能够重现的问题，才是好问题。

错误提示：在action中无法获取session中的user信息；

原因：使用Struts2中的SessionAware通过的Map对象来注入session时，没有引入SessionAware接口。

错误提示：The method parseInt(String) is undefined for the type Loginsample

之前运行的时候，没有问题，之间没有改变，却报错。

解决办法：

之前是使用Integer.parseInt函数将String类型转为int，现在改成下面这种形式。

        Integer rowid = new Integer(id);

        row.setId(rowid);

        //row.setSampleid(Integer.parseInt(sampleid));

没有找到错误原因，直接改变使用的方法，以后知道原因的时候，写在这儿。

出现错误提示弹框： “Building workspace has encountered a problem. Errors during build.”

解决方法： MyEclipse 里面project->Build Automatically上的对勾去掉就OK了；

错误信息：ids for this class must be manually assigned before calling save(): sample.db.Completedsample

原因分析：从字面上理解的意思是，在save之前，必须手动指定id，其中id的<generator class="assigned" />；

解决办法：将主键设为自增，原先需要主键表示的数据，重新设置一个键。此时主键id的 <generator class="identity" />；之前主键没有设为自增，将id的generator的class设为increment也是同样的效果。

资料：<id>元素映射了相应数据库表的主键字段，对其子元素<generator class="">,其中class的取值可以为increment、identity、sequence、hilo、native……等，更多的可参考hibernate参考文档，一般取其值为native ；ps：我用的是MySQL，取值为native时也会报错。

<http://blog.163.com/starfish_yang/blog/static/686256852010102785912790/>  这个blog中有每种class值的详解。

总结：还是没有弄清楚为何会报错，开始的时候数据库中id不是自增的，程序中也setId了，出现了如上的报错信息。使用main函数测试，非自增的情况下，是可以直接指定id值的，然后，使用网页测试，也是可以的，这个问题没有能够重现。

故障表现:找不到网页；

原因：使用Struts的标签获取后台数据，没有写setter和getter，该故障不具有重复性。

这是一个按钮，实现了onclick事件（js函数带参数），使用了Struts的property标签，放在这儿做一个备忘。

                    <td>

                        <button id=<s:property value="#row.id"/> type="submit"

                            class="btn btn-default"

                            onclick="welcome(<s:property value="#row.id"/>)">结果录入</button>

                    </td>

故障表现：tomcat可以正常启动，但是浏览器显示找不到页面，404。

     经观察发现tomcat启动时控制台出现错误提示：

严重: Dispatcher initialization failed

严重: Exception starting filter struts2

严重: Error filterStart

2017-3-4 16:06:35 org.apache.catalina.core.StandardContext startInternal

严重: Context [/mylims1.2] startup failed due to previous errors

2017-3-4 16:06:35 org.apache.catalina.loader.WebappClassLoader checkThreadLocalMapForLeaks

严重: The web application [/mylims1.2] created a ThreadLocal with key of type [com.sun.faces.util.Util$1] (value [com.sun.faces.util.Util$1@377e4cec]) and a value of type [java.util.HashMap] (value [{com.sun.faces.patternCache={ = }}]) but failed to remove it when the web application was stopped. Threads are going to be renewed over time to try and avoid a probable memory leak.

2017-3-4 16:06:35 org.apache.catalina.startup.HostConfig deployDirectory

原因：经排除发现，一个新的<action></action>配置文件有问题，把他注释之后，可以正常使用。最后发现，是action的class属性中没有写包名。

新建一个项目，并布置各框架：

tomcat的项目保存路径： D:\C++\JavaSoft\apache-tomcat-7.0.39-x64\webapps

myeclipse中更改项目名称：右击项目->copy->paste->更改名称->确定。（还要更改一个设置，以后用到了再补吧）

错误提示：Syntax error, insert "Finally" to complete TryStatement

原因：复制代码的时候，多了一个“try{”

故障：hibernate操作数据库时，可以正常添加数据库，但是使用下面的语句时，报错：Query为空。

dbss = HibernateSessionFactory.getSession();

return  dbss.createQuery("FROM LoginSample").list();

原因：数据库的名称中间不能有大写字母，开头可以大写。以后都用小写，无论是数据库名称，还是列名。。

Myeclipse显示行数：windows菜单->选择preference->general->Editors->Text Editors->Show line numbers,勾选确定

故障：使用Struts2框架，表单提交后，抛异常，得不到自己想要的回复。

原因：使用日志功能，下大量的记录，确定是参数没有从jsp传入action，最后发现jsp中的变量名和action中的大小写不同。

错误信息：The return type is incompatible with ActionSupport.getLocale()

原因：这是一个自动生成的getter，但是现实类型不匹配，可能是库中已经有了一个getLocale，我这个与之重复了。（有点类似于使用关键字命名变量）

批量注释：ctrl   + ?(/) ，取消也是

Java：String和Date、Timestamp之间的转换

<http://yunnick.iteye.com/blog/1074495>

错误信息：java.lang.NullPointerException //空指针

出错: null

原因：hibernate自动生成POJO类后，没有更改生成的类的配置文件（去空格）。

该报错信息又出现了一次，原因也是配置信息出错，之前更改了数据库的名字，然后重新生成POJO类的时候，原先的配置都没有删除，so，报错。手动删除后，恢复，共三处，和新建时一一对应。

错误信息：Batch update returned unexpected row count from update [0]; actual row count: 0; expected: 1

原因：这个异常是由于主键设置为自增长，而在我们插入记录的时候又设置了ID的值导致的。

报错信息：Type mismatch: cannot convert from org.hibernate.Transaction to javax.transaction.Transaction

原因： 要引用（import） hibernate中的 Transaction，而不是java.sql.Transaction，直接右键引用的时候，有两个import，一个是java的，一个是hibernate的；数据库方面的，要用hibernate的。

如何在Myeclipse中显示servers：

<http://jingyan.baidu.com/article/60ccbceb03814364cab1978d.html>

点左下角的小图标（show view as a fast view），然后点“Other”，点击“MyEclispe Java Enterprise”前面的小三角形，就可以看到Servers了。

20170113：

错误提示：

The project was not built since its build path is incomplete. Cannot find the class file for java.util.Map$Entry. Fix the build path then try building this project

The type java.util.Map$Entry cannot be resolved. It is indirectly referenced from required .class files    HibernateSessionFactory.java

原因：项目不适用jdk1.8，改成MyEclipse自带的1.6就行了。tomcat还是原来的1.8。（更改默认的jdk，然后重启）

20170110：

问题表现：MyEclipse中启动tomcat，浏览器中无法找到网页，即404；单独启动tomcat可以正常访问；

原因：jdk版本错误，tomcat设置中的jdk和安装的jdk版本不一致，另，环境变量Path和JAVA\_HOME的路径也要多加注意，防止另外安装了jdk，却没有更改路径。

MyEclipse Servers视窗出现“Could not create the view: An unexpected exception was thrown”错误解决办法

RT：

打开所在的wordspace文件夹，在下面子文件夹

.metadata\.plugins\org.eclipse.core.runtime\.settings

删除

com.genuitec.eclipse.ast.deploy.core.prefs

即可

这个问题的出现是在启动myeclipse之后出现的，这时还没有启动tomcat，因此可以排除tomcat中项目代码问题。应该是myeclipse在初始化工作空间workspace 的时出的问题。

仔细看下错误发现有一条 at com.genuitec.eclipse.ast.deploy.core.Deployment.这个错误，于是到工作空间下面的D:\Workspaces\.metadata\.plugins\org.eclipse.core.runtime\.settings目录下面找到了名称为at com.genuitec.eclipse.ast.deploy的prefs文件，打开文件一开，哈哈，原来里面都是原来的一些项目信息，怪不得会报java.lang.NullPointerException错误呢。果断将此文件删除，重启myeclipse10问题解决

在hibernate.cfg.xml文件里面加入如下的配置（c3p0 ）：

<!-- 最大连接数 -->

<property name="hibernate.c3p0.max\_size">20</property>

<!-- 最小连接数 -->

<property name="hibernate.c3p0.min\_size">5</property>

<!-- 获得连接的超时时间,如果超过这个时间,会抛出异常，单位毫秒 -->

<property name="hibernate.c3p0.timeout">120</property>

<!-- 最大的PreparedStatement的数量 -->

<property name="hibernate.c3p0.max\_statements">100</property>

<!-- 每隔120秒检查连接池里的空闲连接 ，单位是秒-->

<property name="hibernate.c3p0.idle\_test\_period">120</property>

<!-- 当连接池里面的连接用完的时候，C3P0一下获取的新的连接数 -->

<property name="hibernate.c3p0.acquire\_increment">2</property>

<!-- 每次都验证连接是否可用 -->

<property name="hibernate.c3p0.validate">true</property>

用的MyEclipse，选中代码，然后Ctrl+Shift+F就可以格式化，自动缩进。

快捷键shift+ctrl+f不管用的话，用source-->format

CTRL+Z:撤销上一步

    CTRL+Y：撤销下一步

    ALT+?：用于弹出网页代码（XML，html）中的提示信息

    F6：单步跟踪断

    F7：进入调用方法内部继续跟踪断点

    F8：直接去下一个断点

新建一个Web Project需要做哪些事情？

1.把index.jsp字符集改成需要的，在第一行pageEncoding="UTF-8"；

2.servlet中doGet和doPost中，把原来的那段代码换成这一段，然后再在上边加上自己的代码。

       // 应答

        response.setContentType("text/plain");

        response.setCharacterEncoding("UTF-8");

        PrintWriter out = response.getWriter();

        out.write("ok");

        out.close();

Java中File类的构造函数：

public class FileDemo {

     public static void main(String[] args){

         //构造函数File(String pathname)

         File f1 =new File("c:\\abc\\1.txt");

         //File(String parent,String child)

         File f2 =new File("c:\\abc","2.txt");

         //File(File parent,String child)

         File f3 =new File("c:"+File.separator+"abc");//separator 跨平台分隔符

         File f4 =new File(f3,"3.txt");

         System.out.println(f1);//c:\abc\1.txt

     }

 }

该类其他的方法，见java学习指南的笔记的文件篇

java中的super()是什么

要说super就先要说this。

"this"，作为一个特殊的关键字，它的规则如下：

1。可以表示构造函数传递。this(a,b)表示调用另外一个构造函数。这里面的this就是一个特殊语法，不是变量，没有什么类型。

2。可以在一个类的非static成员内部使用，表示当前这个对象。此时，this就是一个final的普通变量，它有静态类型，就是这个类C本身；它有动态类型，就是当前这个对象的类型。你可以对它调用成员函数，把它传递给别的函数，等等等等。只要一个C类型的final变量可以出现的地方，它就可以出现。

"super"。它和"this"类似，但是也有不同的地方。

1。表示调用父类的构造函数。也是一个特殊语法，不是变量，没有什么类型。

2。可以在一个类的非static成员内部使用。比如super.method()。

但是，注意，这个super.method()只是长得跟some\_var.method()一样，一个语法糖而已。实质上，"super"根本不是一个变量。

为什么不是？因为如果是就坏了。java里面有一个金科玉律：任何public非static函数的调用都是多态的。

所以，如果super是个变量，也指向当前对象，那么，不管super的静态类型是什么super.method()必然调用的是子类的那个版本，而不会是我们期望的，静态地选择父类的那个版本。

所以，你只要把super.xxx看成一个特殊的语法，比如理解为“super::xxx”就好了。

既然super不是一个变量，那么不能把它象一个普通变量那样"=="，或者传递给某个函数就看起来很正常了，是么？何况，你其实也用不着它，有this你这些要求就都可以办到了。

3。super的另外一个作用是调用父类的protected函数。只有通过"super"这个魔咒，我们才能操作父类的protected成员，别无它法。

1.2 网页文件(html)

HTML

HTML, HyperText Markup Language

超文本标记语言 ：用于描述网页里的内容与显示

参考教程：(w3school)

<http://www.w3school.com.cn/html/>

创建网页

用HTML语法写成的文件称为网页文件 ，后缀.html 或 .htm ( 显示后缀: 控制面板 | 文件夹选项 | 查看 。。)

一个最简单的网页

<html>

     <head>

          <title> 第一个网页 </title>

     </head>

     <body>

          <p> 阿发你好 </p>

     </body>

</html>

规则：

（1）标签一般成对出现 <xxx> </xxx>

（2）基本框架为

<html> <head>…</head><body>…</body></html>

最基本的标签

<html> ：该标签是根，所有内容放在它里面

<head> : 该网页的相关参数，里面内容并不 直接显示。例如，<title>用于指定窗口标题

<body> : 正文部分。里面的内容直接用于显示。

其中，<head></head>可以不提供，但一般都 是写上的。

更多语法

(1) 注释：用于对HTML代码进行注释，并不影响网页的显 示

<!-- this is comment -->

(2) 标签属性

例如，

<a href="http://afanihao.cn" > 去学习 </a>

(3) 单标签

有两类标签, 还有一种是只有开始没有结束

<img src="haha.jpg" />

HTML语言的执行

HTML语言是一种解释性语言，直接由流览器解 释执行。

例如，

当浏览器看到

<img src="image/book.jpg" />

的时候，就会根据参数来加载相应的图片显示。

小结

介绍了HTML语言的概念，基本框架，执行方 式

<html>

    <head>

        <title> first test </title>

    </head>

    <body>

        <!--注释 -->

        <p> 206425 </p>  <!-- 段落paragraph -->

        <!-- anchor 超文本链接，href是其属性 -->

        <a href="http:afanihao.cn" > 去去去  </a>

    </body>

</html>

1.3 样式单(CSS)

CSS, Cascading Style Sheets，层叠样式单，简称样式单，用于描述网页的元素的样式

参考教程：(w3school)

<http://www.w3school.com.cn/css/>

什么样式？前面，我们已经用HTML指定网页的内容（文本、图片）。。。现在，需要用CSS来规定它们的显示样式。

比如，一段文字的字体、大小、颜色、对齐方式。一个图片的显示尺寸。

一个简单的样式单 example-A

首先，在<head>添加一段<style>

<style type="text/css">

     .example

     {

          color: red;

     }

</style>

然后，在<body>里使用它

<p class="example"> 阿发你好 </p>

另一个样式单 example-B

定义两个样式red\_text和blue\_text

.red\_text { … }

.blue\_text{ … }

使用样式

<p class="red\_text"> 本篇包含Java的所有核心概念。</p>

<p class="blue\_text">面向以下两类学员：<p>

<p class="red\_text"> (1) 若已经有Java基础，前面可以略读一遍，第14节开始正式进入。</p>

<p class="blue\_text"> (2) Java零基础、但已经学完"C/C++语法篇"的初学者 ，请从第1节开始学习。</p>

单独作为文件 example-C

可以把CSS样式单独写成一个文件，然后在.html中包含进来

例如，建立一个common.css，在里面写上样式定义。

然后在example-C.html中包含进来:

<link type="text/css" rel="stylesheet" href="css/common.css">

小结

介绍了CSS的用途，定义与使用

CSS不是本课程的重点，更多用法请参考w3school教程。

或查看课程的补充文档：《CSS使用方法简介.pdf》

以后使用的时候，在看文档吧

1.4 建立网站(tomcat)

Tomcat，全称Apache Tomcat ，一款支持Java的Web服务器 ，支持windows, linux多种平台, 本篇教程在 windows平台上做演示。

官网 http://tomcat.apache.org/

下载与安装

1. 安装JDK

如果不知JDK为何物，请看Java基础篇

2. 下载 tomcat压缩包

apache-tomcat-8.0.32-windows-x86.zip

apache-tomcat-8.0.32-windows-x64.zip

直接解压缩即可使用

(如果你的JDK是x64的，则tomcat也要选择x64 版 本)

运行tomcat

在apache-tomcat-8.0.32\bin下，

startup.bat - 启动服务

shutdown.bat – 关闭服务

直接双击startup.bat，就可以启动tomcat服务 器。。。

测试: 在浏览器里输入

http://127.0.0.1:8080

tomcat启动失败？

1. 没有安装JDK?

