一、什么是JSP

JSP全称Java Server Page，基于Java语言，是一种动态网页技术。

它使用JSP标签在HTML网页中插入Java代码。标签通常以<%开头以%>结束。

JSP本质是简化版的Servlet，JSP在编译后就变成了Servlet。JVM只能识别Java的类，是无法识别JSP代码的。所以WEB服务器会将JSP编译成JVM能识别的Java类。

JSP跟Servlet区别在于，JSP常用于动态页面显示，Servlet常用于逻辑控制。在代码中常使用JSP做前端动态页面，在接收到用户输入后交给对应的Servlet进行处理。当然JSP也可以当做后端代码进行逻辑控制。

二、JSP基础知识

以下相关概念了解即可。后续可再根据自己兴趣拓展学习。

1、JSP文件后缀名为 `*.jsp`

2、JSP代码需要写在指定的标签之中，比如：

```
常用：
<%
out.println("hellpo JSP!");
%>
```

```
<jsp:scriptlet>
    代码片段
</jsp:scriptlet>
```

3、JSP生命周期：`编译阶段 -> 初始化阶段 -> 执行阶段 -> 销毁阶段`，此处多了一个编译阶段，是将JSP编译成Servlet的阶段。而这个阶段也是有三个步骤的：`解析JSP文件 -> 将JSP文件转为servlet -> 编译servlet`。

4、JSP指令：是用来设置JSP整个页面属性的。格式为：`<%@ directive attribute="value" %>`。JSP中的三种指令标签：

指令	描述
<%@ page ... %>	定义网页依赖属性，比如脚本语言、error页面、缓存需求等等
<%@ include ... %>	包含其他文件
<%@ taglib ... %>	引入标签库的定义

5、JSP的九大内置对象（隐式对象），这九个对象，可以不用声明直接使用。

名称	类型	描述
out	javax.servlet.jsp.JspWriter	页面输出
request	javax.servlet.http.HttpServletRequest	获得用户请求
response	javax.servlet.http.HttpServletResponse	服务器向客户端的回应信息
config	javax.servlet.ServletConfig	服务器配置，可以取得初始化参数
session	javax.servlet.http.HttpSession	保存用户的信息
application	javax.servlet.ServletContext	所有用户的共享信息
page	java.lang.Object	指当前页面转换后的Servlet类的实例
pageContext	javax.servlet.jsp.PageContext	JSP的页面容器
exception	java.lang.Throwable	表示JSP页面所发生的异常，在错误页中才起作用

