<result name="success" type="dispatcher">success.jsp</result>

<result name="error">success.jsp</result>

</action>

</package>

Struts中的Action类和Servlet类有什么区别？

0.每次请求都会创建一个Action类的对象，Servlet是单实例的。

1.Action是一个普通类，不需要继承或者实现任何的类或接口，Servlet必须实现Servlet接口

2.Action不与容器耦合，测试方便，Servlet必须与容器耦合。

两种获得客户端参数的方式?

属性驱动

第一种获得客户端参数的方式：定义名字与客户端一样的属性名字+setter方法

模型驱动

第二种获得客户端参数的方式: 对象+setter+getter/前端 对象.属性

===========================================================================================

1.获得ServletAPI的几种方式？

1.ActionContext工具类 【获得四大Map 】

2.ServletActionContext 工具类【获得原始对象】

3.实现RequestAware SessionAware ApplicationAware 接口

4.实现ServletRequestAware ServletResponseAware..

2.几个资源文件

1.struts-default.xml:

该文件中定义了默认的包 struts-default包，该包定义了很多跳转的类型 dispatcher/chain/redirect/redirectAction，还定义了很多的拦截器和拦截器栈。struts的基础功能都被拦截器实现了。

2.struts-x.x.dtd:struts的dtd文件

3.default.properties:该文件定义了struts的默认行为。比如: 默认拦截后缀.action或者没有后缀， 默认文件上传的最大限制为2097152，该文件中的内容都可以在struts.xml中通过<constant>元素修改

3.struts流程？

服务器启动阶段: 当容器启动时，StrutsPrepareAndExecuteFilter就会被构造并初始化。在初始化方法执行时，就会读取配置文件(struts-default.xml和default.properties和struts.xml)此时出现配置文件的错误，服务器报错。没错误 正常启动。

请求到达阶段: 根据后缀链接相应的请求，到struts.xml文件寻找指定名字的action，使用指定类中的指定方法处理请求[我们自己的业务]，根据方法的返回值。到result元素中找到对应的结果，跳转到指定的页面或者是其他Action

4.struts底层并不是servlet实现的，Action也并没有封装Servlet，Action是个普通类。

===========================================================================================

servlet文件上传:

客户端: <form method="post" enctype="multipart/form-data">

服务器:接收请求，不解析，只负责setInputStream,所以我们在request只能获得getInputStream

{需要手动获得分隔符，将每一部分分割，手动解析文件数据}

以上工作交给第三方插件-smartupload

我们在servlet中：只需要负责将request交给第三方插件解析，然后从插件中获得文件数据即可。

struts文件上传:

客户端: <form method="post" enctype="multipart/form-data">

服务器:接收请求，不解析，只负责setInputStream,所以我们在request只能获得getInputStream

{需要手动获得分隔符，将每一部分分割，手动解析文件数据}

以上工作交给第三方插件-commons-fileupload

拦截器：文件上传拦截器负责从第三方组件获取文件数据，赋值给Action中的相应的属性，我们在Action中像使用不同参数一样，使用文件属性。直接获得文件对象

===========================================================================================

STRUTS中action如何返回json数据?

1.获得response直接写出到客户端

2.使用struts-json插件

1.导入插件jar包 struts-json-plugin-xx.x.x..jar;

2.继承json-default

===========================================================================================

interceptor:拦截器

1.拦截器是在拦截器模式的下的无状态的类。

2.拦截器是拦截Action执行的对象。拦截器提供了一种在执行ACTION中的方法之前或者之后执行另外一些业务逻辑代码的能力。

3. 拦截器必须实现的接口是Interceptor

案例:只有登陆之后，才能到主页面。

===========================================================================================

Filter:过滤器

1.过滤器是用来执行过滤请求的【响应】对象。

2.方法列表

Filter Servlet

init(FilterConfig) init(ServletConfig)

doFilter(ServletRequest,ServletResponse,Chain) service(..)

destroy() destroy

getServletConfig()

getServletInfo()

3.Filter执行过滤任务是在doFilter方法中。

4.FilterConfig和Filter的关系是一一对应的，一个Filter对应一个FilterConfig对象，存放一些初始化的参数信息。

5.FilterConfig: ServletConfig

getInitParameter() getInitParameter()

getInitParameterNames(); getInitParameterNames();

getFilterName(); getServletName();

getServletContext(); getServletContext();

6.获得ServletContext的方式？

ServletConfig.getServletContext();

FilterConfig.getServletContext()

HttpSession.getServletContext();

ServletContextEvent.getServletContext();

7.每一个filter都需要在web.xml中注册

8.案例:算命

过滤器必须实现Filter接口

Filter和Servlet的区别?