在控制台里输入 java命令，看有没有反应。

2. 端口冲突

tomcat默认端口为8080，如果你的电脑上已经运 行了一个8080的服务器，则tomcat启动失败。。。

解决办法：修改conf/server.xml，

<Connector port="8080" protocol=…

添加自己的网站

在webapps目录下，建立一个文件夹，把自己 的网页放进去。

例如，

(1) 创建一个子目录my

(2) 添加一些网页 my\test.html

(3) 在浏览器输入

http://127.0.0.1:8080/my/test.html

放到公网上？

为了让别人也能访问你的网站，你需要公网上有 台服务器。。。

(1) 申请IP。。

(2) 申请域名 xxx.cn 。。

(3) 域名绑定。。

然后，在全国各地都能在浏览器访问你的网站了。

例如，http://afanihao.cn

流程

假设用户在浏览器里输入 http://afanihao.cn/test.html

则内部过程依次为：

1. 浏览器通过网络连接到服务器123.57.248.214 (socket编程)

2. 浏览器发送 get /test.html消息给服务器

3. 服务器应答，读取/test.html文件的数据，发给浏览器

4. 浏览器解析test.html中的内容，显示给用户

流程

用wireshark 抓包显示

过滤 ip.addr == 123.57.248.214

follow tcp stream …

默认网页

例如，我们在浏览器里输入

http://127.0.0.1/my/

没有指定文件名，则tomcat会自动寻找index.html这个 文件 即，默认文件为 index.html (或 index.jsp)

图片及css

如果在网页使用了图片，则图片目录也拷贝过来。

例如，下面的目录结构

my

|\_ image

|\_ css

|\_ index.html

|\_ help.html

小结

介绍了tomcat服务器的基本方法

2.1 表格(table)

表格数据

在实际工程中，经常要显示一些表格数据。 展示。。

<table>

此标签用于显示表格状的数据

<table>

     <tr> </tr>

     <tr> </tr>

</table>

用tr表示一行的内容（table row) ，每一行里再定义每一列的显示

<th> 表头

<th> table header用于定义表头，即列标题或 行表示。一般默认粗体显示。

如，

<tr>

     <th> 学号 </th> <th> 姓名</th>

</tr>

<td>

<td> 用于定义单元格的内容

如，

<tr>

     <td> 201601 </td> <td> shaofa </td>

</tr>

CSS 定义表格的默认样式

用CSS修改标签的默认样式

table

{

     border-collapse:collapse;

}

table,th, td

{

     border: 1px solid black;

}

小结

介绍了HTML中表格的简单用法

        <table>

            <tr>

                <th> 学号 </th>

                <th> 姓名 </th>

                <th> 手机号 </th>

            </tr>

            <tr>

                <td> 201601 </td>

                <td> shao </td>

                <td> 156412536541 </td>

            </tr>

2.3 超链接(a)

<a> anchor ，超链接，点击后浏览器将打开另一网址

例：

<a href="http://afanihao.cn" > go to study </a>

属性

href: 目标URL ，可以跳到一个相对地址 ，如， href="help/start.html"

也可以跳到一个绝对地址 ，如，href="http://afanihao.cn"

target: \_self \_blank \_top \_parent

\_self是指在本窗口中打开新地址

\_blank是让浏览器新开一个窗口

把图片作成超链接 ，例如，

<a href="http://afanihao.cn">

<img src="image/logo.jpg" /> 点击进入

</a>

则点击图片也能打开目标链接。

CSS : 定义样式

效果：平时无下划线，当鼠标悬停到上方时 显示下划线。

a:hover

{

background-color:#FF704D;

text-decoration: underline;

}

参考example-B.html

小结

介绍了超链接的用法

2.4 表单控件(form controls)

表单 form

<form> 用于构建一个表单 ，所谓表单，是用于收集用户输入的一个界面 (相当于一个对话框)

<form> … 控件放在这里… </form>

一个简单的表单

<form>

用户名:

<input type="text" /> <br/><br/>

密码 :

<input type="password" /> <br/><br/>

<input type="button" value="提交" />

</form>

表单的布局

布局：可以把各表单元素放在表格或div里

例如，

<form>

<table>

<tr> … 第一行控件 </tr>

<tr> … 第二行控件 … </tr>

…

</table

</form>

详见example-B.html

表单控件: 文本框

例,

<input type="text" />

可以设置很多属性，具体请参考w3school教程。

注意：有些属性是在HTML5中引入的。

常用属性：

name, 字段名称

value, 初始值

表单控件: 复选框

例，

<input type="checkbox" name="married"/>已婚

<input type="checkbox" name="hasCar"/>有车

<input type="checkbox" name="hasHouse"/>有房

注：checkbox通常与一段文字配合显示，控件本身只 是一个小框框。

常用属性：

name, 字段名称，如name="married"

checked, 是否选中，如checked="checked"

表单控件: 下拉列表

例，

友情链接: <select name="friendLinks">

<option value="youku">优酷</option>

<option value="tudou">土豆</option>

<option value="qiyi">爱奇艺</option>

</select>

注：<select> 通常与一段文字配合显示，控件本身只是一 个小框框。

设置默认选项:

<option value="tudou" selected="selected" >土豆</option>

表单控件: 按钮

有两种方式:

<input type="button" value="点我" />

或

<button type="button">Click Me!</button>

两种均可使用

其他控件

在w3school演示。。

初学者没必要把每一项都深入了解。。

尤其是对于后台工程师，只需要稍微了 解。。。

需要的时候再来看w3school上的教程就行了

小结

介绍了常见控件的基本写法

3.1 JavaScript简介

JavaScript，一种嵌在HTML网页中脚本语言

（1）它是程序语言，由浏览器解释执行

（2）它和Java没有关系

（3）使用场合：主要用于网页，但也可独立 运行（比较少见）

“雷锋 和 雷锋塔 的关系！”

JavaScript

嵌在网页里面，放在<script>标签里

<script>

… JavaScript代码 …

</script>

例子 example-A

当鼠标点击一个按钮时，弹出一个欢迎窗口。。。

<html>

<head>

    <script>

        /\* 按钮被点击时的处理函数 \*/

        function welcome()

        {

            alert("欢迎来到afanihao.cn");

        }

    </script>

</head>

<body>

    <button type="button" onclick="welcome()" > 戳我 </button>

</body>

</html>

例子 example-B

当鼠标经过时，更改显示文字。。。

<html>

<head>

    <style>

        /\* 定义样式 \*/

        #example

        {

            width: 400px;

            height: 20px;

            background-color: blue;

            color: white;

            text-align: center;

            padding: 4px;

        }

    </style>

    <script>

        /\* 鼠标经过 \*/

        function onMouseOver( obj )

        {

            obj.innerHTML="Click Me!";

        }

        /\* 鼠标移出 \*/

        function onMouseOut( obj )

        {

            obj.innerHTML="戳我!";

        }

    </script>

</head>

<body>

    <div id="example"

        onmouseover="onMouseOver(this)"

        onmouseout="onMouseOut(this)" >

        戳我!

    </div>

    </body>

</html>

JavaScript的作用

网页里有什么?

HTML: 内容

CSS: 显示样式

JavaScript: 控制网页行为

小结

介绍了JavaScript中概念和作用

JavaScript是属于前端的技术，由浏览器执行

推荐参考教程：w3school

[http://www.w3school.com.cn](http://www.w3school.com.cn/)

3.3 JavaScript对象类型

内置的对象类型

Number, String, Boolean : 数字，字符串，布尔值

Array : 数组

Date : 日期操作

Math: 数学函数

RegExp : 正则表达式

Global : 全局函数

Number 数字

Number是浮点类型，无论是整数还是小数，都用它表示。

var n1 = 34;

var n2 = 12.345;

document.write("n1: " + n1 + ", n2:" + n2);

创建了2个对象，n1,n2都是Number类型 ，也可写作：

var n2 = new Number(12.345);

String 字符串类型

String表示字符串

var str1 = "haha";

document.write("str1: " + str1 + "<br>");

这里创建了1个String对象 ，也可写作：

var str1 = new String("haha");

Boolean 布尔类型

Boolean类型表示真假，可以为true或false这两个值

var ok = true;

document.write("ok ?: " + ok + "<br>");

其中，创建了1个Boolean对象 ，也可写作：

var ok = new Boolean(true);

Number <-> String

例：

/\* Number -> String \*/

var n = 123.45;

var str = n.toString();

document.write("str: " + str + "<br>");

/\* String -> Number \*/

var n2 = new Number(str);

document.write("n2: " + n2 + "<br>");

Boolean <-> String

例：

/\* Boolean -> String \*/

var v = true;

var str2 = v.toString();

document.write("str2: " + str2 + "<br>");

/\* String -> Boolean \*/

var v2 = new Boolean(str2);

document.write("v2: " + v2 + "<br>");

字符串拼接

字符串用+拼接，直接把对象toString()之后再 合起来

var result = str + n + ok ;

str: String

n: Number

ok: Boolean

则result也为String类型

以下result值为多少?

var n1=1, n2=2;

var result1 = "got: " + (n1 + n2);

var result2 = "got: " + n1 + n2;

注：小括号优先级高，先对小括号内的表达 式运算

var是什么意思？

以下代码是没有问题的：

var x = null;

x = "shaofa";

x = 12;

你能理解吗？

类比Java代码：

Object x = null;

x = new String("shaofa");

x = new Integer(12);

即：所有的对象都是Object类型。在JavaScript 用var来表示一个对象。

小结

介绍了三种内置类型: Number, String, Boolean 以及它们之间的转换

3.4 JavaScript调试技术

这些按钮的功能点如下：

* **Elements**:查找网页源代码HTML中的任一元素,手动修改任一元素的属性和样式且能实时在浏览器里面得到反馈。
* **Console**:记录开发者开发过程中的日志信息，且可以作为与JS进行交互的命令行Shell。
* **Sources**:断点调试JS。
* **Network**:从发起网页页面请求Request后分析HTTP请求后得到的各个请求资源信息（包括状态、资源类型、大小、所用时间等），可以根据这个进行网络性能优化。
* **Timeline**:记录并分析在网站的生命周期内所发生的各类事件，以此可以提高网页的运行时间的性能。
* **Profiles**:如果你需要Timeline所能提供的更多信息时，可以尝试一下Profiles,比如记录JS CPU执行时间细节、显示JS对象和相关的DOM节点的内存消耗、记录内存的分配细节。
* **Application**:记录网站加载的所有资源信息，包括存储数据（Local Storage、Session Storage、IndexedDB、Web SQL、Cookies）、缓存数据、字体、图片、脚本、样式表等。
* **Security**:判断当前网页是否安全。

**Audits**:对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。比如列出所有没有用到的CSS文件等。

JavaScript 调试

由于编辑器一般比较简陋，JavaScript书写时非常容 易出错，必须尽早学会调试技术。

两种手段:

1. 单步调试

2. 打印调试

console.log( … ) 或 console.trace(…)

JavaScript 调试

演示：打开调试界面。。。

注：请使用Chrome，别的浏览器不一定支持调试

打印调试

例如，

var n1 = 34;

var n2 = 12.345;

console.log("n1: " + n1 + ", n2:" + n2);

调试语句不会影响正常代码浏览，日志被输出到控 制台。。。

单步调试

使用Chrome单步调试：既可以发现语法错误，也可 以发现运行错误

1. 下载断点

2. 刷新页面

3. 开始调试

更多用法会在后面的视频中穿插演示。。。

3.5 JavaScript数组Array (1)

Array 数组对象

Array, 数组 ，例，

var a = [ 1, 2, 3 ];

var books = [ "c++", "Java" ];

下标范围: 0~ N-1

例如, books[0] 为 "c++"

数组中的每个元素都是对象。

第2种方式：

var b= new Array();

b[0] = "C++";

b[1] = "Java";

第3种方式：(元素对象直接列在参数里)

var c = new Array("C++", "Java", "JavaScript");

遍历

/\* 常规遍历方法 \*/

var books = [ "c++", "Java" ];

for(var i=0; i<books.length; i++)

{

     … books[i] …

}

books.length: 数组长度

books[i] : 第i个对象

数组的对象类型

对象的类型不要求一致，例如，

var a = new Array(4);

a[0] = 34;

a[1] = "shaofa";

a[2] = null;

此时，a[0]为Number, a[1]为String, a[2]为null,  a[3]为undefined

数组的对象类型

跟Java对比一下

Object[] arr = new Object[3];

arr[0] = new Integer(32);

arr[1] = new String("shaofa");

arr[2] = null;

join() 转成字符串

使用join函数，可以把数组转成字符串，并加上分隔 符。例如，

var a = [ "c++", "Java" , "JavaScript" ];

var t = a.join(";"); // 指定分隔符

则t的值为 "c++;Java;JavaScript"

sort() 排序

使用 sort()函数对数组进行排序，例如

var b = [ "Mike", "Jack" , "Bob", "Jenny" ];

b.sort();

会对每个对象从小到大排序。

注：

(1) 在原数组上排序，不创建新的副本

(2) 对任意类型的对象都能排序

reverse() 颠倒顺序

使用 reverse()函数对数组中的所有对象反序排序， 例如

var c = [ "Mike", "Jack" , "Bob", "Jenny" ];

c.reverse();

则变为：Jenny, Bob, Jack, Mike

注：

(1) 在原数组上排序，不创建新的副本

更多用法

Array还可以当链表一样使用(就像Java里的ArrayList)

还可以：

(1) push/pop : 在末尾添加/删除元素

(2) splice : 从中间插入/删除元素

(3) shift / unshift: 从开头删除/添加元素

(4) slice : 得取子列

(5) concat : 合并两个数组

<http://www.w3school.com.cn/jsref/jsref_obj_array.asp>

Chrome设置

调试器里的字体

在Chrome设置里把网页放大即可

小结

了解数组的概念

学会定义数组对象Array，以及数组的基本操 作

3.6 \* JavaScript数组Array (2)

更多用法

Array还可以当链表一样使用(就像Java里的ArrayList)

还可以：

(1) push/pop : 在末尾添加/删除元素

(2) splice : 从中间插入/删除元素

(3) shift / unshift: 从开头删除/添加元素

(4) slice : 得取子列

(5) concat : 合并两个数组

http://www.w3school.com.cn/jsref/jsref\_obj\_array.asp

push, pop

push/pop : 从末尾添加元素, 删除元素

注：

push可以一次添加多个元素

push返回值为链表的新的长度

splice 添加/删除/替换

在链表中间添加、删除、替换元素都是用splice来完成

var b = [10, 20, 30, 40, 50 ];

b.splice(2, 2); // b[2], b[3] 被删除

b.splice(2, 0, 12345, 666); // 在b[2]处插入2个元素

b.splice(2, 1, 777); // 删除b[2], 插入777

参数：前两个参数用于删除，后面的参数起表示插入

返回值：Array ，包含了被删除的元素

shift / unshift

shift : 移出头部第1个元素

var obj = c.shift(); // 返回值为第1个对象

unshift : 在头部添加N个元素

var len = c.unshift(123, 444);

// 返回值为链表的新的长度

slice 子列

slice : 取出一个子列

var d = [10, 20, 30, 40, 50 ];

var sublist = d.slice(1, 4); // d[1],d[2],d[3]

用法

slice(start, end)

取出从start到end下标的元素（不包含end)

concat 合并

concat用于合并N个链表

var e1 = [10, 20];

var e2 = [30, 40, 50 ];

var e = e1.concat(e2);

用法：

result = concat (listX, listY …)

返回值是一个新的Array，原来的链表内容不 变

小结

学习了Array对象的更多操作方法

3.7 JavaScript函数(function)

function 函数

在JavaScript里也可以定义函数，如

function add ( a1, a2 )

{

     return a1 + a2;

}

其中，定义一个函数名为add，带2个参数，有返回值。

函数的调用：

var result = add ( 1, 2 );

注意

（1）函数的参数不需要加 var声明，直接写 就行。

（2）如果需要返回一个对象，那么直接 return就行，不需要声明。

function 作为函数指针

可以把function理解为C++里的函数指针 ；例如，在对数组排序时。。

example-B.html ..

function 的特殊写法

就地写一个函数。。

var data = [ 2, 98, 11, 78 ];

data.sort( function (a, b){

return b - a;

} );

即，相当于一个临时函数，没有具体的函数名。

全局对象与局部对象

函数里创建的对象，为局部对象

函数外的为全局对象

观看 local 与 global 对象。。。

位置与顺序

注意：当函数被执行时，该函数的定义必须 已经被加载。。。

example-D.html

也就是说，函数必须在调用函数之前。

小结

介绍JavaScript里函数的写法

3.8 自定义对象Object (1)

内置对象类型

Number

String

Boolean

Array

Date

RegExp

Math

Global

除了使用它的语法自带的类型，还可以创建自定义的对 象。。。

创建一个对象

例如，创建一个对象表示学生信息，

第1类写法：定义时给定成员名字和初始值

var a = {id: 201601, name:"Ming" };

或给名字叫上双引号

var b = {"id": 201602, "name":"Bo" };