三、JSP程序运行

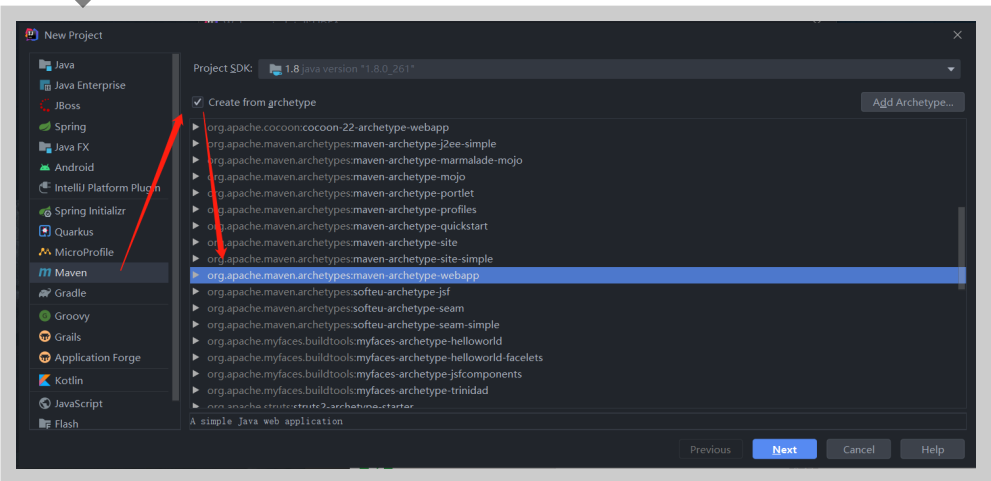
光说不练假把式。

通过创建JSP项目加上代码demo来进一步理解下吧。

2.1、创建项目

下面步骤讲解是基于Maven创建的JSP项目。

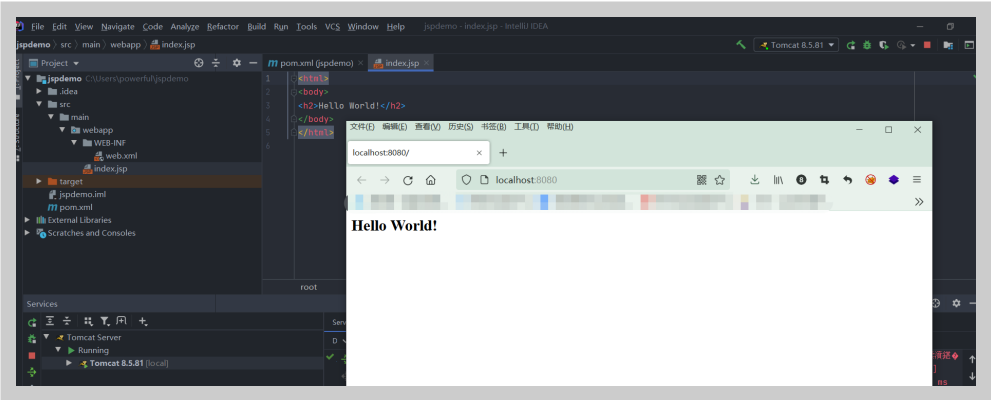
①、打开IDEA后，点击 `Create New Project` ，选择Mavan，勾选 `Create from archetype` ，选择的模板是 `maven-archetype-webapp` ，如下图所示：



②、然后点击Next，给项目起个名字，其他默认即可。此处步骤和 `1.2.1 JavaWeb基础（一）Servlet以及过滤器、监听器和拦截器 中 1.3.1、创建项目` 相同，不再演示。

③、下面开始配置Tomcat。此处步骤和 `1.2.1 JavaWeb基础（一）Servlet以及过滤器、监听器和拦截器 中 1.3.2、环境配置Tomcat` 相同，不再演示。

④、配置完成后，点击运行，最终结果如下图所示：

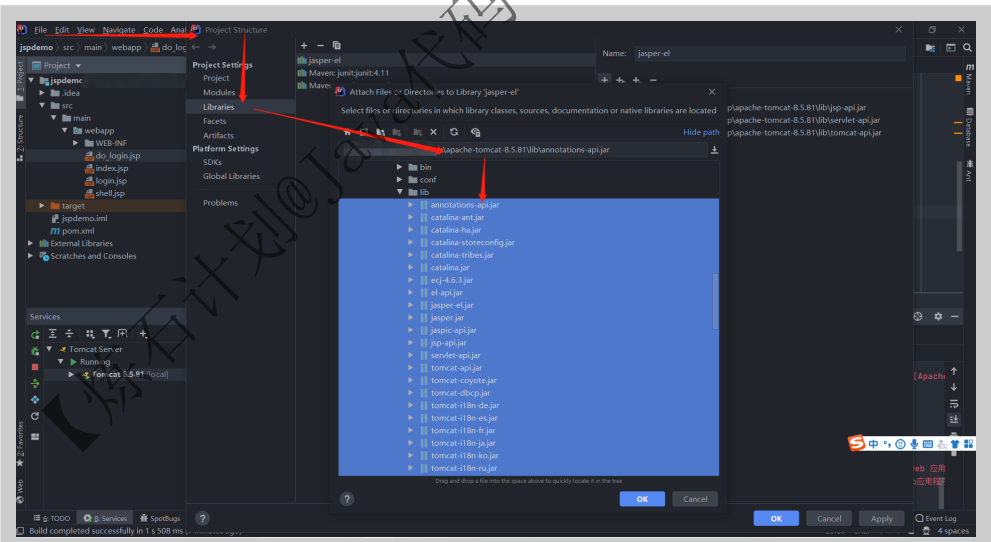


也有较为原始，创建配置JSP项目的方法，此处不过多讲解了。可以参考这篇文章：`https://blog.csdn.net/qq_41647999/article/details/89011686`。

