

# 1.使用下面的 东西来得到什么是自定义标签

我的 ip是 ：

<%

String ip = request.getRemoteAddr();

String host = request.getRemoteHost();

out.println(ip);

%>

<br/>

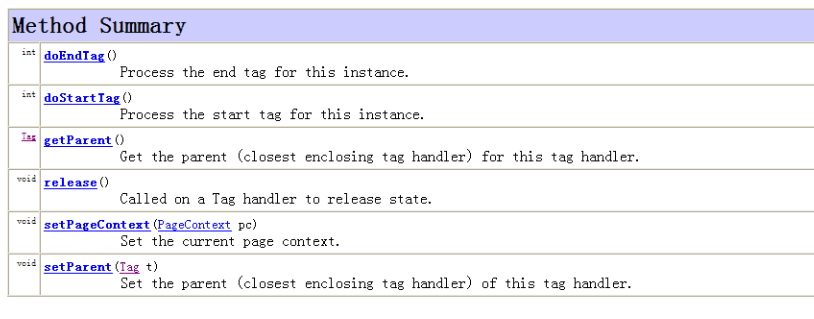
<%

out.println(host);

System.out.println(ip);

%>

## 2.现在就将上面的java的功能，利用自定义标签得到



## 2.1现在为了移除上面的代码，我们需要新建一个类ViewIpTag，同时也需要继承一个类TagSupport（然后重写方法）

**public** **class** ViewIpTag **extends** TagSupport {

@Override

**public** **int** doStartTag() **throws** JspException {

//标签的调用时通过服务器来调用的

//request

HttpServletRequest request = (HttpServletRequest)**this**.pageContext.getRequest();

//得带out

JspWriter out = **this**.pageContext.getOut();

String ip = request.getRemoteAddr();

**try** {

out.println(ip);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

**return** **super**.doStartTag();

}

## 3.这样我们的jsp文件中的java文件变成了一个tag标签类文件，下面就是为了我们的标签处理器配置一个标签名，这个需要一个tId文件，来进行配置，这个文件的位置在WEB-INF文件夹下面

### 3.1。这个tId文件

<?xml version="1.0" encoding="UTF-8" ?>

<taglib 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-jsptaglibrary\_2\_1.xsd"