成员放在大括号里，以逗号分隔，名字和值以冒号 分隔。名字可以加双引号。

创建一个对象

第2类写法：创建一个Object，然后再指定成员的值

var c = new Object();

c.id = 201603;

c.name = "Gang";

或

var d = new Object();

d["id"] = 201604;

d["name"] = "Fang";

注：不需要事先声明它有哪些成员，直接加进去就行（相当 于Java里的HashMap)

遍历所有成员

使用 for 可以遍历该对象的所有成员的名字和 值，例如

var obj = {"id": 201602, "name":"Bo" };

for( x in obj )

{

     console.log (x + ": " + obj[x]);

}

其中，x为成员的名字

作为哈希表使用

Object可以作为哈希表使用，因为它内部内存的是键 值对，直接通过名字可以找到一个对象。(相当于Java 里的HashMap类）

var result = students["hua"];

if(result != undefined && result != null)

{

     console.log ("got: " + result.id + ", " + result.name );

}

example-B.html

添加成员函数

可以直接给一个对象添加一个成员函数，

var a = new Object();

a.id = 1;

a.test = function(x)

{

     console.log("example: id " + this.id + ", x" + x );

};

注意：this.id表示访问自己的成员变量id

按如下方式调用该函数：

a.test(123);

小结

介绍了JavaScript里如何创建一个自定义的 Object

obj["id"] 这种访问方式，最能够反映它的本质。

（但是如果想和java语法相呼应的话，还是建议用obj.id ）

3.9 自定义对象 Object (2)

自定义一个类型

JavaScript也可以自己定义一个类型，如

function Student(id, name)

{

     // 成员变量

     this.id = id;

     this.name = name;

}

创建一个对象

var a = new Student(201501, "shaofa");

容易写错的地方

1. 构造函数

function xxx ()

{

}

2. 成员函数

this.yyy = function() {

}

3. 访问成员必须用this. 不可以省略 置换成员函数

置换成员函数（有点像函数的重写）

例如，

a.print = function()

{

     console.log("神奇!" + this.id);

}

a.print();

小结

介绍了如何添加一个自定义的类型

(相当于在Java里写一个class)

3.10 出错处理(try…catch…)

try…catch…

可以捕捉代码块中的错误：语法错误和运行错误

(javascript是解释执行的，所以语法错误也只能在运 行的时候发现）

和Java一样，当错误发生时，try{}里剩余的语句不再 执行。。。

throw

甚至可以自己throw一个异常对象。。。

错误的类型

语法错误：经常发生

不兼容：不同的浏览器对javascript的支持略有差异

自定义异常：人为throw出来的错误

其他

小结

了解JavaScript里也有try…catch…机制

3.11 浏览器对象模型(BOM)

引例

点击一个按钮，让浏览器加载另一网址。。

function example()

{

     window.location.href="http://afanihao.cn";

     //window.open("http://afanihao.cn");

}

window是个什么东西？

BOM

BOM , browser object model 浏览器对象模型

并非标准，而是约定俗成的规则

每一种浏览器在支持JavaScript的时候，都提供一个一些 全局对象来让用户能够控制浏览器窗口：

window

screen

history

location

document

这五个全局对象是所有浏览器都支持的，可以直接用。

window

window对象表示浏览器中打开的窗口

注：window是变量名，Window是类型名

成员变量

window.document

window.location

window.status

…

成员函数

window.alert("haha"); // 弹出一个警告框

window.open("http://afanihao.cn"); // 打开一个新窗口并加载 一个网址

location

location 表示当前的URL

注：

window.location.href = "http://afanihao.cn";

location.href = " http://afanihao.cn ";

这两种写法是一样的。因为 window.location 和location 指向了同一对象。

screen

Screen 屏幕，该对象可以获得用户屏幕的信息，比如长度，高度，分辨率。

History 对象包含用户的访问历史

history.back()

history.forward()

注意

所有的全局变量、全局函数都自动成为window的成 员

window.example()  //自己写的全局函数，全局变量

window.aaa = 123;

兼容性

这几个对象是大多数浏览器都支持的。 （约定俗成，并没有一个国际标准来规定） 另外，像console.log() 里的console对象，并不是

所有的浏览器都支持。IE不支持。 这意味着，如果你在代码里调用了console.log() ，那你的网页在IE可能执行不了。。

演示。。example-C

兼容性

解决办法 1：

function trace ( text )

{

     if( "undefined" != typeof console

     && "undefined" != typeof console.log)

          console.log ( text );

}

这样就可以放心使用这个trace函数 了，如果不兼容的话，就不会执行console.log语句。

兼容性

解决办法 2：try…catch

function trace ( text )

{

     try{

          console.log ( text );

     }catch(e){}

}

可以也放心使用这个trace函数

小节

介绍了一些全局对象，使用这些对象可以和

浏览器打交道：

获取一些值，或者控制浏览器的行为。

常用的：location, document

参考w3school教程：

<http://www.w3school.com.cn/jsref/index.asp>

3.12  JavaScript文档对象模型 (DOM)

比较重要，但是没有ppt，以后补上，先上源码，自己感受下

example-A.html

<html>

<head>

    <!-- 指定本页面的字符集 -->

    <meta charset="GBK">

    <script>

        function example()

        {

            var e1 = document.getElementById("t1");

            e1.innerHTML = "aaaa";

            var e2 = document.getElementById("t2");

            e2.innerHTML = "百度";

            e2.href = "http://www.baidu.com";

        }

    </script>

</head>

<body>

    <button type="button" onclick="example()" > 测试 </button>

    <div>

        <div id="t1">

            ..ss..dd..

        </div>

    </div>

    <div>

        <a id="t2" target="\_blank" href="http://afanihao.cn"> 阿发你好 </a>

    </div>

</body>

</html>

这个是获得对象的id，并根据id更改对象属性

<http://www.w3school.com.cn/jsref/dom_obj_document.asp>

example-B.html

<html>

<head>

    <!-- 指定本页面的字符集 -->

    <meta charset="GBK">

    <style>

        .common

        {

           color: blue;

        }

        .fashion

        {

           color: red;

        }

    </style>

    <script>

        function example()

        {

            var e2 = document.getElementById("t2");

            //e2.className = "fashion";

            e2.style = "color:#AA2300; font-size: 12px; ";

            e2.style.backgroundColor  = "yellow";

        }

    </script>

</head>

<body>

    <button type="button" onclick="example()" > 测试 </button>

    <div>

        <a id="t2" class="common" href="http://afanihao.cn"> 阿发你好 </a>

    </div>

</body>

</html>

根据对象属性（type属性），规定对象样式，并通过style方法改变其样式，也可以用classname属性。style方法下还有一些其他的属性，可以利用。 <http://www.w3school.com.cn/jsref/dom_obj_style.asp>

其中，style的编写样板要记住，函数和style都是用的大括号。

example-D.html

<html>

<head>

    <!-- 指定本页面的字符集 -->

    <meta charset="GBK">

    <script>

        function example()

        {

            var e = document.getElementById("friendLinks");

            e.remove(2);

        }

    </script>

</head>

<body>

    <button type="button" onclick="example()" > 测试 </button>

    友情链接: <select id="friendLinks" name="friendLinks">

        <option value="youku">优酷</option>

        <option value="tudou">土豆</option>

        <option value="qiyi">爱奇艺</option>

    </select>

</body>

</html>

每个控件都有它的属性和方法，研读文档可以掌控一切。

这个例子是select控件，即下拉列表。

example-E.html

<html>

<head>

    <!-- 指定本页面的字符集 -->

    <meta charset="GBK">

    <script>

        function example()

        {

            // 动态创建一个元素

            var opt =document.createElement('option');

            opt.text = "百度";

            opt.value = "baidu";

            // 添加到下拉列表(因为add函数就是这么要求的)

            var sel = document.getElementById("friendLinks");

            sel.add(opt, sel.options[0]);

        }

    </script>

</head>

<body>

    <button type="button" onclick="example()" > 测试 </button>

    友情链接: <select id="friendLinks" name="friendLinks">

        <option value="youku">优酷</option>

        <option value="tudou">土豆</option>

        <option value="qiyi">爱奇艺</option>

    </select>

</body>

</html>

给select添加一个选项，需要先创建一个选项，然后调用其add方法，添加进去。

3.13 事件处理

事件处理

什么叫事件处理？

比如，当用户点一个按钮时，想让它执行一些代 码。。

<button type="button" onclick="example(this)" >

测试

</button>

HTML事件参考

[http://www.w3school.com.cn/tags/html\_ref\_eventattri butes.asp](http://www.w3school.com.cn/tags/html_ref_eventattributes.asp)

主要分为：

Window事件

鼠标事件

键盘事件

表单事件

Window事件

当页面加载完成时，触发onload事件

当页面关闭时，触发onunload事件

例example-B,

<body onload="onPageReady()">

Mouse事件

鼠标事件: onclick, ondblclick, onmouseover…

其实不仅button有鼠标事件，其他元素也都有鼠标 事件

例example-C

Form事件

例example-E

指定回调函数

可以在JavaScript代码里，为某个元素指定回 调函数。

例example-D, (相关语法参考3.8, 3.9节）

var e = document.getElementById("btnTest");

e.onclick = example;

指定回调函数

第1种方法：

<button type="button" onclick="example(this)" > 测试 </button>

这里的example(this);是函数调用语句，其实可以 放多行语句。

第2种方法：

e.onclick = example;

这是成员函数的替换语法，为其指定一个成员函 数 (所以在example的函数定义里能用this)

小节

介绍了事件处理的方法

比较常用的是鼠标事件和onload事件

3.14 JavaScript运行过程

1. 从上往下执行

<script>代码可以分散写，但被认为是同一个程序

2. 以页面为执行单位

不同页面的javascript互相没有影响

可以认为，在页面加载时启动了一个JavaScript程 序。。。

在页面关闭时，该程序所有相关的东西都被销毁了

3. 先后顺序

先后顺序大多数时候没有关系 ，只要在真正执行的时候，所有需要的东西都有定义 就行。（执行时才检查）

例example-B

3.15 JavaScript第三方库

引用第三方库

可以把一些成熟的功能写在单独的文件里，交给别 人使用。

<head>

     <!– 包含第三方JavaScript库 -->

     <script src="example.js"></script>

</head>

常用的库

jQuery： 最常用

ext.js: 可以做出类似桌面UI的效率

4.1 jQuery: 基本使用

jQuery

一个JavaScript的函数库 ，官网 <http://jquery.com/>

主要提供以下功能：

1. 提供了AJAX的实现 (手工与后台交互）

2. 封装了一些函数，方便对DOM元素进行操作 (文 本、属性、事件等)

jQuery

下载 jQuery 1.12，只有一个文件 \*.js

（两种格式：压缩格式，非压缩格式）

在<head>里包含这个js文件

<script src="jquery/jquery.js"> </script>

把下载的\*.js文件改名为jquery.js，并放入jquery文件夹，和项目放在同一级目录下。

一个例子

例example-A.html

function example()

{

     var obj = jQuery ("#target");

     obj.html("一个简单的例子");

}

其中，jQuery是一个函数，返回值是基于目标元素 创建的一个对象。

对比

jQuery写法:

function example()

{

     var obj = jQuery ("#target");

     obj.html("一个简单的例子");

}

普通写法:

function example()

{

     var e = document.getElementById("target");

     e.innerHTML = "一个简单的例子";

}

对比

var obj = jQuery ("#target");

对象obj是一个jQuery函数创建出来的对象，该对象 有很多函数，可以方便我们对元素进行操作。

var e = document.getElementById("target");

对象e是一个DOM对象，可以于直接操纵目标元素 ，显然，obj和e是两个不同的对象，可以认为obj内部 有一个属性(成员)指向了e

简洁的写法

以下两种写法相同：

jQuery("#target").html("hello");

$("#target").html("hello");

也就是说，$ 是一个函数名，作用等同于jQuery

API | Core | jQuery()

小结

引入了jQuery这个库

使用jQuery来操作DOM会方便一些，但是我们直接 使用上一节介绍的基本操作也是可以的。

jQuery主要用于实现AJAX操作，即jQuery.ajax() 这个 函数。

API | Ajax

4.2 jQuery: 操作DOM元素的属性

DOM元素的属性

文档： API | Attributes

五类功能：

1. class ：给元素添加/删除class

<div class="selected highlight" > ssss </div>

2. attr : 元素的其他属性

3. prop: 目标对象的成员变量

4. html: 元素的innerHTML

5. val: 元素的value（针对表单元素）

1. 添加修改class属性

用于实现动态效果

addClass()

removeClass()

toggleClass()

hasClass()

例example-A.html

2. attr()

用于get/set目标元素的属性

<a id="t1" href="http://afanihao.cn" xyz="haha" >

阿发你好 </a>

get属性

var href = $("#t1").attr("href");

set属性

$("#t1").attr("href", "http://www.baidu.com");

注：自定义的属性也可以支持

例 example-B.html

3. html() 操作元素的innerHTML

obj. html () 获取元素的innerHTML

obj. html (" xxx ") 设置元素的innerHTML

例 example-C.html

4. val() 表单控件的值

example-D.html

设置表单控件的值

$("#t1").val("Nobody");

$("#t2").val("us");

获取表单控件的值

console.log("姓名: " + $("#t1").val() );

console.log("国籍: " + $("#t2").val() );

5. prop() 对象的属性

这里的属性指“成员变量”的意思

example-E.html

对checkbox操作时，使用prop函数

小结

介绍了几个常用的函数，来操作目标DOM元素

有两种方法来操作HTML元素：

1. 上一章所说的

2. 使用jQuery API（其对上一章中的方法进行了包装，使之更易使用）

4.3 jQuery: DOM事件处理

DOM事件

文档： API | Events

使用jQuery API，也可以实现事件处理

鼠标事件

键盘事件

表单事件

窗口事件 …

1. 鼠标事件

例example-A.html

例如，click函数用于为目标DOM元素指定onclick回调

$("#t1").click( function ()

{

     location.href = "http://afanihao.cn";

});

下面是指定onmouseover回调, $(this)则是对象本身

$("#t1").mouseover( function ()

{

     $(this).html("go to afanihao.cn");

});

2. 表单事件

例 example-B.html

为下拉列表的onchange事件指定回调

$("#t1").change( function ()

{

     var v = $("#t1").val();

     var url = websites[v];

     window.open(url);

});

3. 文档事件

为document指定onready的回调函数

/\* 页面加载完成 \*/

$(document).ready( function()

{

 // ......

});

另一种更简洁的写法

$(function()

{

     // …….

