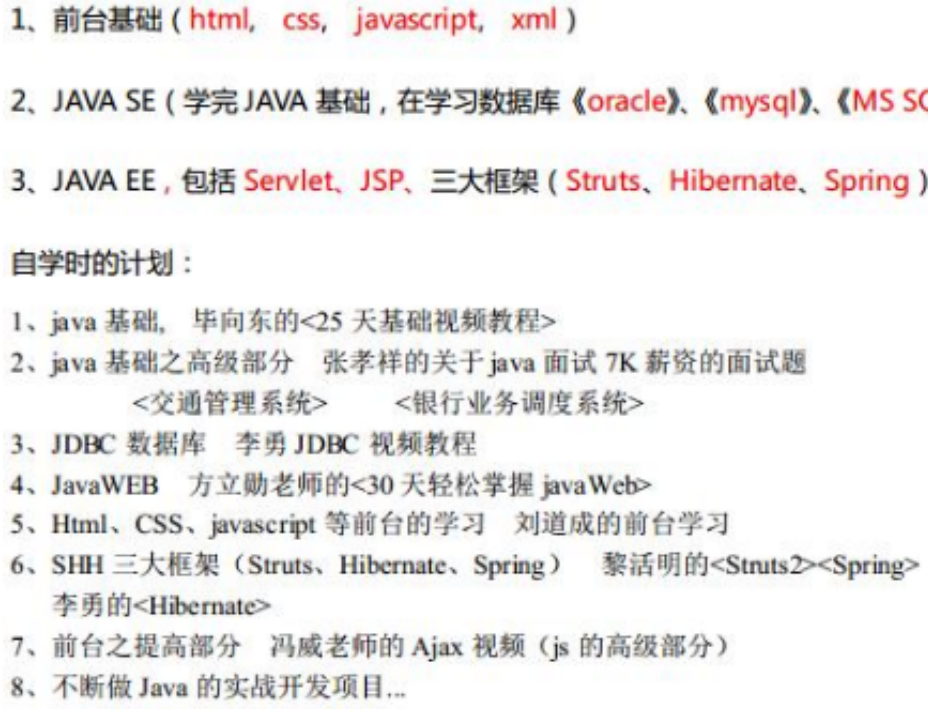
Xiang's Java Server Pages Study

阿里云（镜像），包括atom。https://npm.taobao.org/
http://pan.baidu.com/share/link?shareid=730651746&uk=291977746
某人的javaweb学习路线



1. Java Web Introduction

jsp Environment Established

- jdk7.0
  配置环境变量：
  JAVA_HOME, JAVA_JRE, Path, CLASSPATH
- tomcat7.0
  配置环境变量：

```
CATALINA_HOME=C:\apache-tomcat-7.0.73
```

到tomcat主目录的bin目录中找startup，运行之；关闭是shutdown
测试：explorer地址栏中输入http://localhost:8080

DIY Your First Website：

- webapps里创建自己的文件夹xiang_home
- 在xiang_home/中创建
  index.jsp
  /WEB-INF/

index.jsp中写入：

```
<!doctype jsp>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <meta name="Generator" content="EditPlus?">
  <meta name="Author" content="">
  <meta name="Keywords" content="">
  <meta name="Description" content="">
  <title>我的第一个jsp页面(My first jsp page)</title>
 </head>
 <body>
    <h1>欢迎来到Xiang的家(Welcome to Xiang's home)</h1>
    <hr>
 </body>
</html>
```

/WEB-INF/中创建 classes 文件夹和 lib文件夹 以及 web.xml文件
web.xml文件的内容是：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

</web-app>
```

最后验证，输入地址http://localhost:8080/xiang_home/index.jsp查看结果（乱码的话，在浏览器空白处右键选择“编码”选项）。

认识WEB-INF目录

1. WEB-INF是安全目录，client-site无法访问，而server-site可以访问；
2. web.xml，项目部署文件。原来默认index.jsp为网站默认访问页，想修改，则改web.xml，代码如下：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

<welcome-file-list>
    <welcome-file>/haha.jsp</welcome-file>
</welcome-file-list>

</web-app>
```