2.2、代码Demo

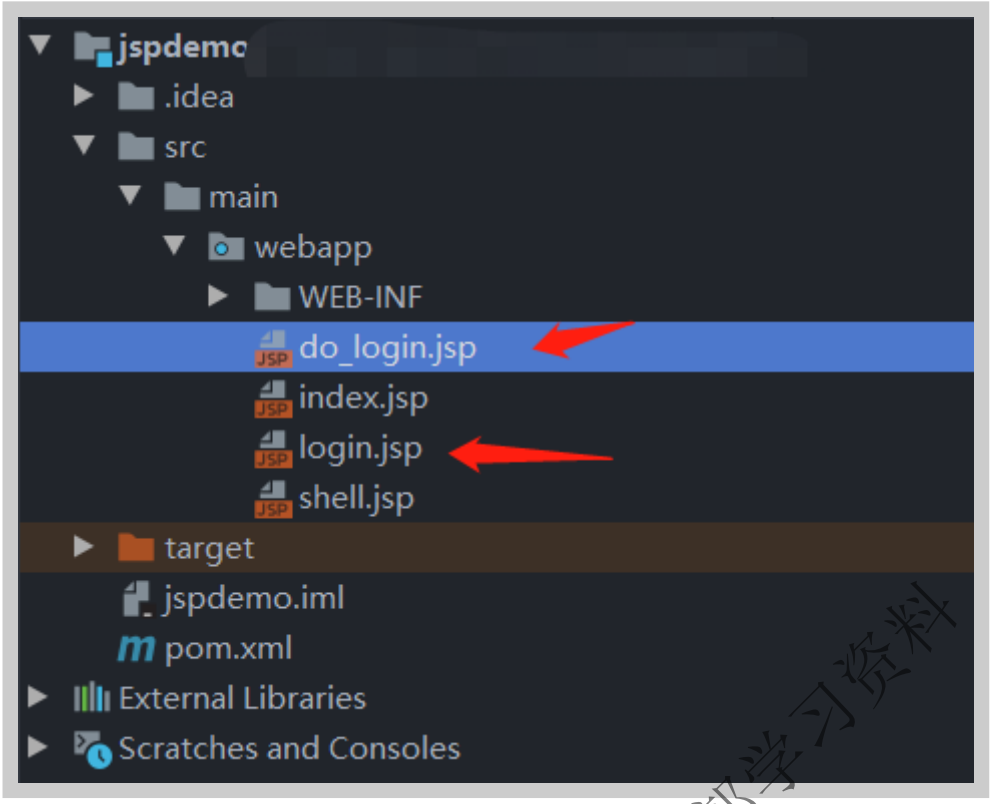
我们通过JSP模拟一下登录的过程。

首先引入一下依赖Jar包。点击左上角 `File - Project Structure - Libraries`，点击 `+` 号，进入Tomcat的lib文件夹下，将依赖全部引入（对于我们这样最省事），最后点击OK。如下图所示：