});

小结

介绍了使用jQuery来设置事件处理函数（回调函数）的方法

4.4 jQuery: 选择器Selector

Selector

使用jQuery() 或 $() ，可以在document中选择指定的 元素。

最基本的几种 (Selectors | Basic)

1. 全部 var objs = $("\*");

2. 按class选择 var objs = $(".fashion");

3. 按tag选择 var objs = $("a");

4. 按id选择 var objs = $("#target");

5. 按attr选择 var objs=$("[name='xyz']");

Selector: 选中N个对象

显然，当使用选择器时，有可能选中了多个DOM元素。

例example-A.html

var objs = $("div");

objs.each( function(i, e)

{

     console.log(i + ":" + e.innerHTML);

});

objs.html("444");

Selector: 综合选择

还有很多种选择元素的方法 ，比如，tag与 class 结合

$("div.fashion")

选择 <div class="fashion">的元素

比如，tag与attribute结合

$("div[code='123']" )

选择 <div code='123' >这样的元素

注：并非重点，不一一演示

选择子项

默认情况下，是在<body>里查找元素 ，也可以指定父Element

例example-B.html，在文档里有多个Form，不同的

Form中有元素的名字相同。。。

$("[name='isbn']", f1).val("9787302419891");

表示: 在f1下查找name='isbn'的元素

小结

介绍了几种选择目标Element的方法

4.5 jQuery: 理解jQuery对象

对比

比较以下代码：

var obj = jQuery("#t1");

和

var e = document.getElementById( "t1" );

我们使用obj和e都可以操纵目标元素，那么, obj和e 之间有什么有关系？

一个类似的实现

自定义一个封装

function af( target )

{

}

然后使用af()来操作目标Element

var obj = af("#t1");

var text = obj.html();

obj.html("afanihao.cn");

注：JavaScript不支持函数名重载

结论

var obj = jQuery ( e )

则obj只是对e进了一个封装

Element -> jQuery Object

可以直接由DOM Element对象来创建

// DOM Element -> jQuery Object

var domObj = document.getElementById("t1");

var $obj = $(domObj);

$obj.html("afanihao");

jQuery Object -> Element

也可以反过来，得到其内部的原始对象

// jQuery Object -> DOM Element

var $obj = $("#t1");

var domObj = $obj.get(0);

domObj.innerHTML += "JAVA";

注意：内部是一个数组

通用对象的封装

其实jQuery并不限制对象是不是DOM对象

对普通对象也是可以封装的

$(document)

$(this)

$(someobj)

例如，example-D.html

具体多看文档，jquery就此结束，以后有大用

4.6 Bootstrap – 基本使用

Bootstrap

由Twitter推出的一个用于前端开发的开源工具包 ，主要是控制WEB页面显示 （美化、特效）

官网: <http://getbootstrap.com/>

中文学习资料：<http://www.bootcss.com/>

引入Bootstrap

由于Bootstrap内部调用了jQuery，所以要在 Bootstrap之前先引入jQuery

<link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">

<script src="jquery/jquery.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

例 表格样式

example-B.html

例 表单样式

example-C.html

小结

引用了Bootstrap,这个库对于前端UI工程师至关重要。

本于本课程来说，引入这个库的主要目的是让演示 界面稍微美观一些。

5.2 创建第一个Web项目

创建 Web Project

File | New | Web Project

目标结构

src: 源代码目录 (java代码)

WebRoot: 输出目录

- WEB-INF : classes, lib

- 网页、图片、js/css

注：此目录结构是tomcat规定好的 （META-INF和WEB-INF）

部署到tomcat

Window | Show View | Servers

在Servers视图下， 部署到tomcat

注：

（1）在tomcat7.x\webapps\下产生一个myweb目录。

（2）WebRoot 与myweb目录是同步的，在WebRoot 下做的修改，会自动拷贝到myweb下

启动tomcat

1. 在Servers视图下选中Tomcat 7.x

2. 点 Run Server按钮

在浏览器中打开

打开网址 http://127.0.0.1:8080/myweb/

其中，127.0.0.1是本地IP地址，8080是tomcat的端口 号。（参考1.4讲）

停止Tomcat

点停止按钮，关闭tomcat

删除部署

remove deployment

将从tomcat的webapps目录下，删除myweb项目目录

小结

如何创建一个Web Project

如何部署到tomcat服务器里

图片没有复制，需要的时候看ppt

5.3 Web应用的几种部署方式

Web Application

每一个Web项目称为一个Web Application

(把WebRoot目录下的文件，称为一个Web App)

网站上已经运行了Tomcat。。。当开发完成后，需 要将WebRoot下的内容部署到网站上。。怎么做？

几种部署方式

1. 直接拷贝到 $TOMCAT\webapps目录下

这就是add deployment菜单做的事情

（直接手工拷贝也是一样的）

2. 打包成\*.war

add deployment | Deploy Type: Packed Archive

（打了个war包放在webapps下，由tomcat解压缩后加载）

过程：tomcat在启动的时候，会解压webapps下的\*.war文件

（可以直接把\*.war -> \*.rar，然后查看里面的东西）

3. Context,虚拟路径

配置$TOMCAT\conf\server.xml

<Host … >

     <Context path ="/myweb"

          docBase="D:\WebProjects\myweb\WebRoot"

          reloadable="true"/>

</Host>

即，直接将WebRoot映射到/myweb这个虚拟路径下

( web app可以放在任意位置，不需要放在tomcat/webapps/下）

(注：修改配置后，要重启tomcat)

4. 其他方式

参考 <http://tomcat.apache.org/>

本课程采用 Context 路径映射 方式

特点：不需要拷贝文件

小结

介绍了几种部署WEB应用的方式

6.1 第一个JSP页面

JSP

JSP, Java Server Page , Java服务端页面 ，后缀为\*.jsp

类似的还有asp, php

创建一个JSP文件

演示：在MyEclipse中添加一个JSP文件。。。

New | JSP

编辑JSP

JSP: 可以认为嵌入了Java代码的HTML页面 添加

<%

     out.println("Hello,World!");

%>

其中，用<% %>来嵌入一段Java代码

在浏览器中打开

运行Tomcat服务器，

在浏览器输入http://127.0.0.1:8080/myweb/test.jsp

JSP的执行过程

图示：见ppt

JSP的执行过程

1. 用 WireShark 演示。。。

参考《WireShark使用教程》

2. 在浏览器中查看网页源代码。。。

看以发现，在浏览器端是看不到 <% %>这一段代码 的。

重要的事实

1. 浏览器接收到的是纯HTML，不含JAVA代码

2. 所有的嵌入代码段 <% %> 均由服务器端先处理和 替换

3. <% %>里面的是Java代码，要符合Java的语法

思考：浏览器能直接打开显示本地文件JSP文件吗？

答：可以，显示的是源代码。

小结

创建了第一个JSP文件。

过程：JSP是在服务器端处理/转换之后，再把转换 后的结果发给浏览器的。

6.2 JSP中的Java代码

JSP中的Java代码

在<% %> 中间可以写Java代码，

例如，

<%

     SimpleDateFormat fmt

          = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");

     Date now = new Date();

     String strNow = fmt.format(now);

     out.print(strNow);

%>

可以得到服务器的时间，并显示出来

import

由于代码里使用了java.text.SimpleDateFormat，所以要import 这个库。

<%@ page import="java.text.\*" %>

<%@ page import="java.io.\*" %>

可以写多行，也一行内用逗号分开，例如

<%@ page import="java.text.\*, java.io.\*" %>

JSP指令

把%@开头的称为JSP指令(directive)

例如，

page指令： <%@ page … %>

include指令： <%@ include ... %>

taglib指令： <%@ taglib ... %>

JSP指令

其中，page指令比较常用，其参数常有：

<%@ page language="java"

import="java.util.\*" pageEncoding="UTF-8"%>

注：将pageEncoding设为UTF-8或GBK，否则无法支持中文

使用自己写的类

1. 在src下面建立一个自己的类 my.Hello

2. 在 jsp 中调用这个类

<%

     my.Hello obj = new my.Hello();

     double result = obj.pow(12.3, 4);

     out.println("result: " + result);

%>

观察：在WebRoot\WEB-INF\classes\发现了什么？

使用out

out是一个对象，该对象的类型为JspWriter

它有哪个函数可以使用？

查看tomcat官方文档。。。

使用out

JRE : Java API

Java EE 6 Libraries: Tomcat 自带API

例 test3.jsp

在自己的类中使用JspWriter

小结

1. 在Jsp中的Java代码可以调用Java API, Tomcat API， 以及自己写的类。

2. 使用page指令来import一个包

3. 自己写的类被编译到 WEB-INF\classes\目录下

6.3 JSP的运行

分散的代码段

在一个JSP页面里，可以有多段<% %>代码

例如，

<%

     SimpleDateFormat fmt

          = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");

     Date now = new Date();

     String strNow = fmt.format(now);

     out.print(strNow);

%>

它们可以分散来写，穿插在<html>里面

JSP的处理

（1）从上到下依次执行

（2）多段代码就可以像一个函数里的多个语句

（3）对象out, request, response从哪来的？就如同函数的参数。。

可以想象，tomcat在处理JSP时，也许会自动创建一个对象

class MyServlet

{

     void handleRequest( JspWriter out)

     {

          // 把JSP的代码抽出来, 放在这里逐行执行

     }

}

JSP表达式

以下两种写法作用相同：

<p> 今天是 <% out.print(strNow); %> ，欢迎登录！ </p>

<p> 今天是 <%= strNow %> ，欢迎登录！ </p>

当需要输出一个表达式的值时，可以采用<%= %>简化写法

注意：JSP表达式里面不加分号

优点：可读性较好， 把计算和显示分开。

小结

进一步理解JSP的运行原理

6.4 表单查询

表单查询

应用场景：

（1）用户打开网站，在浏览器输入QQ号，点提交

（2）后台检查此QQ号对应的信息，显示给用户。

前端：设计一个表单

由前端的UI工程师，制作一个网页，里面有一个表单

<form method="get" action="query.jsp" >

输入QQ号

<input type="text" name="qq" />

<input type="submit" value="提交" />

</form>

注意：form必须添加2个属性：method和action

method: "get" / "post"，这里先使用"get"

action: 指定由谁来处理这个请求 query.jsp

2. 按钮的type为"submit"，点此按钮后，由浏览器 发起请求，将参数传给后台。

3. 文本框须设置name属性，此name就是参数的名 称。

注意观察浏览器中的URL

后台：JSP处理代码

新建 query.jsp，在里面嵌入JAVA代码，处理用户的请求

1. 获取用户的请求参数

String qq = request.getParameter("qq");

2. 查询数据库

暂时没有真实的数据库，用DataSource类模拟

DataSource db = new DataSource();

DataSource.Record e = db.query(qq);

3. 显示结果

out.write (…)

观察URL

query.jsp?qq=1323

query.jsp: action

qq=123 : qq参数名，123参数的值, 即用户的输入

--------------

多个参数

qq=123&name=shao

抓包演示

1. 打开wireshark，开始抓包

2. 浏览器打开

http://afanihao.cn/test/index.jsp

3. 输入qq号

点提交

4. 结束抓包

（保存抓包结果）

本节重点

1. 理解整个前后台处理流程

2. 明白参数名name的作用

3. 明白后台JAVA代码如何获取请求里的参数

6.5 中文URL编码

背景知识: GBK / UTF-8

字符集与中文编码：

《Java学习指南》基础篇， 35，36，37讲

例如，“中国”

GBK: D6 D0 B9 FA

UTF-8 : E4 B8 AD E5 9B BD

问题演示

演示：

1. 输入纯英文字母

2. 输入中文参数

前端：URL编码

浏览器自动会把form中的各字段编到URL中

例如，

query.jsp?qq=1234&name=li

规则：

(1)追加?号

(2)key=value

(3)每对参数以&分开

如果用户输入了中文怎么办？

URL编码结果：

query.jsp?name=%E4%B8%AD%E5%9B%BD&qq=1234

注：浏览器地址栏虽然显示的中文、但实际发送给 服务器的是编码后的URL… 通过抓包可以看清楚。。。

后台：接收参数

接收参数

String name = request.getParameter("name");

接收下来的字符串name是一个乱码。。

调试。。。

原因：tomcat把%E4%B8%AD%E5%9B%BD转成了6个 字符

00E4 00B8 00AD 00E5 009B 00BD

解决转码问题：

// 1. 接收参数

String name = request.getParameter("qq");

// 2. 得到 6个字节

byte[] bb = qq.getBytes("iso-8859-1");

// 3. 构造字符串

name = new String(bb, "UTF-8");

小结

1. 了解浏览器的URL编码规则

2. 了解中文参数如何编码到URL

3. 掌握从request中取出参数值、并解决乱码问题的方法

6.6 两种方式: GET与POST

GET/POST

<form>的method属性有两个可选的值：

GET / POST

如下形式：

<form method="post" action="query.jsp" >

     …

</form>

它是用来指定浏览器以何种方式将数据提交到服务 器的。

GET方式

在HTTP请求时，浏览器将各表单域的值附在URL后 面（即，放在HTTP头部）。。

例如

GET query.jsp?name=%E4%B8%AD%E5%9B%BD&qq=1234

POST方式

浏览器发起HTTP请求时，将各参数的值放在HTTP body中

POST /test/query.jsp HTTP/1.1

Host: afanihao.cn

name=%E9%82%B5%E5%8F%91&qq=1234

什么时候需要POST方式？

有些时候不适合使用GET方式，例如，

（1）密码不适合显示在URL中

（2）表单域非常多时，不适合放在URL中

（3）表单域比较复杂时，不能放在URL中。。

用途

GET: 用于从服务器GET信息（查询类操作、读操 作），不会影响服务器原有的数据。

POST: 用于把信息上传到服务器（上传类操作、写操 作），会增加、删除、修改服务器上的数据。

注：GET方式用于查询，其实可以直接构造一个URL， 把参数放在后面。。。演示。。。

POST下的中文处理

两种处理乱码的办法：

1. 手工转码

String name = request.getParameter("qq");

name = new String(name.getBytes("iso-8859-1"); , "UTF-8");

2. 自动转码：设置request、让它自动按UTF-8解码

request.setCharacterEncoding("UTF-8");

(注：自动转码仅适用于POST方式)

小结

1. GET 与 POST的区别

2. POST时如何处理中文乱码

3. 什么时候用GET，什么时候用POST

6.7 HTTP会话 (1)

HTTP 会话

HTTP Session, 会话

指的是“从打开网站第一个页面、到关闭浏览器” 期间的所有访问。

例如，在浏览器打开http://afanihao.cn/index.jsp，然 看浏览A.jsp, B.jsp, C.jsp …，最后，关闭浏览器。

演示

某网站有3个页面:

index.jsp : 首页

a.jsp : 电影

b.jsp : 音乐

用以下代码获取Session对象，并显示其ID

<%

     HttpSession hs = request.getSession();

     out.print("当前会话: " + hs.getId());

%>

注意观察：

1. 会话期间，Session ID不变

2. 会话什么时候结束？浏览器关闭的时候，会话结 束。

3. 不同的浏览器，彼此没有影响。例如，用IE打开 是一个会话，用Chrome打开是另一个会话。

小结

当用户在浏览器上打开网站页面时，tomcat服务器 创建一个会话Session对象，用来保存一次会话的信 息。

借助于HttpSession对象，我们可以保存相关信息。。。例如， 当前登录用户。。。

6.8 HTTP会话 (2)

HttpSession对象

获取会话对象

HttpSession hs = request.getSession();

存入一个属性

hs.setAttribute("user", "shaofa");

取得一个属性的值

String user = (String)hs.getAttribute("user");

用Session实现用户登录

假设一个网站：

1. index.jsp "首页",无论有没有登录，均可以观看

2. account.jsp "我的帐户"设置页，只有登录后可见

3. login.jsp 用户登录

4. logout.jsp 注销用户

安全性要求

1. account.jsp

本页将显示用户的私密信息，必须保证登录后可见

2. 自动注销

若用户没有点"注销", 则要保证在关闭浏览器/系统

后，能自动注销。

(保证别人在使用该电脑时，不会看到刚才的用户信 息）

技术细节

（1）HttpSession对象用于保存会话信息

（2）response.sendRedirect()可以令客户端浏览器跳

转到某个页面

小结

认识到Session的作用：保存本次会话相关的信息，

当浏览器关闭时，Session自动销毁。

个人观点：主要是Session的应用，实现网站的登录，注销，示例源码中有相关代码。

6.9 页面重定向(Redirect)

Redirect页面重定向

使用 response.sendRedirect() 可以令浏览器跳转到另 一页面。

此过程称为：页面重定向，Redirect

演示

1. 普通打开

http://127.0.0.1:8080/a.jsp

显示的是a.jsp里的内容

2. 重定向

http://127.0.0.1:8080/a.jsp?jump=yes

显示的是b.jsp的内容

代码实现

在a.jsp里插入代码

<%

     String jump = request.getParameter("jump");

     if( "yes".equals(jump) )

     {

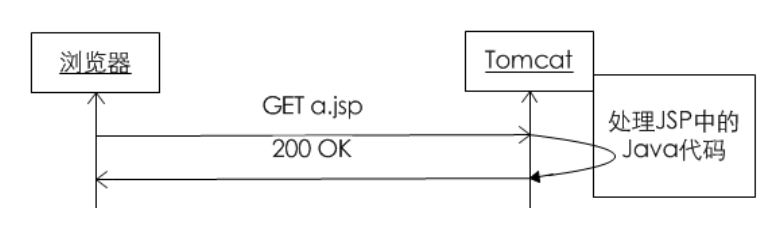
          response.sendRedirect("b.jsp");

     }

%>

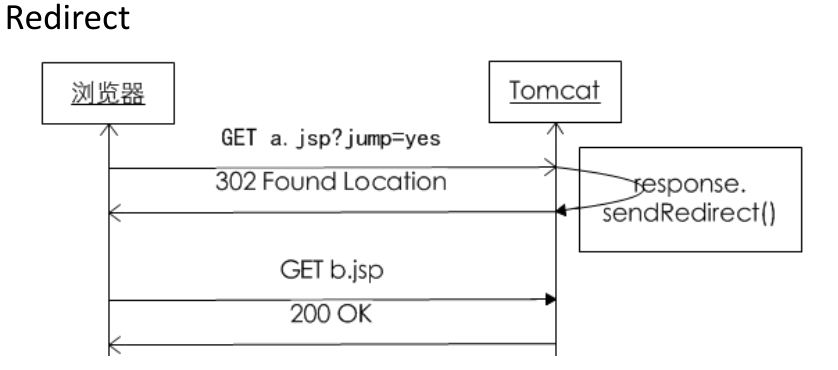
HTTP交互原理

普通方式



抓包演示。。。

HTTP交互原理



抓包演示。。。

小结

当调用response.sendRedirect()时，返回给浏览器的 是302消息，浏览器接着会请求目标页面。

7.1 第一个Servlet

**Servlet**（Server Applet），全称**Java Servlet**，暂无中文译文。是用Java编写的服务器端程序。其主要功能在于交互式地浏览和修改数据，生成动态Web内容。狭义的Servlet是指Java语言实现的一个接口，广义的Servlet是指任何实现了这个Servlet接口的类，一般情况下，人们将Servlet理解为后者。