上面代码把默认index.jsp改成了haha.jsp。

3. /classes，存放编译了的字节码文件；
4. /lib, 存放项目用到的jar包。

MyEclipse10 与 Eclipse 关于JavaWeb开发的配置

- **eclipse**仍然可以开发**javaweb**应用程序，视频**1-10**有详细过程，但众多网友，包括我在内砸了键盘！太坑了**(**视频先讲了**MyEclipse10**的配置，配置了半天（下载**MyEclipse10**时间挺长的），最后又讲了**Eclipse**的配置 **http://www.imooc.com/video/2931/0)**！。
    - 打开eclipse, 打开javeEE视图,右上角
    - new-->project-->Web-->Dynamic Web Project-->下一步按钮
    - Target runtime 处选择 tomcat7-->下一步按钮
    - 选择Tomcat7安装的主目录 JRE-->选择jdk7.0 然后一顿finish.
    - 在创建好的JavaEE项目中的 WebContent文件夹中创建 index.jsp, 编辑之；
    - 更改index.jsp文件中的字符集，既：charset = UTF-8，修改标题title, body中增加一个

```
<h1>欢迎来搞javaEE!!</h1>
```

    - 运行这个project: 点击本工程--> Run As --> Run on Server, 运行完成，并可在浏览器中观察结果。
- 一大块内容，MyEclipse10中搭建Tomcat7。
- 项目的虚拟路径设置
    - 项目名右击
    - 属性
    - MyEclipse
    - Web
    - 修改虚拟路径
- 修改Tomcat Server 默认端口号
    - 到Tomcat主目录中找并修改conf/server.xml文件

```
<Connector port = "8080"
    protocol = "HTTP/1.1"
    connection Timeout = "20000"
    redirectPort = "8443"
/>
```

MyEclispse2014 for ubuntu14.04 Configuration

1. 安装jdk环境 Java 1.7
2. 下载https://www.genuitec.com/products/myeclipse/download/get/?2014-ga-pro-linux
3. 运行 ./myeclipse-pro-2014-GA-offline-installer-linux.run
4. 下载破解文件http://pan.baidu.com/s/1jG0twlK，并解压文件。
   打开文件 run.bat ，将里面的内容复制到终端（javaw -jar cracker.jar）并修改为Java -jar cracker.jar运行
5. 接下来参照windows破解
   http://jingyan.baidu.com/album/7082dc1c57eb19e40a89bdcd.html?picindex=8 ，即可完成破解。

2. JSP Basic Grammar

> 1. page command
> 2. Jsp annotation
> 3. Jsp script

> 4. Jsp declaration
> 5. Jsp expression
> 6. Jsp pages life cycle
> 7. Stage project

page指令

page指令有language import contentType等字段。

```
<%@ page language="java" import="java.util.*" contentType="text/html; cha
rset=utf-8"%>
```

- 在MyEclispse2014中，命令提示是：**alt+/**

Jsp注释

```
- <!-- html注释 --> //客户端查看源代码时，注释可见
- <%-- JSP注释 --%> //客户端不可见
- ```xml
    //单行注释    /* */多行注释
    ```

-   ```xml
    <% //单行     /*多行*/  %>
    ```
```

Jsp脚本

在JSP页面中执行的java代码。
Grammar:

```
<%java代码%>
```

e.g.

```
<% out.println("欢迎大家学习javaee开发"); %>
```

Jsp声明

在Jsp页面中定义变量或者方法。
Grammar:

```
<%! java code %>
```

e.g.

```
<%!
    String s = "张3"; //declare a string var.
    int add(int a, int b){
        return a+b;
    }
%>
```

Jsp表达式

Grammar:
<%= 表达式 %> //注意：表达式不以分号结束
e.g.

```
    <%!
        String s = "张3"; //declare a string var.
        int add(int a, int b){
            return a+b;
        }
    %>

    <%
        out.println("欢迎大家学习javaee开发");
            out.println(s);//调用生明里的s
    %>
    <br>
    你好，<%=s %><br> //表达式
    x+y=<%=add(10,22) %> //表达式
```