version="2.1">

<description>JSTL 1.2 core library</description>

<tlib-version>1.2</tlib-version>

<short-name>c</short-name>

<uri>http://java.sun.com/jsp/jstl/core</uri>

<tag>

<name>viewIP</name>

<tag-class>zidingyibiaoqian.ViewIpTag</tag-class>

<body-content>empty</body-content>

</tag>

</taglib>

### 3.2 uri绑定 这个标签

<short-name>ha</short-name>

<uri>http://www.zhangyujin.cn</uri>

### 3.3 jsp中开始通过这个uri导入这个标签

<%@taglib uri=*"http://www.zhangyujin.cn"* prefix=*"zhang"*%>

### 3.3下面的使用就是

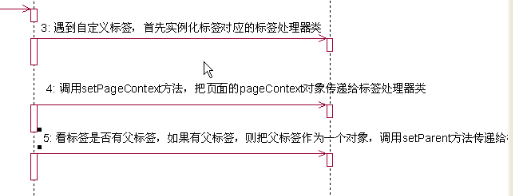
<body>

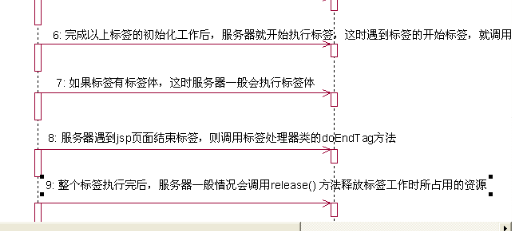
我的id是

<zhang:viewIP/>

</body>

# 二、自定义标签的执行原理





5.然后完成上面的初始化工作后，服务器开始执行标签，遇到开始标签 就调用dostart方法

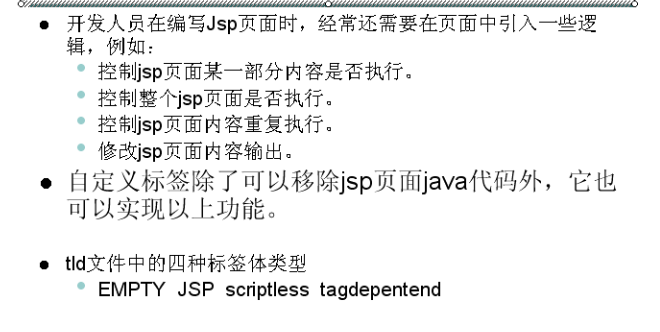
6.然后就开始，这时服务器一帮会有执行标签体

7.然后服务器遇到jsp页面结束标签，则调用标签树立器的doEndtag（）方法

8.整个标签执行完后，服务器一般情况会调用release方法来释放资源

9.然后接着执行之后的jsp标签

# 三、自定义标签的功能（非常重要）



## 1. 控制某部分内容是不是输出

**public** **class** kongzhi **extends** TagSupport {

@Override

**public** **int** doStartTag() **throws** JspException {

**return** Tag.*EVAL\_BODY\_INCLUDE*;

}

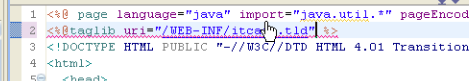
}

### 2.然后配置好tId文件的内容uri，和其他的东西

设置标签体，这个是jsp页面的文件

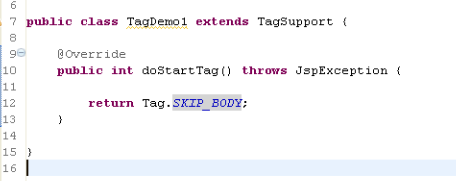


### 3.配置好了上面的东西之后，然后通过上面的uri开始进行jsp的引入（下面的文件可以使用路径这个是直接告诉这个标签是在那个文件夹中，也可以像之前的那样使用文件的uri地址都是可以的）





## 5.下面这个是不能够显示部分内容



## 6.控制整个jsp是不是输出

**public** **class** KZpage **extends** TagSupport {

**public** **int** doEndTag() **throws** JspException {

//设置是不是显示整个jsp

**return** Tag.*SKIP\_PAGE*;

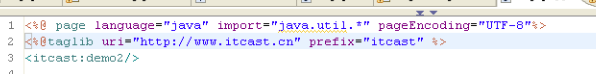
}

}

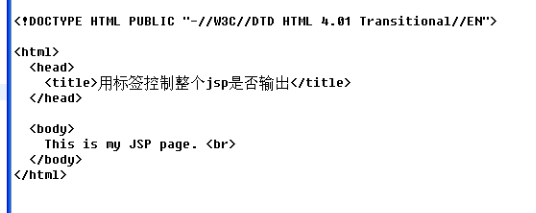
### 6.1这个标签是没有标签体的，所以是empty



### 6.2然后我们的jsp中应该加入这个东西



### 6.3这个时候运行的是不会执行的jsp的，没有源代码（看下面）



## 7.开始控制部分内容重复输入 （这个类是执行5次）

//控制标签体执行5次

**public** **class** KZtimes **extends** TagSupport {

**int** a = 5;

//这个我们需要覆盖两个方法

**public** **int** doStartTag() **throws** JspException {

**return** Tag.*EVAL\_BODY\_INCLUDE*;

}

**public** **int** doAfterBody() **throws** JspException {

a--;

**if**(a>0){

**return** IterationTag.*EVAL\_BODY\_INCLUDE*;

}**else**{

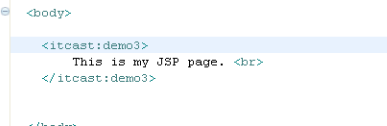
**return** IterationTag.*SKIP\_BODY*;

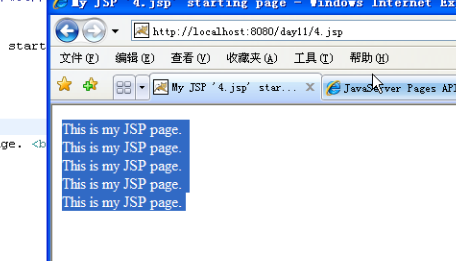
}

}

}

### 2.然后tId配置(这个是jsp的)





## 8.用标签修改jsp页面内容

**public** **class** KZxiugai **extends** BodyTagSupport{

**public** **int** doStartTag() **throws** JspException {

**return** BodyTag.*EVAL\_BODY\_BUFFERED*;

}

**public** **int** doEndTag() **throws** JspException {

//得到标签体

BodyContent bc = **this**.getBodyContent();

String contentString = bc.getString();

//将内容变成大写的,然后再输出

contentString = contentString.toUpperCase();

**try** {

**this**.pageContext.getOut().write(contentString);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

**return** Tag.*EVAL\_PAGE*;

}

}

### 2.这个标签体也是jsp的