（以后有什么不理解的，参见百度百科）

添加Servlet

演示。。

在MyEclipse中添加Servlet, 观察：

1. 添加了一个servlet类 HelloServlet      (New->Servlet)

2. 在web.xml中添加了相关配置      (Auto add)

演示

访问Servlet

http://127.0.0.1:8080/myweb/servlet/HelloServlet

观察：浏览器的输出显示

修改HelloServlet

在doGet（）函数里 (代替原先的代码)

response.setContentType("text/plain");

response.setCharacterEncoding("UTF-8");

PrintWriter out = response.getWriter();

out.write("Hello,World!");

out.close();

重启Tomcat，在浏览器访问此Servlet。。。

HTTP交互原理

观察Wireshark抓包。。。

修改web.xml

观察web.xml中的配置项 （MyEclipse是自动加上的，但手工 写也是可以的）

<servlet>

<servlet-name>HelloServlet</servlet-name>

<servlet-class>my.HelloServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>HelloServlet</servlet-name>

<url-pattern>/servlet/HelloServlet</url-pattern>

</servlet-mapping>

后台处理流程

后台如何匹配客户端的请求：

（1）接收浏览器的请求，访问/servlet/HelloServlet

（2）在<servlet-mapping>里查找，确定由HelloServlet来处理

（3）从<servlet>里查找，确定对应的类是my.HelloServlet

注意：完整的url是怎么拼凑的 … （<url-pattern> 中可以设置url）

删除Servlet

1. 删除HelloServlet类

2. 删除WEB-INF\classes\下的HelloServlet.class文件 （Auto delete）

3. 在web.xml里删除相关配置（否则tomcat在启动时会提示 错误) (ps:实测，启动时不会，访问被删除的网页会报错)

要删干净。。。

小结

介绍了如何创建一个Servlet:

(1) 创建一个Java类

class HelloServlet extends HttpServlet{}

(2) 在web.xml中添加相关配置

<servlet> … </servlet>

<servlet-mapping> … </servlet-mapping>

7.2 Servlet的更多细节

1. doGet 与 doPost

当请求为GET方式时，doGet被调用

当请求为POST方式时，doPost被调用

2. init与destroy

init() : 初始化，当第一次请求到来时，创建 Servlet 对象，并调用init()初始化。

destroy(): 当tomcat退出时被调用

注：HelloServlet只创建一个实例，init()只调用一次 ，所以，Servlet类里一般不保存成员变量，如果有成 员变量，也是用来保存一些全局的公共配置。

3. setContentType()

Content-Type: HTTP正文内容的类型

常用的用：

text/plain

text/xml

text/json

text/html

image/jpeg

image/png

image/gif

application/octet-stream (任意类型的文件)

4. 中文编码

request.setCharacterEncoding("UTF-8");

response.setCharacterEncoding("UTF-8");

5. Input与Output

也可以直接控制InputStream和OutputStream

InputStream streamIn = request.getInputStream();

OutputStream streamOut = response.getOutputStream();

比如，

InputStream streamIn = request.getInputStream();

byte[] buf = new byte[1024];

int n = streamIn.read(buf);

String submitText = new String(buf, 0, n, "UTF-8");

System.out.println("text: " + submitText);

5. Input与Output

两种方式不可以混用

（1）request.getParameter()获取参数

（2）request.getInputStream()

输出也是一样，不能混用

（1）out = response.getWriter()

（2）streamOut = response.getOutputStream();

6. jsp里的request和response

request: HttpServletRequest

response: HttpServletResponse

小结

介绍了Servlet的更多细节

7.3  Servlet实现文件下载

这一节没有ppt，只能在源码上分析，还是很有收获的

下面只列举了HelloServlet类的部分代码，包括构造函数，destroy，doGet方法，主要是doGet方法。

public class HelloServlet extends HttpServlet

{

    /\*\*

     \* Constructor of the object.

     \*/

    public HelloServlet()

    {

        super();     //ps：调用父类的构造函数

        System.out.println("constructor ...");

    }

    /\*\*

     \* Destruction of the servlet. <br>

     \*/

    public void destroy()

    {

        super.destroy(); // Just puts "destroy" string in log ，结束的时候自动调用。

        // Put your code here，不是应该顺序执行吗？放在这儿还会被执行 ，ps：貌似是先调用父类的destroy函数，然后在销毁自己的东西。

    }

    /\*\*

     \* The doGet method of the servlet. <br>

     \*

     \* This method is called when a form has its tag value method equals to get.

     \*

     \* @param request the request send by the client to the server

     \* @param response the response send by the server to the client

     \* @throws ServletException if an error occurred

     \* @throws IOException if an error occurred

     \*/

    public void doGet(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException

    {

        //参数

        String id = request.getParameter("id");

        /\* request.getParameter()方法传递的数据，会从Web客户端传到Web服务器端，

         代表HTTP请求数据。request.getParameter()方法返回String类型的数据。\*/

        String filename = id + ".jpg";

        response.setContentType("image/jpeg");//告诉浏览器，这是一个jpeg格式的图片；然后浏览器会采取相应的措施

        response.setCharacterEncoding("UTF-8");//设置字符集

        //数据源

        File imageDir = new File("c:/images");

        File imageFile = new File(imageDir,filename);

        FileInputStream fis = new FileInputStream(imageFile);

        //response.setContentLength((int)imageFile.length());

        //输出

        OutputStream streamOut = response.getOutputStream();

        byte[] buf = new byte[4096];

        while(true)

        {

            int n = fis.read(buf);

            if(n<=0) break;//如果没有读取到文件，则退出

            streamOut.write(buf,0,n);

            streamOut.flush();//可选，trucked输出

            /\*Response.Flush()的作用是将缓冲信息输出到页面。

            比如我们在点击一个按钮后，执行多个任务，每个任务执行成功后都有一个Response.Write("成功信息") 。

            如果我们在Response.Write('成功信息')后再加上Response.Flush(),产生的效果就是每个任务执行完

            之后都会直接输出信息到页面。

            如果没有Response.Flush()就需要等所有任务都执行完毕之后才将所有的Response.Write("信息") 一起输出。

             \*/

        }

        //关闭

        fis.close();

        streamOut.close();

        /\*

        垃圾回收机制仅在Java虚拟机所控制的范围内释放资源。

        对于类似于数据库连接、socket以及文件操作等，

        如果有close方法，在你完成任务后执行它 \*/

    }

7.4 HTTP chunked编码

Content-Length

通常情况下，我们在发送应答数据之前，应该先设置一个内 容长度Content-Length

// 转成字节

String content = "Hello,World!";

byte[] data = content.getBytes("UTF-8");

// 数据长度, 以字节为单位

response.setContentLength(data.length);

// 发送数据

OutputStream out = response.getOutputStream();

out.write(data);

out.close();

则实际发送的HTTP是这样的：(在wireshark中观察)

HTTP/1.1 200 OK\r\n

Server: Apache-Coyote/1.\r\n

Content-Type: text/plain;charset=UTF-8\r\n

Content-Length: 12\r\n

Date: Tue, 29 Mar 2016 06:24:56 GMT\r\n

\r\n

Hello,World!

chunked方式编码      （ps：分块发送）

也可以不设置Content-Length，使用chunked编码

OutputStream out = response.getOutputStream();

byte[] data1 = "Hello,World!".getBytes("UTF-8");

out.write(data1);

out.flush(); // 第1块

byte[] data2 = "Yes, it's java!".getBytes("UTF-8");

out.write(data2);

out.flush(); // 第2块

out.close();

chunked方式编码

HTTP应答的内容格式为：(在wireshark中观察)

HTTP/1.1 200 OK \r\n    （ps：200 OK：表示发送成功）

Server: Apache-Coyote/1.1\r\n

Content-Type: text/plain;charset=UTF-8\r\n      （ps：内容类型，字符集编码）

Transfer-Encoding: chunked\r\n      （ps：传输编码：分块的）

Date: Tue, 29 Mar 2016 06:30:11 GMT\r\n

\r\n

b\r\n     （ps：先发送字节的个数，b个，也就是十进制的11个）

Hello,World

f\r\n      (ps：f：15个字节)

Yes, it's java!

0\r\n     （ps：0个字节）

\r\n

比较

1. Content-Length比较简单， chunked较复杂 （ps：是在客户端解读的时候比较复杂）

2. 如果能预知长度，则使用Content-Length；

如果事先不知道数据总长度，则使用chunked

小结

介绍了两种编码方式

• 一般尽量使用Content-Length

• 如果不理解chunked方式，则使用Content-Length

注：在后面 AJAX / RESTful 技术中要注意这个问题

使用PrintWriter      （ps：和之前的两种发送方式一样可以用来给客户端发送数据）

在处理文本时，我们经常使用的是PrintWriter对象

例如，

PrintWriter out = response.getWriter();

out.write("Hello,World!");

out.write("Yes,it's java!");

out.close();

但是，它到底是按Content-Length还是按chunked方 式编码呢？

使用PrintWriter

1. 如果显式的设置了Content-Length，则采用

Content-Length编码 (超出的内容被截断)

response.setContentType("text/plain"); //

response.setCharacterEncoding("UTF-8");

response.setContentLength(8);

PrintWriter out = response.getWriter();

out.write("Hello,World!");

out.close();

结果：实际发送了8个字节，按Content-Length方式

2.

\* 没有设置Content-Length

\* 调用了out. flush()

则每一次flush()都发送一个chunk

response.setContentType("text/plain");

response.setCharacterEncoding("UTF-8");

PrintWriter out = response.getWriter();

out.write("Hello,World!");

out.flush(); // 第1块

out.write("Yes,it's java!");

out.flush(); // 第2块

out.close();

3. 既没有Content-Length，又没显式的调用flush()，则当内部 缓冲区满的时候，会自动flush()

String content = "………… ";

PrintWriter out = response.getWriter();

out.write(content);

out.close();

也就是说，如果content较短，是按Content-Length发送; 如果 content较长，则以chunked方式发送。

ps：能够使用Content-Length方式就用，不能用再用chunked方式，尽量不要用PrintWrite方式，因为不可控

8.1 AJAX - 基本使用方法

与服务器交互的方法

1. 在浏览器里输入URL

原理：由浏览器发起一个HTTP GET请求

2. 在网页里提交表单

原理：由浏览器发起一个HTTP GET/POST请求

3. 在JavaScript里使用AJAX方法

原理：使用 jQuery.ajax()函数，令浏览器发起一个HTTP GET/POST请 求

4. 用Java代码完成HTTP请求

使用Java的API，完成HTTP请求 (可使用apache http-component)

AJAX方法

AJAX, [ˈeˌdʒæks]

Asynchronous Javascript And XML

一种异步更新的方法：指的是在不刷新整个网页的 情况，与服务器交互数据，对网页的某个部分做局 部更新。 （可以实现交互式网页界面）

实现方法

使用jQuery提供的函数来实现AJAX交互

1. 引入jQuery脚本

2. 使用jQuery.ajax()函数

jQuery.ajax()

使用 jQuery.ajax()可实现AJAX交互 （也可以写成$.ajax() )

jQuery.ajax ( {

     method: "GET",

     url: "servlet/HelloServlet",

     success: function(data, textStatus, jqXHR)

     {

          $("#userCount").html(data);

     },

     error: function( jqXHR, textStatus, errorThrown){

          trace( "error: " + errorThrown );

     }

});

jQuery.ajax()

jQuery.ajax(settings) 参数说明

参数settings对象是一个Object（可以认为是一个结 构体），有以下属性：

- method: "GET" 或 "POST"

- url: 字符串

- success: 回调函数, AJAX交互成功；其中，回调函数中的data是url的返回值，可以是各种形式。

- error: 回调函数，AJAX交互失败

与传统网页比较

传统网页：只有刷新全部页面，才可以内容显示 (演示： old.jsp)

AJAX: 整体不变，局部更新

小结

介绍了AJAX技术的基本实现方法 jQuery.ajax()

介绍了AJAX技术与传统网页刷新的区别

结论：使用AJAX技术，可以实现交互式页面  (效果就像桌面GUI程序一样）

eg：我的基础很不好，很多简单的应用写不出来

8.2 AJAX – 与定时器结合使用

目标效果

上面实时显示在线人数 (jQuery Timer + AJAX)

下面播放视频 （HTML5 <video>）

演示：。。。

使用定时器

jquery.timer.js

一个基于jQuery的脚本库

例如，

     $(document).ready( function(){

          // 启动定义器

          var timerId = $.timer(1000, onQuery, false);

     });

则启动一个定时器，每1000毫秒执行一次onQuery函数

与传统网页比较

传统网页：只有刷新全部页面，才可以内容显示 (演示： old.jsp)

AJAX: 整体不变，局部更新

小结

整体不变，局部刷新，增加页面的交互性

例子中使用了<vidio>标签，同时介绍了定时器的使用，包括需要的组件，组件的引入。

具体使用的时候，见源代码。

java web其实就是一级一级的调用，网页上的控件调用函数，函数调用sevlet，等等。。

8.3 AJAX – GET与POST数据传递

上传数据

例如，用户在文本框里输入ISBN, Keyword， 点“检索”，查询相关图书，显示结果

怎么把用户输入的值通过AJAX传到后台呢？

上传数据

通过设置data字段，来把用户的输入传递上去

jQuery.ajax({

     method: "GET",

     url: "servlet/HelloServlet",

     // 要上传的数据

     data: {

          "isbn" : isbn,

          "keyword" : keyword

     } ,

     。。。

});

data字段指定一个Object

GET与POST

GET: AJAX构造一个GET请求，url末尾附加上参数 servlet/HelloServlet?isbn=123456&keyword=java

POST: AJAX构造一个POST请求，在HTTP body中传递 参数值

isbn=123456&keyword=java

其实就是和<form>的效果是一样的，参考抓包文件

服务器返回数据

当Content-Type设置为 text/plain时，

返回

success: function(data, textStatus, jqXHR)

{

     $("#result").html(data);

}

其中，data为服务器返回的数据，为String类型对象

注：如果不是text/plain，则data可能不是String类型

小结

介绍了AJAX交互方式下如何传递数据

8.4 AJAX - JSON数据格式(1)

应用场景

在页面上输入一个ISBN点，点“检索”

后台检索得到一本书的详细信息：

title, author, isbn, publisher

数据格式

服务器以何种格式返回数据，是核心问题

例如，服务器可以按如下格式返回

title=xxx&author=yyy&isbn=zzz&publisher=www

这对服务器并不困难，直接用Java代码就可以拼凑 出来。但是，对于客户端来说，解析这段文本却很 困难！！

数据格式

格式1：Java代码

replyText = "title=" + title

+ "&" + "author=" + author

+ "&" + "isbn=" + isbn

+ "&" + "publisher=" + publisher;

所得文本的格式为

title=xxx&author=yyy&isbn=zzz&publisher=www

JSON 数据格式

JSON , /ˈdʒeɪsən/ JavaScript Object Notation ，一种文本化表示的JavaScript Object

其格式为：

{

"title":"xxx",

"author":"yyy",

"isbn":"zzz",

"publisher":"www"

}

其实就是定义一个Object的JavaScript代码样式 （第3章）

如果JSON格式的文本传给浏览器，那么浏览器端能很方 便地解出数据。

var result = jQuery.parseJSON(data);

result就是一个Java Script对象，其属性成员可以直接访 问了：

result.title

result.author

result.isbn

result.publisherr

用Java代码拼凑JSON格式的字符串：

replyText =

"{"

+ " \"title\": " + "\"" + title + "\"" +","

+ " \"author\": " + "\"" + author + "\"" +","

+ " \"isbn\": " + "\"" + isbn + "\"" +","

+ " \"publisher\": " + "\"" + publisher +"\""

+ "}";