Jsp页面的生命周期

本节说了个

```
C:\apache-tomcat-8.5.9\work\Catalina\localhost\test\org\apache\jsp
```

其中，\work目录是服务器存放java源文件和字节码文件的地方。当第一个client访问该jsp页面时，该页面将被编译，jsp页面在服务器更新时，有人访问的话，也会重新编译。否则，不再编译。
视频未解释清楚的一个方法，如下：
当用户第一次请求一个jsp页面时，首先被执行的方法是jspInit(), 一个initiate方法, 答案是不对， 是构造方法~！ 什么玩意~

```
    <%
        //ctrl+/自动import所需类
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String s = sdf.format(new Date());
    %>
```

阶段项目

打印99乘法表

```
<%!
        String mul(){
            String s = "";
            for(int i = 1; i <= 9; i++){
                for (int j=1; j <= i; j++)
                {
                    s += i+" * "+j+" = "+(i*j)+"   &nbs
p;";
                }
                s += "<br>";
            }
            return s;
        }
    %>
    <h1>99乘法表</h1>
    下面是99乘法表<br>
    <%=mul() %>
```

```
<%!
        void printmt(JspWriter out) throws Exception {//注意和上例的区别
            for(int i = 1; i <= 9; i++){
                for (int j=1; j <= i; j++)
                {
                    out.println(i+"*"+j+"="+(i*j)+"   &nbs
p;");
                }
                out.println("<br>");
            }
        }
    %>
    <h1>99乘法表</h1>
    下面是99乘法表<br>
    <%printmt(out); %> //再注意这里。
```

3. Jsp内置对象(上)

本章内容：

> 1. 内置对象简介
> 2. 四种作用域范围
> 3. out 它的作用域是 page
> 4. request 作用域是request /response
> 5. session
> 6. application
> 7. 其他内置对象
> 8. 项目案例

内置对象简介

JSP内置对象是Web容器创建的一组对象，不使用**new**创建之，就可使用的内置对象。

e.g.

```
int[] value={60,70,80};
for(int i:value){
    out.println(i); //out是内置对象
}
```

JSP有九大内置对象：

- out
- request/response
- session
- application
- 不太常用的：page, pageContext, excption, config.
  client(可以认为是浏览器)发request, web server 接收到request, 然后response.

四种作用域范围

out

out对象是JspWriter类的实例，经常用于向客户端（浏览器）输出内容。
out对象的常用发法：

- void println() 打印字符串
- void clear() 清除buffer的内容，如果在flush后调用，会抛出异常
  - void clearBuffer() 同上，但在flush后调用不会抛出异常
- void flush() 冲厕所技能，它将buffer内的内容输出到客户端
- int getBufferSize() 返回buffer字节数大小，若不设buffer则为0
- int getRemaining() 返回buffer还剩多少可用
- boolean isAutoFlush() 缓冲区满时，是自动清空还是抛出异常
- void close() 关流
  e.g.

```
<body>
          <h1>out内置对象</h1>
    <%
        out.println("<h2>静夜思</h2>");
        out.println("床前明月光<br>");
        out.println("疑是地上霜<br>");
        out.flush();
        //在flush()之后，用out.clear()，服务器会抛出异常;
        out.println("举头望明月<br>");
        out.println("低头思故乡<br>");
    %>
    缓冲区大小：<%=out.getBufferSize() %><br>
    缓冲区剩余大小：<%=out.getRemaining() %><br>
    是否自动清空缓冲区：<%=out.isAutoFlush() %><br>
</body>
```

get 与 post 区别

```
<form name="regForm" action="动作" method="提交方式">
</form>
```

表单提交方式有两种：get和post

- get: 提交的数据在url中可以看到，数据最多不超过2KB。适合提交小数据量，安全性低的数据。如：搜索、查询；
- post: 提交信息会封装在 HTML HEADER 内。适合数据量大，安全性高的数据。如：注册、修改、上传等功能。
  下面的例子将展现上面所述的特性：
  e.g.
  建立login.jsp:

```
<body>
    <h1>用户登录</h1>
    <hr>
    <form action="dologin.jsp" name="loginform" method="post"> <!--试一试post-->
      <table>
        <tr>
            <td>用户名：</td>
            <td><input type="text" name="username"/></td>
        </tr>
        <tr>
            <td>密码：</td>
            <td><input type="password" name="password"/></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="登录"></td>
        </tr>
      </table>
    </form>
  </body>
```