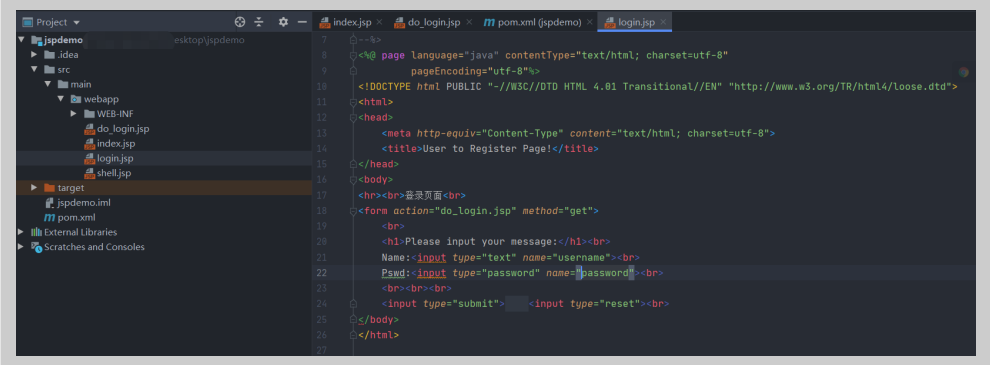
下面编写代码。

①、在webapp下面创建两个JSP文件，分别为 `login.jsp` 和 `do_login.jsp`，如下图所示：



②、在 `login.jsp` 键入以下代码，如下图所示：

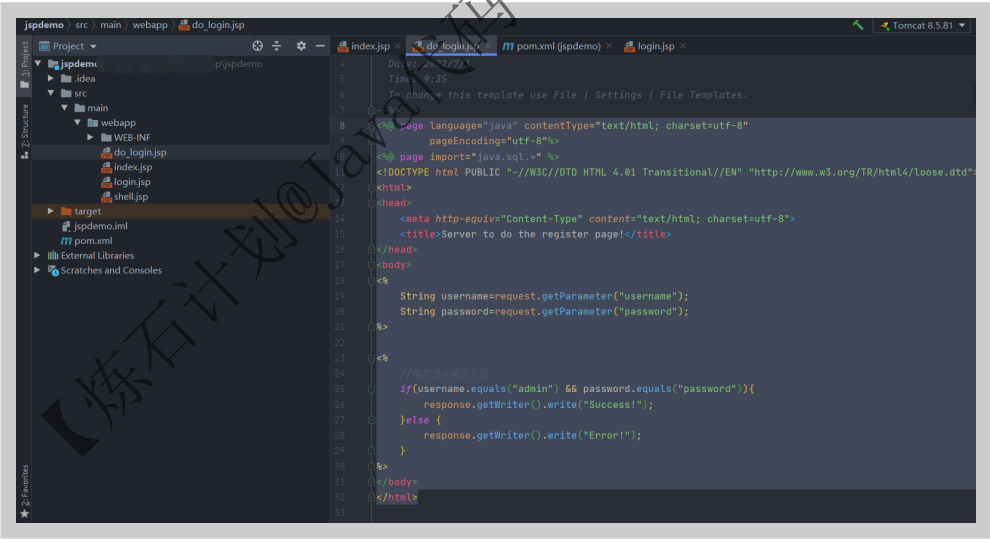
```
<%@ page language="java" contentType="text/html; charset=utf-8"
        pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>User to Register Page!</title>
</head>
<body>
<hr><br>登录页面<br>
<form action="do_login.jsp" method="get">
    <br>
    <h1>Please input your message:</h1><br>
    Name:<input type="text" name="username"><br>
    Pswd:<input type="password" name="password"><br>
    <br><br><br>
    <input type="submit">    <input type="reset">
<br>
</body>
</html>
```



③、在 `do_login.jsp` 键入以下代码，如下图所示：

```
<%@ page language="java" contentType="text/html; charset=utf-8"
         pageEncoding="utf-8"%>
<%@ page import="java.sql.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Server to do the register page!</title>
</head>
<body>
<%
    String username=request.getParameter("username");
    String password=request.getParameter("password");
%>

<%
    //模拟登录成功与否
    if(username.equals("admin") && password.equals("password")){
        response.getWriter().write("Success!");
    }else {
        response.getWriter().write("Error!");
    }
%>
</body>
</html>
```



④、左上角运行代码，浏览器访问 `127.0.0.1:8080/login.jsp` ，分别输入正确和错误的账号密码，观察操作结果。

以上仅是Demo案例，仅用于演示理解JSP代码。

推荐一个关于jsp配合servlet的小案例，感兴趣的朋友可以私下学习：`https://blog.csdn.net/u011486491/article/details/103710087` 。

四、JSP木马

前几年是JSP木马的丰盛时期，由于技术的迭代。现在大型企业使用SprinBoot框架来开发的系统了，该框架默认是不引入JSP解析的，需要引入特定依赖才可以。

而且现在前端大多使用vue，thymeleaf，freemarker等等。因此JSP木马也算逐渐没落了。

但我们还是得学习了解，毕竟网站数量基数很大，难免会在授权的测试中遇见。

JSP木马也可以称作JSP Webshell，如果对方在上传文件或其他功能没有做防护的话，攻击者可以利用任意文件上传漏洞将恶意代码传到后端，继而攻击者可以达到操作目标网站的目的。

切勿对未授权的系统进行非法测试，这是违法行为。

推荐一些较为老派的JSP木马的github仓库：

```
https://github.com/theralfbrown/WebShell-2/tree/master/jsp
```

近两年主流webshell管理工具：冰蝎，哥斯拉，蚁剑.....