（注：以后将使用API来完成）

前台与后台联调

完成所有代码，前台与后台联调通过

小结

1.引入了JSON数据格式

2. 以JSON格式返回数据时，前端的解析将十分便捷

JSON格式：就是用JavaScript定义对象的代码格式

index.jsp的部分代码，可以看到onQuery函数及其调用：

function onQuery()

    {

        var isbn = $("#isbn").val();

        // ajax request

        jQuery.ajax({

            method: "GET",

            url: "servlet/HelloServlet",

            // 要上传的数据

            data: {

                "isbn" : isbn

            },

            success: function(data, textStatus, jqXHR)

            {

                trace(data);

                var result = jQuery.parseJSON(data);

                showResult(result);

            },

            error: function( jqXHR, textStatus, errorThrown){

                $("#result").html("error: " + errorThrown );

            }

        });

    }

    // 显示到表格里

    function showResult ( result )

    {

        $("#title").html( result.title );

        $("#author").html( result.author );

        $("#isbn").html( result.isbn );

        $("#publisher").html( result.publisher );

    }

    // 页面加载后的初始化工作

    $(document).ready( function(){

    });

    </script>

  </head>

  <body>

   <div class="container">

   <!-- 查询 -->

       <div class="panel">

              <br>

         ISBN <input id="isbn" type="text" >

        <input type="button" value="检索" onclick="onQuery()" >

        <br>

       </div>

       <!-- 结果 -->

       <table id="result"  class="table" style="width:400;" >

           <tr> <th> 书名 </th> <td id="title"> xxxx </td> </tr>

           <tr> <th> 作者 </th> <td id="author"> xxxx </td> </tr>

           <tr> <th> ISBN </th> <td id="isbn"> xxxx </td> </tr>

           <tr> <th> 出版社 </th> <td id="publisher"> xxxx </td> </tr>

       </table>

   </div>

  </body>

HelloServlet.java的部分代码，可以看到JSON这种数据函数的构建，但是以后应该不用这么麻烦。

public void doGet(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException

    {

        String isbn = request.getParameter("isbn");

        String title = "C/C++学习指南";

        String author= "邵发";

        String publisher = "清华大学出版社";

        // 将title, author, press, isbn发给客户端

        String replyText = "";

//        replyText = "title=" + title

//                + "&" + "author=" + author

//                + "&" + "isbn=" + isbn

//                + "&" + "publisher=" + publisher;

        // JSON

        replyText =

                "{"

                + "   \"title\": " + "\"" + title + "\"" +","

                + "   \"author\": " + "\"" + author + "\"" +","

                + "   \"isbn\": " + "\"" +   isbn + "\"" +","

                + "   \"publisher\": " + "\"" +  publisher +"\""

                + "}";

        // 应答

        response.setContentType("text/plain");

        response.setCharacterEncoding("UTF-8");

        PrintWriter out = response.getWriter();

        out.write(replyText);

        out.close();

    }

8.5 AJAX - JSON数据格式(2)

JSON API

显然，手工拼写一个JSON字符串比较麻烦，容易出错，需要 一个API来完成。

JSON官网 http://www.json.org/

下载JAVA版本的解析器 ，下载得到是源代码形式 ，我们可以直接使用源代码， 或者自己打包成jar包 json-org.jar

注：能操作JSON的API不止一个，还有其他库也支持JSON。。

使用JSON API

把json-org.jar拷贝到 WEB-INF\lib\目录下

注：在MyEclipse界面下粘贴，会自动把jar配置到到 Build-Path下

下面，使用org.json.\*提供的API来改编上节课的代码

JSONObject jsReply = new JSONObject();

jsReply.put("title", title);

jsReply.put("author", author);

jsReply.put("isbn", isbn);

jsReply.put("publisher", publisher);

String replyText = jsReply.toString();

是不是很容易呢？

JSON的构造

测试1

JSON中，除了字符串值需要双引号，其他的Integer,

Boolean, Double型都不需要双引号

{

"id":123,

"male":true,

"name":"邵发",

"avg":98.5

}

测试2 : Object内部元素也可以是Object

{

"title":"学习指南",

"audio":

{

"sampleRate":44100,

"channel":2

},

"video":

{

"height":720,

"width":1280

}

}

注意

1. JSONObject的属性不分前后顺序

2. 构造的时候，不会有异常

测试3 : Array

[90,91,92]

JSON的构造

测试4: Array的元素是 Object

[

{"name":"Li","score":91},

{"name":"Li","score":91},

{"name":"Li","score":91}

]

测试5: Object的属性可以是Array

{

"success":true

"reason":"OK",

"data":

[

{"name":"Li","score":91},

{"name":"Li","score":91},

{"name":"Li","score":91}

]

}

测试6: 可以把一个Map转成JSONObject

Map values = new HashMap();

values.put("id", 123);

values.put("name", "shaofa");

JSONObject result = new JSONObject(values);

JSON的解析

已经有一个JSON格式的字符串，可以转成JSONObject或 者JSONArray

// 如果str格式不规范, 会抛异常

JSONObject mp4 = new JSONObject(str);

// 若相应字段不存在，会抛异常

String title = mp4.getString("title");

String author = mp4.getString("author");

// 若字段类型不匹配, 会抛异常

小结

介绍了如何使用JSON API来构造/解析JSON格式的字 符串

ps：其实就是别人写了一个包，然后我直接拿来用，这其中有很多函数，可以去了解。

另，JSON的包作者已经打好包了，可以直接去项目中复制，也可以下载源码，然后自己建一个包放进去，视频中有方法。

8.6 AJAX – 上行数据的格式

默认的上行数据格式

在使用GET/POST时，默认地，jQuery.ajax()会把数据转成

k1=v1&k2=v2&…的格式

例如，

jQuery.ajax({

     method: "POST",

     url: "servlet/HelloServlet",

     data: jsReq,

     。。。

})

则会自动POST如下形式的字符串：

id=201601&name=%E9%82%B5%E5%8F%91&phone=13800012345

JSON格式的上行数据

可以令jQuery上传一段JSON格式的数据，方法为：

1. 设 processData 属性为false

2. 设data为一段JSON格式的字符串

怎么把一个JavaScript对象转成JSON字符串呢？

使用jquery.json.js API

<script src="jquery/jquery.js"></script>

<script src="jquery/jquery.json-2.3.js"></script>

然后就可以转换了：

var jsonText = jQuery.toJSON(jsReq);

JSON格式的上行数据

前端代码：

jQuery.ajax({

     method: "POST",

     url: "servlet/HelloServlet",

     processData : false,

     data: jQuery.toJSON(jsReq),

     …

});

JSON格式的上行数据

服务器端代码：

由于客户端上传的文本是JSON格式，所以不能再用 getParameter()来获取参数了，改成：

InputStream streamIn = request.getInputStream();

String userText = readAsText(streamIn, "UTF-8");

JSONObject jreq = new JSONObject(userText);

String reqId = jreq.getString("id");

String reqName = jreq.getString("name");

String reqPhone = jreq.getString("phone");

小结

介绍了一种方法，可以把数据以JSON格式上传

注：当数据较为复杂时，应使用这种方式 （数组，链表等）

(POST时用，get没法用)

8.7 AJAX – 复杂数据的上传

上行下行数据的格式

上行: 普通格式 或 JSON格式

下行: JSON格式

下面这段代码就是复杂数据使用JSON格式上传的方式，和上一节中没有什么差别。个人猜测，jQuery会把符合JSON格式的数据按照一定格式自动转换，我们直接使用其封装好的函数就好了。

    function onSubmit()

    {

        // ajax request

        jQuery.ajax({

            method: "POST",

            url: "servlet/HelloServlet",

            // 上行数据

            processData: false,

            data: $.toJSON(rows),

            success: function(data, textStatus, jqXHR)

            {

                var result = jQuery.parseJSON(data);

                showResult(result);

            },

            error: function( jqXHR, textStatus, errorThrown){

                $("#result").html("error: " + errorThrown );

            }

        });

    }

小结

当要上传的数据比较复杂时，先构造一个JavaScript 对象，然后以JSON格式上传。

8.8 AJAX – 其他细节

指定格式: dataType

dataType: "json"

告诉jQuery：服务器发来的数据是JSON文本，请自动转 成一个对象

dataType: "json",

success: function(data, textStatus, jqXHR)

{

//var result = jQuery.parseJSON(data);

//data: 已经转成了Object

},

指定格式: dataType

如果确定文本的格式是JSON

显示地加上：

dataType: "json",

如果不指定dataType

则jQuery会根据Content-Type，自动判断

1. 如果是text/plain，data就是String

2. 如果是text/json，data就自动地转成了Object

简化版ajax()

Shorthand Methods:

jQuery.get()

jQuery.post()

jQuery.getJSON()

小结

有空可以仔细研究jQuery的说明文档

9.1 RESTful - 在App里访问HTTP服务

在app里访问服务器

Servlet所提供的服务，不仅在浏览器里可以访问，

也可以在其他任意app里通过HTTP访问。

在app里访问服务器

服务URL：  http://127.0.0.1:8080/myweb/servlet/HelloServlet

在任意APP里，都可以通过HTTP来访问这个服务。

APP: >>> HTTP请求 >>> Tomcat

APP: <<< HTTP应答 <<< Tomcat

Apache HttpComponents

Java的App项目里，可以使用apache.org下的一个库 来完成HTTP交互。

Apache HttpComponents :

http://www.apache.org/

Apache下提供了很多开源库/框架，称为Apache Projects

在下面找到HttpComponents，其提供了HTTP的支持

下载官方文档PDF， 及软件包。。。

Apache HttpComponents

它提供了两个模块：

HttpCore: HTTP服务器应用

HttpClient: HTTP客户端应用

本课程只使用HttpClient

演示：在Java项目中添加HttpClient库。。。

HTTP GET 交互

CloseableHttpClient httpclient = HttpClients.createDefault();

HttpGet httpget = new HttpGet(url);

CloseableHttpResponse response = httpclient.execute(httpget);

try {

     HttpEntity entity = response.getEntity();

     if (entity != null)

     {

          long len = entity.getContentLength();

          if (len != -1 && len < 16384)

          {

               String replyText = EntityUtils.toString(entity);

               return replyText;

          } else

          {

               // Stream content out

          }

      }

}

finally {

      response.close();

}

HTTP POST 交互

String url = "http://127.0.0.1:8080/myweb/servlet/HelloServlet?isbn=1001";

String reqText = "{ some data }";

CloseableHttpClient httpclient = HttpClients.createDefault();

HttpPost httppost = new HttpPost(url);

// 上行数据

StringEntity dataSent = new StringEntity(reqText,ContentType.create("text/plain", "UTF-8"));

httppost.setEntity(dataSent);

CloseableHttpResponse response = httpclient.execute(httppost);

try {

     // 下行数据

     HttpEntity dataRecv = response.getEntity();

     if (dataRecv != null)

     {

          long len = dataRecv.getContentLength();

          if (len != -1 && len < 16384)

          {

               String replyText = EntityUtils.toString(dataRecv);

               return replyText;

          } else

          {

               // Stream content out

          }

     }

}

finally {

     response.close();

}

以上的代码都是直接复制Apache帮助文档中的，更改需要的参数就可以了

注意

1. 在服务器端，建议使用Content-Length方式来发 送应答（参考7.4讲，目的是让方便客户端的解 析）

2. 在客户端，HttpEntity表示发送/接收的数据内容， 含Content-Type, Content-Length, charset 和正文信 息。

3. EntityUtils不能接收chunked方式的数据

小结

介绍了在应用程序里访问HTTP服务的实现方法

9.2 RESTful - RESTful API (基于JSON)

应用示例

假设有一个网站，提供电影票订单的功能

普通方式：

1. list\_movie.jsp 浏览正在上映的电影列表

2. book.jsp 预订电影票

也就是说，该网站以WEB形式提供服务，用户必须

用浏览器登录该网站才能使用相关服务。

RESTful API

RESTful API: representational state transfer (REST)

可以用另外一种形式提供服务, 用户以API调用的方 式来访问这些服务。

实现方式：HTTP+JSON(或 XML)

例如，安卓app一般都是使用这种接口与后台交互

RESTful API示例

以上述院线预订服务为例

提供至少2个服务：

1. /api/ListMovie 用于查询电影的列表

(电影名称，主演，票价，区域）

2. /api/Book 用于预订电影票

( 用户上传客户信息，用于快递送票上门)

RESTful API示例

1. /api/ListMovie 用于查询电影的列表

请求：GET

应答：

{

error:

reason:

data:[ … 电影信息 ]

}

RESTful API示例

2. /api/Book 预订电影票

请求：POST

{

movieId:

time:

client: { 用户姓名、快递地址、联系电话 }

}

应答：

{

error:

reason:

data:[ … 订单信息 … ]

}

代码实现

具体的界面样式可以由客户端自己决定

本例中，使用控制台界面：

 list

列出电影列表

 book [id]

订票

 quit

退出

注：使用json-org.jar提供JSON支持

小结

可以，一个网站可以用RESTful API的形式来提供服务，

客户端通过HTTP+JSON来调用这些服务。

也可以用HTTP + XML来实现，原理相同

以后模拟写的时候，可以把源码好好看看；其实和之前的网站式的访问差不多，只是app端需要多写一点东西而已。

10.1 log4j – 输出日志

日志

为什么需要日志？

服务器一般都是长年累月运行的，需要把重要的事 件记录下来（时间、操作者、事件、其他参数） ，日志可以用于分析。

比如，某天发生一个错误，则需要根据日志来判断 是什么事情导致了这个错误。

apache log4j

log4j : logging for java

官网 http://logging.apache.org/log4j/1.2/

jar文件： log4j-1.2.17.jar，放在WEB-INF\lib下

配置文件：log4j.properties， 放在src根目录下

使用

1. 修改log4j.properties

(里面不能输入中文，具体的修改方式，邵发在其首行注明了，就是改一个对应的包的名字，日志输出目录可改可不改)

2. 在代码里调用log4j API

static Logger logger = Logger.getLogger(HelloServlet.class);

然后，

logger.debug()

logger.info()

logger.warn()

logger.error

…

使用

观察输出。。。

1. 在控制台里的输出

2. 在mylogs目录下的日志文件

日志级别

TRACE,

DEBUG,

INFO,

WARN,

ERROR

FATAL

其中，FATAL最严重, TRACE最轻微

(设置级别是为了方便过滤和分类)

日志的过滤：阈值 Threshold

在log4j.properties里可以按级别过滤，让低级别的不 显示。

1. 在调试的时候，一般设为DBG，这样所有>=DBG的 日志都会被记录

2. 在正式上线运行的时候，一般设为INFO，只有 >=INFO级别的日志才会被记录

记录器 Appender

Appender指定输出目标, log4j已经内置的几个类：

org.apache.log4j.ConsoleAppender 控制台

org.apache.log4j.FileAppender 文件

org.apache.log4j.DailyRollingFileAppender 每天一个日 志文件

org.apache.log4j.RollingFileAppender 文件过大时，产 生一个备份文件（可以指定备份的数量） 其他。。。还可以输出到其他地方，如MySQL。。

记录器 Appender

指定多个Appender

log4j.logger.my=DEBUG,C,R

其中，

DEBUG是总的日志级别

C, 第1个Appender的名字，是自己起的名字

R, 第2个Appender的名字，随便起

小结

先能够使用就行了。

11.1 文件上传

文件上传

目标：客户端选择一个文件，点上传，将文件传到 服务器上。

普通表单域：<input> 数据较小（短的文本）

文件域：<file> 文件数据较大，处理起来相对复杂

文件上传: 前端网页

<form method="POST"

enctype="multipart/form-data"

action="servlet/HelloServlet">

选择文件 :<input type="file" name="upfile">

说明: <input type="text" name="descr">

<input type="submit" value="上传">

</form>

文件上传: 后台

使用apache commons fileupload库 （直接到源码中复制jar包，然后使用就可以了）

1. 下载库，放到 WEB-INF\lib

2. 在doPost()中，调用ServletFileUpload类

（处理过程是通用的，作者是复制帮助文档中的）

小结

文件上传:

(1) 前端 enctype="multipart/form-data"（数据形式要更改）

<file>  域 （文件域和普通的表单域不同）

(2) ServletFileUpload ，请不要使用getParameter()

(3) 可以应答一段文字（可以写上传成功之类的）

11.2 文件上传进度的显示(HTML5)