建立dologin.jsp:

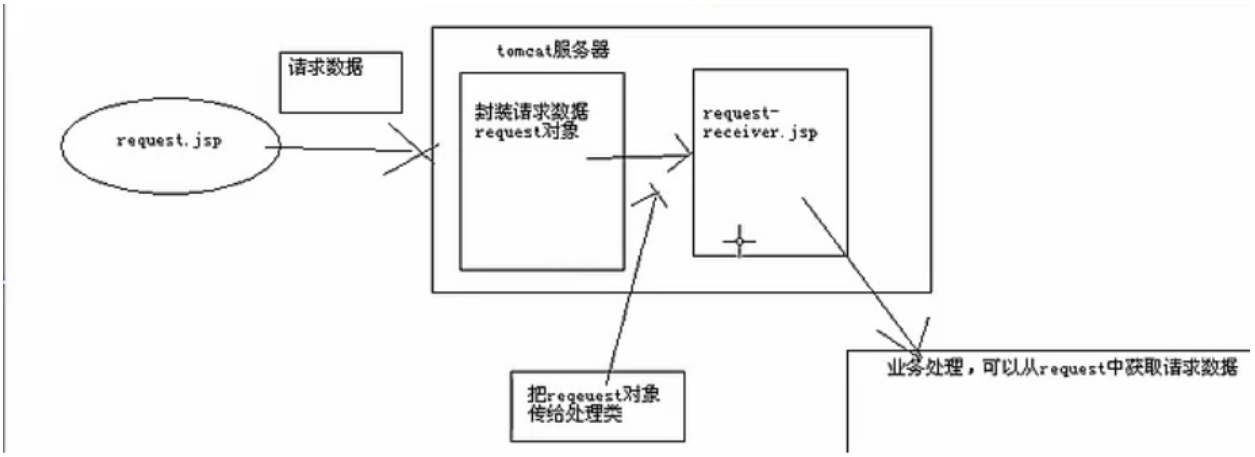
```
<body>
    <h1>登录成功</h1>
    <hr>
</body>
```

这样，服务器端就获得了用户提交的信息，具体是什么情况？ 看下节。

request/response

request对象

request.jsp-->request_receiver.jsp(用它处理)-->处理的结果再返回



客户端的请求信息被封装在request对象中，它是HttpServletRequest类的实例，常用方法如下：

- String getParameter(String name) 返回name指定参数的参数值
- String[] getParameterValues(String name) 返回包含参数name的所有值得数组
- void setAttribute(String, Object) 存储此请求中的属性
- object getAttribute(String name) 返回某属性值
- String getContentType() 得到请求体的MIME类型
- String getProtocol() 返回请求所用的协议类型及版本号
- String getServerName() 返回接收请求的服务器主机名
- int getServerPort()
- String getCharacterEncoding()
- void setCharacterEncoding()
- int getContentLength()
- String getRemoteAddr()
- String getRealPath(String path) 返回一虚拟路径的真实路径
- String getContextPath() 返回上下文路径

e.g.