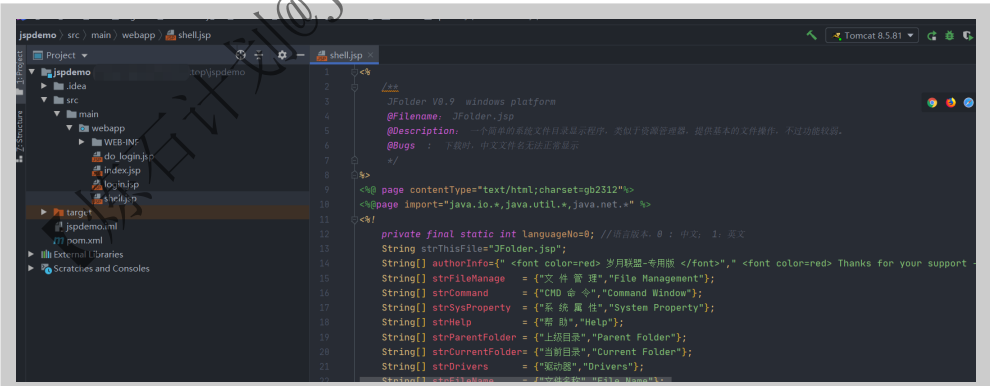
```
冰蝎：https://github.com/rebeyond/Behinder
蚁剑：https://github.com/AntSwordProject
哥斯拉：
```


1、JSP大马

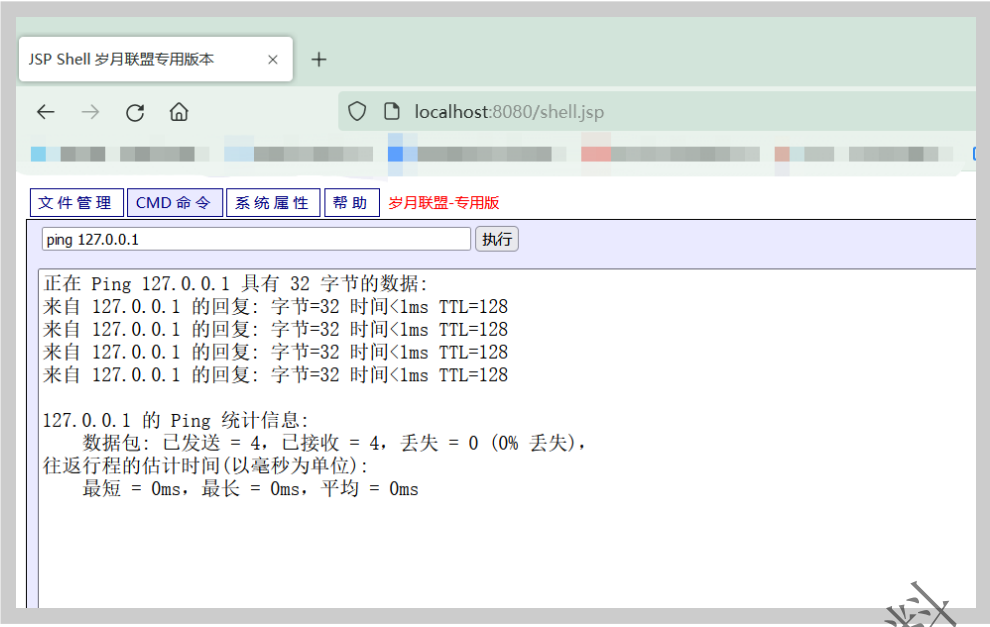
JSP大马，在代码中写入了更多功能，可以实现在网页中进行一些危险操作。

以下演示案例中的JSP大马用的是 `https://github.com/theralfbrown/WebShell-2/tree/master/jsp` 这个仓库中 `无密码的jsp大马.jsp` 。