显示文件上传进度

在上一节课中，点上传之后，由浏览器负责上传文 件，直到完成。完成后显示服务器的应答。。

缺点：

1. 要打开一个新一个页面

2. 不能实时显示进度

文件显示进度一直是个头疼的问题，不能很好的解 决。。。除非浏览器给我们提供API，获取文件的显 示进度。。

HTML5对文件上传的支持

只有最新的浏览器支持以下特性：

1. 支持获取文件对象的属性 (Name, Size …)

2. 支持事件回调

progress : 上传进度

load : 上传完成

error: 出错

abort: 取消

具体参考：

https://www.w3.org/TR/XMLHttpRequest/

实现

1. 服务器端处理不变，使用apache commons  fileupload

2. 客户端使用XMLHttpRequest

前端：获取文件的属性

当选中文件后，可以获知文件的属性

var file = document.getElementById("fileToUpload").files[0];

$("#fileName").html("Name:" + file.name);

$("#fileSize").html("Size:" + file.size);

前端：发起Request

用JavaScript手工发起XMLHttpRequest

var vFD = new FormData(document.getElementById("form1"));

// 事件回调

var oXHR = new XMLHttpRequest();

oXHR.upload.addEventListener("progress", onUploadProgress, false);

oXHR.addEventListener("load", onUploadComplete, false);

oXHR.addEventListener("error", onUploadFailed, false);

oXHR.addEventListener("abort", onUploadCanceled, false);

oXHR.open("POST", "servlet/HelloServlet");

oXHR.send(vFD);

前端：响应事件

显示进度

function onUploadProgress(evt)

{

     if (evt.lengthComputable)

     {

          // evt.load : 已经上传的

          // evt.total : 总数

     }

}

小结

由于现在大多数浏览器已经能够支持HTML5，所以可以使用

这种方法来上传文件 。

12.1 创建一个Filter

应用背景

假设：在网站有一个admin目录，下面的网页和资源只 允许"admin"帐户访问(超级管理员)。

这意味着：如果当前没有用户登录，或者用户名不是 "admin"，都不允许访问 admin/下的任何资源

例如，当用户在浏览器里输入

http://127.0.0.1:8080/myweb/admin/aaa.jpg

时，应提示其无权访问！

怎么实现呢。。。

Filter 过滤器

过滤器：当Tomcat收到WEB请求时，首先调用过滤器来处理， 过滤器可以拦截请求。

例如：设置了3个过滤器

[ filter 1 ]

[ filter 2 ]

[ filter 3 ]

一个请求到来时，首先交给 filter 1 过滤，如果 filter 1没有拦 截该请求，则交给下一个 filter 2 过滤。。。

添加一个Filter

(1) 创建一个类

public class AfSampleFilter implements Filter

(2) 在web.xml中配置

<filter>

     <filter-name>AfSampleFilter</filter-name>

     <filter-class>my.AfSampleFilter</filter-class>

</filter>

<filter-mapping>

     <filter-name>AfSampleFilter</filter-name>

     <url-pattern>/admin/\*</url-pattern>

</filter-mapping>

即：当请求的URL为 /admin/\* 格式时，将交给AfSampleFilter处理

Filter拦截请求

在Filter里，可以决定是否拦截该请求

代码框架为：

void doFilter(request, response, chain)

{

     if ( 需要拦截 )

     {

          … 处理该请求 …

     }

     else

     {

          chain.doFilter(request, response); // 不感兴趣

     }

}

应用 1

检查用户名是否为 "admin"

若不是"admin"，则跳转到登录页

response.sendRidirect ( … )

应用 2

只有某固定网段的IP才允许访问 /admin/\*

小结

Filter机制允许开发人员对请求进行过滤

12.2 Filter的更多细节

1. Filter的生命期

Filter在App启动时一并创建, 直到App被卸载时才被 删除。

也就是说，在App运行期间，Filter的实例是一直存 在的。

init (): 初始化

destroy (): 退出时执行

2. Filter 的配置参数

Filter可以设置一些配置参数, 在web.xml中设置

<filter>

     <filter-name>AfSampleFilter</filter-name>

     <filter-class>my.AfSampleFilter</filter-class>

     <!-- filter的配置参数 -->

     <init-param>

          <param-name>id</param-name>

          <param-value>201501</param-value>

     </init-param>

     <init-param>

          <param-name>location</param-name>

          <param-value> aaa </param-value>

     </init-param>

</filter>

2. Filter 的配置参数

在init函数的参数FilterConfig，就是在web.xml配置的值

public void init(FilterConfig filterConfig) throws ServletException

{

     // 从web.xml中读取配置参数

     Enumeration<String> names = filterConfig.getInitParameterNames();

     while(names.hasMoreElements())

     {

          String key = names.nextElement();

          String value = filterConfig.getInitParameter(key);

          logger.debug("Filter Config: " + key + " = " + value);

     }

}

2. Filter 的配置参数

在init函数的参数FilterConfig，就是在web.xml配置的值

public void init(FilterConfig filterConfig) throws ServletException

{

     // 从web.xml中读取配置参数

     Enumeration<String> names = filterConfig.getInitParameterNames();

     while(names.hasMoreElements())

     {

          String key = names.nextElement();

          String value = filterConfig.getInitParameter(key);

          logger.debug("Filter Config: " + key + " = " + value);

     }

}

3. 多项url-pattern

同一个Filter可以设置多个filter-mapping

<filter-mapping>

     <filter-name>AfSampleFilter</filter-name>

     <url-pattern>\*.html</url-pattern>

</filter-mapping>

<filter-mapping>

     <filter-name>AfSampleFilter</filter-name>

     <url-pattern>\*.jsp</url-pattern>

</filter-mapping>

则凡是\*.html和\*.jsp都会被交给AfSampleFilter过滤

4. FilterChain 多个Filter串连

在web.xml中，可以添加多个Filter，它们组成一串叫做

FilterChain

[ filter 1 ]

[ filter 2 ]

[ filter … ]

WEB请求会依次交给它们过滤。。

注：多个filter会按web.xml里的出现的顺序排列

小结

介绍了Filter的更多使用细节。

Filter的主要用途：对一类请求统一处理

例如，

filter 1 用于处理 \*.jsp请求

filter 2用于处理 /admin/\* 请求。。

Filter很有用，但我们自己写Filter的时候并不会太多。。也就 是说，在你的项目一般不会需要创建太多Filter。。。

12.3 \*自定义应用框架 (1)

\* 提示

需要掌握 Reflection 反射机制

参考 Java学习指南(基础篇) 第47,48,49,50讲 自定义应用框架

其实，Servlet机制就是一个应用框架。它是Tomcat 自带的应用框架。

我们也可以不使用这套框架，自己来定义一套框架。

使用Filter + Reflection 技术，可以添加一套自定义的 应用框架。（如，比较著名的struts框架）

应用背景

在第9章中，在服务器添加RESTful API的一般流程为：

(1) 添加 XXXServlet 类

(2) 在web.xml里添加该Servlet的配置

(3) 在doPost()

- 读取URL里的参数

- 读取上传上来的HTTP请求正文reqText

- ……… 处理请求 ………

- 返回replyText 给客户端

可以把这个标准流程改造成一个应用框架，以提高开发 效率

使用 af.restful 框架

在官网里下载相关文件 ( 第三方库 | af.restful )

按 《AfRestful用户手册》进行

- 拷贝jar文件

- 在web.xml里增加配置

- 创建HelloApi类

- 在af-restful.xml里添加配置项

演示客户端如何调用API。。。

使用框架的好处

1. 精简代码

2. 把业务处理与框架流程独立开来

写HelloApi的人可以不必懂Servlet

小结

介绍了Filter一个重要功能：添加自定义应用框架

( struts2 框架也是类似实现方法 )

对于普通开发人员来说，只需要关心如何使用别人定义好的 框架；

对于高级开发人员/设计人员来说，需要掌握如何自定义一个 框架。

12.4 \*自定义应用框架 (2)

\* 提示

开发人员：只关心怎么使用别人框架 （比较容易）

设计人员：自己定义一个框架

代码解析

小结

struts2框架也是使用了同样的技术

ps：以后想自己写框架，或者是想更加的了解框架的话，可以回来看看，记得先把反射机制弄明白再看。本节主要是通过filter实现框架。

13.1 添加struts2框架支持

Struts2

struts [st'rʌts]

一个WEB应用框架, apache旗下的一个开源项目

官网 http://struts.apache.org/

分1.x 和 2.x两个版本，本教程使用的2.x版本 下载

下载当前最新版本

struts-2.3.28-lib.zip (完整支持包)

struts-2.3.28-min-lib.zip (最小支持包)

struts-2.3.28-docs.zip (文档)

可从 http://afanihao.cn 上直接获取

添加 struts2框架支持

按《在Web Project中添加struts2框架》.pdf进行

三个方面

web.xml

struts.xml

\*.jar

注：手工拷贝到项目里也是可以的

分析

web.xml里添加了一个filter，该filter拦截了 \*.action 请求，交由struts2框架处理

例如，当你输入 xxxxx.action的时候，该请求将被 struts2框架拦截处理。

删除 struts2框架

只需要在web.xml里把相关filter配置删除即可

struts.xml 和 相关jar包可以保留，没有副作用

小结

介绍了如何在Web项目中添加struts2 框架支持

（原理已经在12章讲过，本章只关注它的使用方法）

下一节课将开始学习它的使用方法。

ps：Java学习指南-网站开发->第三方库文件夹中有Apache Struts2的相关软件和配置文档，可以对照着使用。

13.2 第一个Action

背景

URL: example.action?id=201601

查询 id为201601的学生的信息

在页面里显示这个学生的信息

步骤

1. 新建 ExampleAction类

2. 在struts.xml里添加配置

3. 在Result\_Example.jsp里显示结果

Action类

public class ExampleAction extends ActionSupport

{}

（1）重写 execute() 函数

（2）添加成员变量（与URL参数同名），添加setter

struts.xml配置

在struts.xml里添加

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

<action name="Example" class="my.ExampleAction">

     <result name="success"> /Result\_Example.jsp</result>

</action>

</package>

注意：区分大小写

在jsp里显示数据

在jsp文件，显示action里的成员变量（查询结果）

（1）在action里添加成员变量, 及相应的getter

（2）在jsp里显示这个变量

- 添加struts tags

<%@ taglib prefix="s" uri="/struts-tags" %>

- 使用strusts tags来显示变量

<s:property value="name" />

<s:property value="phone" />

struts2框架

在struts框架里，

Model: Action类负责计算得到数据

View : JSP页面负责显示数据（能访问Action对象 里的成员变量）

Controller: xxxx.action找到xxxx的class

Action: 输入参数(input) - 执行(execute) – 输出结果(result)

JSP: 显示result的值

struts2框架

输入参数input:

成员变量名要与表单参数名一致,并有setter ，由框架自动把表单参数的值传进来 (Reflection)

执行 execute():

根据input计算出结果，保存到result里

输出结果result:

成员变量，添加相应的getter

JSP显示结果：

其实是调用action对象相应的getter，来取得结果并显示

小结

初步学习了在struts2框架下处理请求的方法

当接收到 Example.action请求时，struts2框架会根据 struts.xml里的配置，用反射机制创建一个 my.Example类的实例。。。继而进行后面的处

理。。。

原理参考《基础篇》47-50，

《网站开发篇》第12.3, 12.4章

13.3 Action的细节(1)

添加log4j日志支持

按第10章进行

区分大小写

例如，

struts.xml里配置的名字是Example ，那么，在URL里就要写 Example.action，不能写 example.action

表单参数的传递

参数的名称 和 成员变量 的名称要对应

<form ..

<input name="id" … >

<input name="name" …>

</form>

则，Action类里的变量名一般为

int id;

String name;

struts2框架本身并不关心Action类的变量的名字

struts2调用的是setter

例如， 表单参数id=201601&name=shaofa

则，

id 对应 setId ()

name 对应 setName()

Action会调用这两个setter函数。。至于它的成员变量叫 什么，Action是不关心的。。。

（也就是说，成员变量可以不一样，但是setter的格式必须对，set+首字母大写的参数名称，不然Action对应不上）

表单参数的传递

试验1：将setName改成setname，看它会不会被调用 ，（不会）

试验2：保留setName这个函数，但是将成员变量name 改名为name222，看会不会有影响，（没有影响）

试验3：将表单里的字段名改为name2试一下

试验4：在ID栏输出一段不是数字的字符串，试一下 ，（这个不行，如果id栏中不是数字，那么框架在转换格式的时候会报错）

注：为增加可读性，名称应保持一致

表单参数的类型

表单参数都是简单类型，或者为数字，或者为字符 串。

对应的成员变量也是简单类型。可以为：

byte short int long double float boolean String

也可以使用包装类

Byte Short Integer Long Double Float Boolean

表单参数的类型

小结：

1. 名字要对应

如果名字不对应，就无法找到setter，用户输入的值就传不进 来。（但不会报错）

2. 用户输入的类型要匹配

例如，ID字段要求输入数字，如果用户输入的不能转成数字， 则会报：

No result defined for action XXX Action and result input

( struts框架试图调用 Integer.valuesOf("xyz")时抛异常)

解决办法：(1) 在网页里先做检查 (推荐)

(2) 其他办法

注意

在JSP里，

<form method="post" action="Hello.action" >

不能写成

"/Hello.action"

这个目录是根据

<base href="<%=basePath%>"> 来计算的

（这里是相对路径的写法，base路径中已经加了一个“/”，在加一个就重复了，在网页源代码中可以很清楚的看到这一点）

小结

介绍了struts2框架下输入参数传递的相关细节

13.4 Action的细节(2)

result: success与error

以上节课的HelloAction为例：

后台保存操作，如果操作失败了呢？仍然要以

Hello\_success.jsp来显示吗？

如何显示错误页面？

result: success与error

一般来说，一个action要配置2个result:

<result name="success"> /Hello\_success.jsp</result>

<result name="error"> /Hello\_error.jsp</result>

就是说，当HelloAction.execute()成功时，由一个页 面显示结果；当失败时，由一个页面选择结果。

演示。。。

result: success与error

execute()还可以返回多种不同的结果，可以为某种 可能的情况设定一个显示页面。

注：通常我们会有"success"和"error" ，但是你写成"ok", "bad"也是可以的，struts2框架的执 行原理一样的。

method: execute()

默认地，约定调用execute()函数

但是，也可以自己指定一个函数:

<action name="Hello"

class="my.HelloAction" method="execute" >

注：该函数原型必须是

public String xxx () {}

所以，可以设立多个action，指向不同的函数。。

<action name="Login"

class="my.UserAction" method="login" >

</action>

<action name="Logout"

class="my.UserAction" method="logout" >

</action>

小结

重点理解execute() 返回值的作用

：根据不同的结果，由不同的页面显示

13.5 Struts2 Tags – 显示复杂类型的数据

内容提要

在JSP页面里显示复杂对象：

1. 对象

2. 数组

3. Map

4. List

注：struts2 tags，标签 其用法可以其文档中查看

1. 访问对象的成员

Action:

book = new Book();

book.isbn = "9787302419891";

book.title = "C/C++学习指南(语法篇)";

在JSP里，使用struts2 tags显示：

书名: <s:property value="book.title" /> <br><br>

ISBN : <s:property value="book['isbn']" /> <br><br>

访问对象的属性:

(1) book.title 或者 book['title']

(2) 该对象的成员必须有相应的getter函数

2. 访问数组

Action:

boys = new String[4];

boys[0] = "Ming";

boys[1] = "Hua";

boys[2] = "Bo";

strtus-tags:

孩子: <s:property value="boys[0]" /> <br><br>

孩子: <s:property value="boys[1]" /> <br><br>

孩子: <s:property value="boys[2]" /> <br><br>

3. 访问Map

Action:

all = new HashMap<String, Student>();

all.put("Ming", new Student(201601,"Ming", "13800012345"));

all.put("Hua", new Student(201602,"Hua", "13300203434"));

all.put("Bo", new Student(201603,"Hua", "13400034343"));

strtus-tags:

学生信息: <s:property value="all['Ming'].phone" />

即，用all['Ming']可以访问Map中的元素，既而 用.phone来访问这个对象的属性。。

4. 遍历List , Set等

Action:

records = new ArrayList<Student>();

records.add(new Student(201601,"Ming", "13800012345"));

records.add(new Student(201602,"Hua", "13300203434"));

records.add(new Student(201603,"Hua", "13400034343"));

strtus-tags:

<s:iterator value="records" var="row" >

学生: <s:property value="#row.id" /> |

<s:property value="#row.name" /> |

<s:property value="#row.phone" /> <br><br>

</s:iterator>

注意：循环变量为row，取其属性的时候要加#号 （规定如此，不清楚意义）

（iterator：迭代器，用于遍历）

小结

介绍几种复杂数据的显示

最终都调用它的toString()方法来得到一个字符串的

13.6 Struts2 Tags – 访问上下文对象

内容提要

在JSP页面里面访问request, session里的数据

相关概念

Tag Developer Guide | OGNL

|--application

|

|--session

context map--|

|--value stack(root)

|

|--action (the current action)

|

|--request

|

|--parameters

|

|--attr (searches page, request, session, then application scopes)

访问 parameters

在JSP里可以用struts tag来访问request的参数

result: <s:property value="#parameters['check']" />

相当于 request.getParameter("")

访问 session

在JSP里可以用struts tag来访问session的属性

例如，显示session存储的当前用户的信息

result: <s:property value="#session['user']" />

相当于 session.getAttribute("")

访问 request

#request['xxx'] 相当于 request.getAttribute("xxx")

#parameters['xxx'] 相当于 request.getParameter("xxx")

注意两者的区别

访问 action

可以访问action里的成员

result: <s:property value="#action['reason']" />

相当于: <s:property value="reason" />

注意

两种写法：

<s:property value="#session['user']" /> (推荐)

<s:property value="#session.user" />

不能直接访问jsp

http://127.0.0.1:8080/myweb/Test2\_success.jsp

不行！

为什么呢？

strut2框架拦截的是\*.action

实际上，jsp里面的标签是由strut2框架来替换的，把 里面的标签替换成相应的值。

小结

介绍了几种上下文对象的访问方法

它们是使用#号开头来访问的，称为Named Object

13.7 Struts2 Tags – if条件判断

数值判断

<s:if test="result > 100" >

     够大!