1.Servlet默认请求到达时构造对象并初始化，而Filter是在容器启动时构造对象并初始化。

2.Filter必须实现Filter接口，Servlet实现的是Servlet接口

===========================================================================================

MyBATIS: 作者: 克林顿.begin 曾用名:iBATIS

1.MyBATIS和Hibernate一样，也是一个持久层的ORM映射框架，侧重点是SQL的映射。

2.Hibernate全自动的:自动生成SQL，自动执行SQL，执行返回结果

Mybatis:半自动:自动执行SQL，自动返回结果，手动提供SQL语句

JDBC:全手动，手动写SQL，手动执行SQL，手动返回结果

3.MyBATIS的核心API:

SqlSessionFactoryBuilder:

build()

SqlSessionFactory:

openSession();

SqlSession

select

insert

delete

update

param1:执行的SQL语句的id param2:执行SQL语句时需要传入的参数

session.insert("addStu",stu);

$和#的区别？

#：相当于PreparedStatement，支持？占位

select \* from xx where name like '#{name}'

===》et

select \* from xx where name like '"et"'

select \* from #{tablename}

==>student

select \* from "student" XXX

$:相当于普通的Statement，参数是拼装上的

select \* from xx where name like '${name}'

===》et

select \* from xx where name like 'et'

mapper.xml文件范例以及介绍：

<mapper namespace="com.et1709">

namespace与接口名字保持一致

语句的id与方法名字一致

<insert id="addStu" parameterType="stu">

<selectKey keyProperty="id" resultType="int">

select @@last\_insert\_id as id

<!-- select xxx\_seq.next as id from dual -->

</selectKey>

insert into stu(sname,sage,semail)

values

(#{name},#{age},#{email})

</insert>

<insert id="addStu1" parameterType="list">

insert into stu(sname,sage,semail)

values

<foreach collection="list" item="stu" separator=",">

(#{stu.name},#{stu.age},#{stu.email})

</foreach>

</insert>

<!-- 当类中属性和表中字都不一致时：第一种方案：起别名 -->

<select id="queryStuById1" parameterType="int"

resultType="stu">

select

sid as id,

sname as name,

sage as age,

semail as email

from stu where sid=#{id}

</select>

<!-- 当类中属性和表中字都不一致时：第二种方案：使用resultMap -->

<resultMap type="stu" id="rmap\_stu">

<id column="sid" property="id"/>

<result column="sname" property="name" />

<result column="sage" property="age" />

<result column="semail" property="email" />

</resultMap>

<select id="queryStuById2" parameterType="int"

resultMap="rmap\_stu">

select \* from stu where sid=#{id}

</select>

<!-- mybatis中的动态查询标签 sql include if where foreach -->

<sql id="commonsql">select \* from stu</sql>

<select id="queryStuById" parameterType="int"

resultMap="rmap\_stu">

<include refid="commonsql"></include>

where Sid=#{id}

</select>

<select id="queryByConditions" parameterType="map"

resultMap="rmap\_stu">

<include refid="commonsql"></include>

<where>

<if test="name!=null">

and sname like '%${name}%'

</if>

<if test="email!=null">

and semail like '%${email}%'

</if>

</where>

</select>

<select id="queryByAge" parameterType="map"

resultMap="rmap\_stu">

<include refid="commonsql"></include>

<where>

sage in

<foreach collection="ages" item="age" open="("

close=")" separator=",">

#{age}

</foreach>

</where>

</select>

</mapper>

===========================================================================================

Reflection:反射

0.四大类型: class interface enum annotation

1.反射是JAVA语言提供的一套在运行期动态获得类中信息的API。

编译期:.java--->.class

运行期:.class---->逐句执行

2. 通过反射，我们可以在运行期动态的获得类中的属性和方法；通过反射，我们可以在运行期动态的执行类中的方法，通过反射，我们可以在运行期判断对象的类型；通过反射，我们可以在运行期动态的创建类的对象。

3.反射的基础类(平台类)：java.lang.Class类。

1.同学-----》Student 员工----》Emp 部门--》Dept 苹果--》Apple

类----》Class

Class类是用来描述类的类，一个Class类的对象代表一个具体的类。

2.在JVM加载完一个类之后，创建该类的任何对象之前，会首先在内存的堆区创建一个Class类的对象，使用该对象存放JVM所加载的类的信息。在运行期JVM是与一堆的Class对象交互。

3.Class类没有公共的构造方法，Class类的对象是JVM自动构造的

4.一个类的对象可以有很多个，但是这些对象都共享同一个Class对象

5.所有基本数据类型和关键字void都有对应的class对象。

6.基本数据类型和对应的包装类不是一个Class对象

7.所有具有相同元素类型和维数的数组都共享同一个Class对象

如何获得Class类的对象?

1.)类名字.class 固定形式 Student.class Emp.class

2.)对象.getClass()

3.)Class.forName("java.lang.String")

Class代表的是类，一个Class对象代表一个具体的类

Method代表方法，一个Method对象代表一个具体的方法

Field代表属性，一个Field对象代表一个具体的属性

getField()

getFields():获得本类和直接父类中的所有的公共的属性

getDeclaredField(String)

getDeclaredFields();获得本类中所有的属性{包括私有的}

@Entity：表示该类是一个实体类，只有加上该注解才能被Hibernate管理 必选项

@Id：表示该属性是一个注解

@GeneratedValue:主键如何生成

IDENTITY：使用数据库的自动增长语句来实现自动增长。

===========================================================================================