①、在webapp下新建个 `shell.jsp` ，在该文件中复制粘贴以上代码。如下图所示：



②、点击运行，；浏览器访问 `127.0.0.1:8080/shell.jsp` ，从中可以看到，我们可以进行的危险操作有上传文件、执行命令等等，如下图所示：



建议各位动手操作吧，动手操作加深理解。

感兴趣的可以自己私下阅读学习源码。

2、Godzilla（哥斯拉）JSP木马操作

Godzilla（哥斯拉），主要用于管理Webshell的客户端工具。近两年比较主流的工具。哥斯拉内置了3种Payload以及6种加密器,6种支持脚本后缀,20个内置插件。

Github地址：

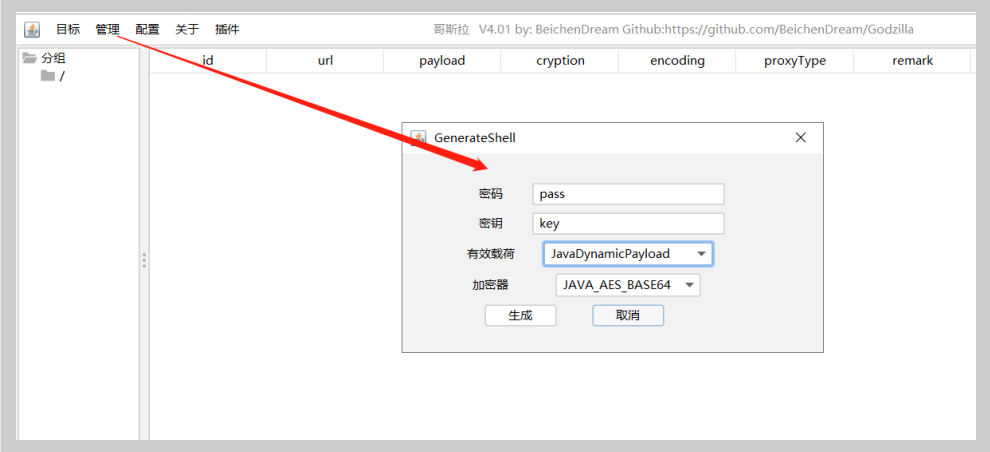
```
https://github.com/BeichenDream/Godzilla
```

下载地址：

```
https://github.com/BeichenDream/Godzilla/releases/tag/v4.0.1-godzilla
```

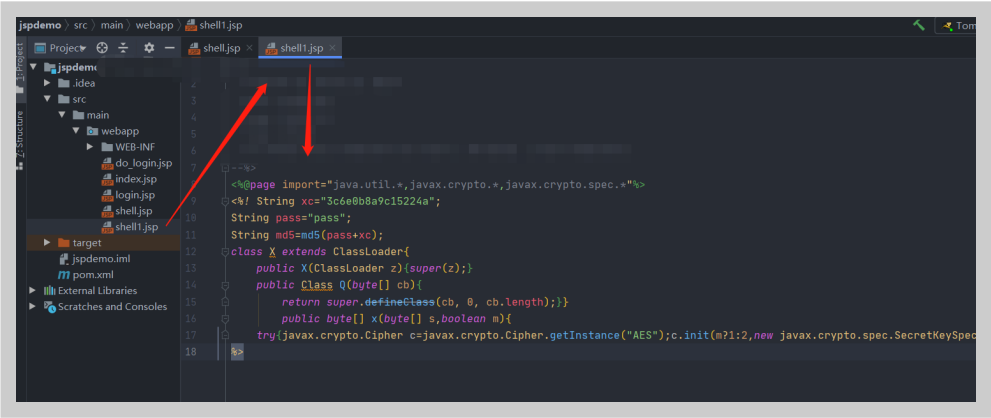
下面我们在代码文件中再新建一个名为 `shell1.jsp` 文件，配合哥斯拉的JSP木马进行操作。