建立reg.jsp

```
<body>
    <h1>用户注册</h1>
    <hr>
    <form name="regForm" action="request.jsp" method="post">//request.jsp
应该有反应
        <table>
            <tr>
                <td>用户名：</td>
                <td><input type="text" name="username"></td>
            </tr>
            <tr>
                <td>爱好：</td>
                <td>
                    <input type="checkbox" name="favorite" value="read">读
书
                    <input type="checkbox" name="favorite" value="music">
音乐
                    <input type="checkbox" name="favorite" value="movie">
电影
                    <input type="checkbox" name="favorite" value="interne
t">上网
                </td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit" value="提交"/></td>
            </tr>
        </table>
    </form>
    <br>
    <br>
    <a href="request.jsp?username=李四">测试URL传参数</a> //需配置tomcat8的
C:\apache-tomcat-8.5.9\conf\server.xml第70行Connector标签中最后面加入 URIEn
coding="utf-8"，重启tomcat服务器后解决中文乱码问题。不过似乎在tomcat8上不用这
么干，也不会出现乱码。
</body>
```

建立request.jsp

```
<body>
    <h1>request内置对象</h1>
    <%request.setCharacterEncoding("utf-8");//解决中文乱码问题
    %>
    用户名：<%=request.getParameter("username") %>
    爱好：<%
            if(request.getParameterValues("favorite")!=null){
                String[] favorites = request.getParameterValues("favorit
e");//和reg.jsp中表单对应名字一致
                for(int i=0; i<favorites.length; i++){
                    out.println(favorites[i]+"  ");
                }
            }
        %>
  </body>
```

把request.jsp又改成了下面的样子，以此体现request的其他方法：

```
<body>
    <h1>request内置对象</h1>
    <%request.setCharacterEncoding("utf-8");//解决中文乱码问题
      request.setAttribute("password", 1111);
    %>

    用户名：<%=request.getParameter("username") %>
    爱好：<%
            if(request.getParameterValues("favorite")!=null){
                String[] favorites = request.getParameterValues("favorit
e");//和reg.jsp中表单对应名字一致
                for(int i=0; i<favorites.length; i++){
                    out.println(favorites[i]+"  ");
                }
            }
        %><br>
   密码：<%=request.getAttribute("password") %><br>
   请求体的MIME类型：<%=request.getContentType() %><br>
   请求所用协议类型：<%=request.getProtocol() %><br>
   服务器主机名：<%=request.getServerName() %><br>
   getServerPort():<%=request.getServerPort() %><br>
   getCharacterEncoding(): <%=request.getCharacterEncoding() %><br>
   请求文件的长度（字节）：<%=request.getContentLength() %><br>
   请求客户端的ip地址：<%=request.getRemoteAddr() %><br>
   请求的真实路径：<%=request.getRealPath("request.jsp") %><br>
   请求的上下文路径：<%=request.getContextPath() %><br>
  </body>
```

response对象

response包含了响应客户请求的有关信息，只对当前访问有效，它是HttpServletResponse类的实例。常用方法如下：

- String getCharacterEncoding() 返回响应用的是哪种字符编码
- void setContentType(String type)
- PrintWriter类的 getWriter()方法 返回可以向客户端输出字符的一个对象
- sendRedirect(String location) 重定向客户端的请求

e.g.
建立 response.jsp

```
<%@page import="java.io.PrintWriter"%>
<%@ page language="java" import="java.util.*" contentType="text/html; cha
rset=utf-8"%>
<%
    response.setContentType("text/html; charset=utf-8");//设置响应的MIMI类
型
    out.println("<h1>response内置对象</h1>");
    out.print("<br>");
    //out.flush();//加flush()后，out就比outer输出优先了。

    PrintWriter outer = response.getWriter();//获取输出流对象
    outer.println("大家好，我是response对象生产的输出流outer对象");
    //显示的结果 看到PrintWriter对象打印的内容在内置对象out之前打印。

    response.sendRedirect("reg.jsp");//请求重定向。当用户访问response.jsp
时，会立即跳转到reg.jsp；相当于客户端又发了个新的请求。
%>
```

> 请求转发与请求重定向
> - 请求重定向： 客户端行为，response.sendRedirect(), 本质上等同于两次请求，前一次请求的对象不保存，URL地址会改变。
> - 请求转发： 服务器行为， request.getRequestDispatcher().forward(req, resp); 是一次请求，转发后请求对象会保存，地址栏URL地址不会改变。

session

application

其他内置对象

项目案例