1.enum 枚举:枚举是JAVA5.0之后提出的新特性，主要表示一些固定值，一周七天，性别，也会生成.class

2.创建枚举使用关键字 enum

3.注解:annotation @Override

JDK中的java.lang包中提供的注解:

@Override:表示子类重写父类的方法，如果子类使用了该注解，

但是没有重写父类方法，则编译器发出错误消息。

@Deprecated:表示方法已经过时,如果使用了过时的方法，则编译器发出一条警告消息。

@SuppressWarnings:取消警告 不显示警告

自定义注解:使用注解的关键字 @interface

java.lang.annotation:

@Retention:表示注解的保留策略，用在其他注解上

value=RetentionPolicy.SOURCE:注解保留在源码中，编译之后不保存

RetentionPolicy.CLASS:注解保留到class文件中，运行期不加载 默认行为

RetentionPolicy.RUNTIME 运行期加载到内存，可以使用反射获得

@Target:表示注解使用的程序元素的种类

value=ElementType: TYPE

ElementType.METHOD

===========================================================================================

1.文件上传的表单和普通表单有什么区别？

文件上传:<form method="post" enctype="multipart/form-data">

普通表单不需要

2.文件上传的服务器端是如何处理的？

服务器:

普通表单：服务器（Tomcat）解析请求中的消息体，获得参数信息，setParameter() setHeader()

我们: request.getParameter() getHeader(String)

文件上传:服务器接收request信息，但是不解析参数，会setHeader，不会setparameter()也就意味着我们不能request.getParameter()，服务器只是setInputStream()我们可以使用request.getInputStream()方法获得输入流，本来我们应该这样处理:getHeader()没问题 ，不能永getParameter,而是通过getInputStream获得输入流，读取并按照请求头的分隔符，自己手动解析请求中每一部分包括普通参数和文件信息。{我们不愿意做这工作}

第三方文件上传插件:负责解析请求，将文件信息和参数信息打包到自己肚子里。

实际: 构造第三方组件，交由第三方组件接管请求，我们只负责从第三方插件中获得各种信息（普通表单数据+文件数据）

3.Hibernate如何分页？

Hibernate使用底层的分页机制进行分页，如果底层使用的mysql数据库，则使用limit分页，如果底层使用的是oracle数据库，则使用rownum分页。Hibernate将分页的功能打包到了相应的方法中，setFirstResult():从第几条记录开始查询从0开始；setMaxResults():最大显示几条

4.MyBAITS和Hibernate的区别?

Mybatis;半自动

Hibernate：全自动

5.MyBATIS中核心API的作用？

SqlSessionFactoryBuilder():构造SqlSessionFactory

SqlSessionFactory;获得SqlSession

SqlSession：核心 CRUD insert update delte selectOne()

6.Hibernate流程？

1.Configuration解析配置文件(属性文件 映射文件)

属性文件:hibernate.cfg.xml

映射文件:\*.hbm.xml

2.创建SessionFactory

3.打开Session

4.开事务 Connection.setAutoCommit(false);

5.crud

6.COMMIT/rollback

7.close

7.Hibernate中的查询方式有哪些？

1.OID get load Object Id

2.HQL

3.QBC

4.sql

8.实现一个将指定字符串转换成日期的方法？

SimpleDateFormat sd = new SimpleDateFormat("yyyy-MM-dd");

sd.parse(new java.util.Date());

9.SmartUpload的步骤？

1.构造实例

2.初始化init（request,）

3.upload

10.MyBATIS配置文件的作用？

config.xml:环境信息+sql映射文件位置

mapper.xml: SQL语句的映射信息

11.MyBATIS中JavaType和JDBCType的区别？(5)

javaType:指的是java中的属性类型 java.lang.String

JDBCType：表中的类型 varchar

12.MyBATIS中的Mapper标签中，namespace属性是做什么的？(5)

1.避免命名冲突

2.Mapper方式实现Mybatis时，需要namesapce需要与接口保持一致

13.MyBATIS中当实体类中的属性名字和字段名字不一致时两种处理方式？(5)

insert into stu(sanme) values(#{name})

parameterMap

查询:

1.起别名

2.<resultMap>

14.列举sql中的5个聚合函数，说明作用？(10)

15.Excel操作的两种控件？(15)

JXL+POI

//非本部分内容？

・1.怎么检查一个字符串只包含数字？(5)

2.Integer source = 12; Integer target=12;

source==target将返回什么？为什么？(10)

3.JAVA中如何格式化一个日期？如格式化为 ddMMyyyy的形式?(5)

4.Eayui如何弹出一个提示框?

5.列举5中不同的Easyui的控件？

6.简介Spring的IOC和AOP？

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

单行函数:每条记录都会返回一个结果

字符：

大小写

lower upper initcap

长度

length lengthb

查找

instr

替换

replace

截取

substr

填充

lpad rpad

空格:trim ltrim rtrim

数值:

round trunc ceil floor

日期:

sysdate

months\_between

add\_month

next\_day

转换:

to\_date(xxx,"yyyy-MM-dd")

通用:

decode nvl nvl2

多行函数(聚组函数)：多条记录经过计算返回一个结果

avg() min() max() count() sum()