①、下载完成后，启动哥斯拉，点击左上角管理，选择生成，此步骤是为了生成JSP木马代码，如下图所示：

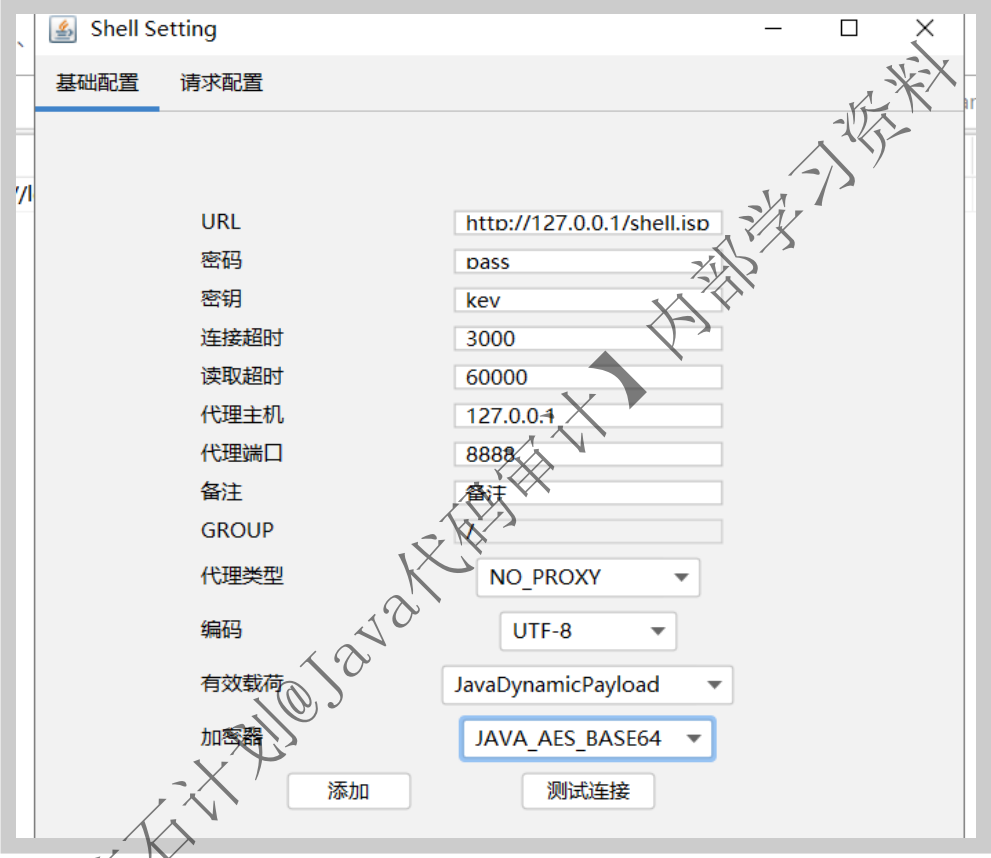


然后点击生成，suffix选择为jsp，点击确定后选择存放目录。

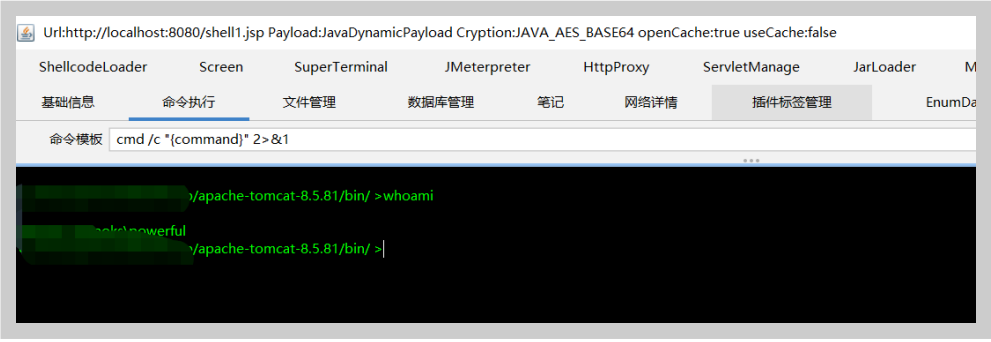
②、将已生成的代码复制粘贴到 `shell1.jsp` 中，然后启动项目。如下图所示：



③、打开哥斯拉，点击左上角目标，选择添加，将shell1.jsp地址添加进去后点击添加，如下图所示：



④、点击添加后，在管理页面即可看到该链接，右键选择进入，成功进入webshell管理页面，剩下的自己研究下吧。



JSP章节到此结束了。关于以上代码建议大家动手操作，便于加深理解。

对于哥斯拉JSP木马代码，在后续提升阶段可以进一步学习，目前理解即可。感兴趣的可以私下进一步学习。