</s:if>

<s:else>

     太小了！

</s:else>

字符串判断

<s:if test="#session['user'] == null" >

     未登录

</s:if>

<s:else>

     <s:property value="#session['user']" />,您好!

</s:else>

其他标签

在文档中查看其他标签的用法

13.8 Struts2 Tags – 问题及注意事项

1. 不能直接访问jsp

浏览器里不能直接访问Test2\_success.jsp ，会出错。。。

原因：struts2框架的处理流程

if ( xxx. action ) // 请求URL为\*.action格式

{

     创建 Action object

     调用 execute();

     解析 xxx\_success.jsp，替换里面的标签为相应的值

     将处理后的xxx\_success.jsp页面的内容返回给浏览器

}

如果直接xxx.jsp。。那么不经过struts这套处理流程， 那么jsp里面的struts2标签就无法被替换。。

2. 不是重定向

从action到jsp，并不是一个重定向的过程。

重定向Redirection 涉及来回两次请求

浏览器 >>> GET a.jsp >>> 服务器

浏览器 <<< 302 Location:b.jsp <<< 服务器

浏览器 >>> GET b.jsp >>> 服务器

浏览器 <<< 200 OK <<< 服务器

最终，浏览器显示的是b.jsp的内容

而struts2框架的处理流程为：

浏览器 >>> GET Test.action >>> 服务器

struts2: 创建Action对象

struts2: execute()

struts2: 处理Test\_success.jsp

浏览器 <<< 200 OK <<< 服务器

从流程器的角度来看，就是一个请求一个应答，并 没有特别的地方。

小结

分析了struts2框架的处理流程

13.9 在Action中访问Session

应用背景

用UserDelete.action来负责删除用户的操作。 这是一个管理员admin才能进行的操作。因此，要在 UserDeleteAction里进行权限检查。

权限检查：如果当前用户不是admin，则禁止此操作。

用户信息已经保存在HttpSession里，那么，怎么在 Action类里取得HttpSession对象呢？

SessionAware

class UserDeleteAction extends ActionSupport

implements SessionAware

{

     private Map<String, Object> userSession ;

     public void setSession(Map<String, Object> session)

     {

          userSession = session ;

     }

}

只要Action类实现SessionAware接口，则struts2框架会自动调用 setSession()函数把Session对象传进来。

注：为什么是Map而不是HttpSession？--因为HttpSession其实是 interface，真正的对象是用Map来实现的。

示例： 权限检测

在execute()函数里，先做权限检查，非法用户不得操作，直接返回 "error"。

String user = (String) userSession.get("user");

if (! "admin".equals(user))

{

     // 非admin用户禁止此项操作

     reason = "无权进行此项操作";

     return "error";

}

实际应用中，可以把各种分级权限都存在session中。在Action检查 session中是否有必需的权限。

获取request和response

在Action里也可以获得HttpServletRequest和HttpServletResponse对象

public class UserEditAction extends ActionSupport

implements ServletRequestAware,

ServletResponseAware

{

     。。。

}

但一般不推荐在Action里使用request和response

(1) 可能破坏struts2的处理流程

(2) 并与设计初衷相违背: 业务与框架流程分离

小结

介绍了在Action里访问Session对象的方法

struts2框架会把Session对象传进来

13.10 模型驱动ModelDriven

应用背景

Book.jsp : 编辑一本书的信息，提交到后台

BookAdd.action: 将书的信息保存

一般写法：BookAddAction，保存4个字段

ModelDriven

ModelDriven : 模型驱动的 （是形容词）

类似的还有:

Event-Driven: 事件驱动的

Object-Oriented: 面向对象的

是对上述应用的简化写法，返回一个对象，让struts 参数自地把参数设置进来（存到model里面）

示例：

实现 ModelDriven接口，在getModel()里返回一个对象

public class BookAdd2Action extends ActionSupport

implements ModelDriven

{

     /\* ModelDriver 支持 \*/

     Book model = new Book();

     public Object getModel()

     {

          return model;

     }

}

struts2框架会自动地将表单参数，按名称一一对应地设给model对象

小结

使用ModelDriven接口可以简化代码

注：表单参数的名字，要与model对象里面的属性名 字对应起来(setter)，不然无法把用户的输入传进去。

eg：例子写的比较清楚，如果不用ModelDriven接口，那么需要把表单参数在action中重写一遍并加上setter，使用ModelDriven接口的话，直接传就行了，不用再写一遍。

13.11 struts result – 文件下载

文件下载

前面已经介绍了用Servlet实现文件下载的方式。

这里用Action也可以实现文件下载。

（但Action并不是设计用来做文件下载的。）

文件下载

例如，用struts2框架下载 c:/xxx.rar这个文件

在struts.xml里配

<action name="GetFile" class="my.GetFileAction">

     <result name="success" type="stream">

          <param name="inputName">fileStream</param>

          <param name="contentType">

               application/octet-stream

          </param>

          <param name="contentDisposition">

               attachment;filename=${fileName}</param>

</result>

<result name="error">GetFile\_error.jsp</result>

</action>

配置参数的说明

1. type要设为stream

2. <param name="inputName">指定了输入流的名字， 就是说在Action类里要有一个InputStream类型的 对象。这里<param>指定了它的名字(getter名)

3. 其他<param>和Servlet的含义相同，参考struts2 的文档说明。

4. 在<param>里用 ${fileName} ，表示这里是可变的。

Result Types

result的类型可以多种选择，如

"dispatcher": 默认值

"chain" : 将多个Action串起来

"redirect" : 页面重定向

"stream" : 文件下载

"httpheader" :

…

但是，struts2设计初衷应该是解决Model/View分离的问 题。。。

小结

了解了struts.xml中的result的各种类型

可以使用stream类型完成文件下载

注：struts2的强项是Model/View，针对的是普通JSP

页面的显示。。。其他的都是附带功能

14.1 Hibernate - 配置数据源

Java数据库开发

有几种API可以选择

1. Java API ( JDBC)

即使用 java.sql.\* 下的API进行开发

2. Hibernate / MyBatis

此类API是对JDBC进行了封装，使之更易于使用

本课程讲解 Hibernate + MySQL 的使用

Hibernate

Hibernate是一个独立的库，可以在Web Project使用， 也可以在普通的Java Project中使用。

http://hibernate.org/

和Struts一样，Hibernate也被称为一个“应用框架” ( 也运用了Reflection机制)； 和Struts不同的是，Struts是一个流程性的框架，必 须在Web项目里使用。而Hibernate则可以认为只是 一个工具类，并不一定要在Web项目里使用。

MySQL: 安装和配置

参考 C/C++学习指南（数据库篇）前2章

1.安装MySQL (修改my.ini，字符集设为utf8)

2. 重新启动MySQL (控制面板|管理工具|服务)

3. 添加数据库`webdb` (字符集utf8 /utf8\_general\_ci )

Hibernate：配置数据源

从官网下载

mysql-connector-java-5.0.8-bin.jar

Hibernate图文教程.pdf

按教程进程，演示。。。

小结

准备好了数据库源

ps：Hibernate图文教程.pdf  这个教程非常好，从Hibernate的配置到测试都是非常的齐全。

在SQLyog中新建数据库时的配置：引擎：InnoDB；字符集：utf8；核对：utf8\_general\_ci;

14.2 项目中加入Hibernate

Hibernate

http://hibernate.org/

项目中加入Hibernate

按文档进行。。 （上一节的文档）

小结

在Web项目中添加了Hibernate

注：可以struts框架同时存在

14.3 生成POJO类

POJO

POJO，Plain Ordinary Java Object

简单Java对象

什么样的Java类才算POJO?

： 只有getter,setter, constructor

Hibernate里的POJO

Hibernate提供了小工具，能够根据MySQL里的表的 结构，自动创建一个与之对应的POJO类。

例如，数据库webdb里有一张student表

则我们可以使用Hibernate提供的工具，使用逆向工 程技术，自动地创建一个Student类。

演示

准备一张表: 导入 webdb.sql 到MySQL

根据文档进行。。。

常见问题

a) 运行时提示：Unable to read XML

逆向工程生成 Student.hbm.xml 有时候会多出一个空格。检查 xml 文件，确保

3.0.dtd 和双引号之间没有空格

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

b) 如果MySQL中的表的结构发生变化，例如，增加 了一列，或者列名、列属性发生改变，一定要先 Refresh、再Reverse Engineering重新生成POJO类。

特别注意：每次重新生成pojo类的时候，一定要重新更改类的配置文件，就是上一点，不然会报错。

java.lang.NullPointerException //空指针

出错: null

小结

介绍了自动生成POJO类的方法

（原则上手工写一个份也没问题）

表名student <-> 类名Student

表的字段 <-> 类的成员变量

这个对应关系是在 Student.hbm.xml里描述的

ps：第一节的那个文档里有非常具体的演示

14.4 Hibernate - 添加记录

添加记录

向数据库里添加一条记录，（先按上一节课生成POJO类）

/\* 准备数据 \*/

Student row = new Student();

row.setId(201601);

… …

/\* 连接数据库 \*/

dbss = HibernateSessionFactory.getSession();

Transaction dbtr = dbss.beginTransaction();

dbss.save(row);

dbtr.commit();

常见问题

1. 在定义表时，列名中不得含有SQL关键字

比如 type, by, text , timestamp, order等

2. 如果有多行数据，则

Transaction dbtr = dbss.beginTransaction();

dbss.save(row0);

dbss.save(row1);

dbss.save(row2)。。。

dbtr.commit();

3. save() 失败？ConstraintViolationException

检查表里是不是已经有了记录，重复的主键是无法 再次插入的。

自增的主键

主键字段已经设置为自增

则在添加记录的时候，可以不设置该列的值(默认null)

演示：。。

小结

介绍了一种向表里插入记录的方法

14.5 Hibernate – 查询记录

查询记录

数据库有已经有数据，现在用Hibernate API从中取出

1. 按主键查询

已经该记录的ID（主键），则可以通过ID来查询

2. HQL查询

使用Hibernate Query Language，跟SQL类似，由

Hibernate自己定义的规则

3. Native SQL查询

即普通的SQL语句

按主键查询

Integer rowid = new Integer(201601);

Student row = (Student)dbss.get(Student.class, rowid);

注意：主键的类型须严格匹配。例如，它需要的是 Integer，那么你不定擅自传入一个

Expected: class java.lang.Integer,

got class java.lang.Long

HQL查询

HQL的语法和SQL类似

String hql = "FROM my.db.Student WHERE id>201601";

List<Student> results = dbss.createQuery(hql).list();

for(Student row : results)

{

     System.out.println("姓名:" + row.getName()

      + ",电话:" + row.getPhone());

}

HQL是由Hibernate定义的规则，得按它的规则来写。

注意：HQL的用的类名，不是表的实际名称

例如

" SELECT \* FROM my.db.Student "

" SELECT \* FROM student" 错误！

my.db.Student可以写为Student，用类名，不是表名

Native SQL查询

也可以不使用HQL，而使用我们熟悉的普通SQL:

String sql = "SELECT \* FROM `student` ";

List<Object[]> results = dbss.createSQLQuery(sql).list();

for(Object[] row : results)

{

     Integer id = Integer.valueOf( row[0].toString() );

     String name = row[1].toString();

     String phone = row[2].toString();

     String brithday = row[3].toString();

}

注意： 每一行的类型为Object[]

小结

介绍了几种查询方法

14.6 Hibernate – 修改记录

修改记录

两种方式

1. update()

/\* 连接数据库 \*/

            dbss = HibernateSessionFactory.getSession();

            Integer rowid = new Integer(201601);

            Student row = (Student)dbss.get(Student.class, rowid);

            if(row != null)

            {

                // 修改数据

                row.setPhone("19900001111");

                // 保存到数据

                Transaction dbtr = dbss.beginTransaction();

                dbss.update(row);

                dbtr.commit();

2. HQL

/\* 连接数据库 \*/

            dbss = HibernateSessionFactory.getSession();

            // HQL

            Transaction dbtr = dbss.beginTransaction();

            String hql = "UPDATE Student SET phone='12233330000' WHERE id='201601' ";

            int rowsAffected = dbss.createQuery(hql).executeUpdate();

            System.out.println(rowsAffected + " rows affected.");

            dbtr.commit();

3. Native SQL

/\* 连接数据库 \*/

            dbss = HibernateSessionFactory.getSession();

            // 构造一个普通的SQL语句

            Transaction dbtr = dbss.beginTransaction();

            String sql = "UPDATE `student` SET `phone`='13355550000' WHERE id='201601' ";

            int rowsAffected = dbss.createSQLQuery(sql).executeUpdate();

            System.out.println(rowsAffected + " rows affected.");

            dbtr.commit();

注意

要使用transaction，不然数据不会真正保存到服务器

Transaction dbtr = dbss.beginTransaction();

…

dbtr.commit();

注意

save() : 插入

update() : 更新

saveOrUpdate(): 插入或更新 (不存在时插入，存在时 更新)

小结

介绍了几种更新数据的方法

update(): 需要事先取得一个对象，更新时所有字段 都被更新。

HQL/SQL: 只更新指定的字段。 （更高效，但不好拼凑）

ps：见源码

14.7 Hibernate – 删除记录

删除记录

三种方式

1. delete()

            /\* 连接数据库 \*/

            dbss = HibernateSessionFactory.getSession();

            Integer rowid = new Integer(201601);

            Student row = (Student)dbss.get(Student.class, rowid);

            if(row != null)

            {

                // 保存到数据

                Transaction dbtr = dbss.beginTransaction();

                dbss.delete(row);

                dbtr.commit();

            }

2. HQL

/\* 连接数据库 \*/

            dbss = HibernateSessionFactory.getSession();

            // HQL

            Transaction dbtr = dbss.beginTransaction();

            String hql = "DELETE FROM Student WHERE id='201602' ";

            int rowsAffected = dbss.createQuery(hql).executeUpdate();

            System.out.println(rowsAffected + " rows affected.");

            dbtr.commit();

3. Native SQL

/\* 连接数据库 \*/

            dbss = HibernateSessionFactory.getSession();

            // 构造一个普通的SQL语句

            Transaction dbtr = dbss.beginTransaction();

            String sql = "DELETE FROM `student` WHERE id='201603' ";

            int rowsAffected = dbss.createSQLQuery(sql).executeUpdate();

            System.out.println(rowsAffected + " rows affected.");

            dbtr.commit();

注意

要使用transaction，不然数据不会真正保存到服务器

Transaction dbtr = dbss.beginTransaction();

…

dbtr.commit();

小结

介绍了删除数据的方法

也许Native SQL更好些

14.8 在Web应用中访问数据库

应用背景

用户在浏览器打开网站，可以执行以下操作：

1. Add.jsp

可以输入一个同学信息，保存到网站上

2. Show.action

可以查看所有同学的信息

实现

前台: bootstrap

后台: Struts2 + Hibernate4

小结

介绍在Web项目里使用Hibernate访问数据库的方法