教育部大学计算机课程改革项目规划教材

丛书主编 卢湘鸿

Java Web 编程技术实用教程

金百东 刘德山 编著

清华大学出版社

教育部大学计算机课程改革项目规划教材

丛书主编 卢湘鸿

Java Web 编程技术实用教程

金百东 刘德山 编著

清华大学出版社
北京

内 容 简 介

本书全面而又系统地介绍了 Java Web 编程开发技术。其中，JSP 部分包含基本语法、内置命令、JavaBean、Servlet、自定义标签库、配置文件、反射与注解等知识；JavaScript 部分包括函数、数组、面向对象技术、DOM 应用等知识；Ajax 部分包括局部刷新技术、XMLHttpRequest 对象、级联 Ajax、类在 Ajax 中的应用等知识。本书注重应用，每章都包含大量示例和详细的结果分析，旨在使读者夯实基础，提高综合运用 Web 各项技术编程能力，学会软件编程的思考方法。

本书可作为专业技术人员、大专院校计算机专业本科生的教材或参考书，对进一步学习 Struts，理解其实质也有一定的指导意义。

图书在版编目（CIP）数据

Java Web 编程技术实用教程/金百东，刘德山编著. --北京：清华大学出版社，2016
教育部大学计算机课程改革项目规划教材
ISBN 978-7-302-43575-4

Ⅰ. ①J… Ⅱ. ①金… ②刘… Ⅲ. ①JAVA 语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2016)第 082034 号

责任编辑：谢　琛
封面设计：常雪影
责任校对：焦丽丽
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954
印 刷 者：清华大学印刷厂
装 订 者：三河市吉祥印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21.75　　**字　　数**：528 千字
版　　次：2016 年 7 月第 1 版　　**印　　次**：2016 年 7 月第1次印刷
印　　数：1～2000
定　　价：44.50 元

产品编号：068418-01

前言

随着网络技术的飞速发展，Web 编程变得越来越重要，出现了许多关于 Web 的书籍。从教材角度来讲，绝大多数是关于 JSP 内容的。对 Web 编程而言，这是远远不够的。Web 编程的特点是综合性强，因此本书增加了有关 JavaScript 及 Ajax 知识的论述，这是作者写本书的一个主要初衷。

全书共分 10 章：第 1～8 章主要介绍 JSP 知识，第 9 章介绍 JavaScript 知识，第 10 章介绍 Ajax 知识，具体内容如下所示：

第 1 章介绍 JSP 开发环境及最简单的“Hello world”程序。

第 2 章介绍 JSP 基本语法知识，指令标签及动作标签的用法。

第 3 章介绍 JSP 内置命令知识。主要包括 request、response、session、application、out 五个常用内置命令的用法，session、application 的区别，利用内置命令如何解决中文乱码问题等内容。

第 4 章介绍 JavaBean 知识。主要包括 JavaBean 的特点，普通方式及动作标签应用 Bean 的区别，Bean 的作用域，动作标签在表单赋值中的作用，session 及 application 内置命令的简易仿真等内容。

第 5 章介绍 Servlet 知识。主要包括与 JSP 的关系，Servlet 常用类与接口，Servlet 中的 request、session、application 通信，过滤器，监听器，cookie 操作等内容。

第 6 章介绍一些稍难的综合示例。主要包含文件上传、下载，邮件发送、接收，数据库操作功能等内容。除了必需的数据库驱动包外，本章中示例没有采用任何第三方组件实现相应功能，旨在提高读者思维能力，增加编程兴趣。

第 7 章介绍自定义标签技术。主要内容包括自定义标签的作用，利用 TagSupport、BodyTagSupport、SimpleTagSupport 类及标签描述库创建自定义标签的方法，较新的 JSP2.0 支持的 tag 自定义标签实现方法。

第 8 章介绍配置文件、反射与注解技术。主要内容包括利用 Properties 解析文本文件和 XML 文件，利用 JDOM 解析 XML 文件；讲述了反射基本原理、反射调用构造方法和实例函数的基本技术，利用反射和配置文件技术实现了一个小型的 Web 应用框架；讲述了注解与配置文件的关系，元注解作用，自定义注解编程方法。

第 9 章介绍 JavaScript 知识。由于 JavaScript 与 Java 有许多相似的地方，因此采用了对比和增量的讲授方法，突出讲解了 JavaScript 与 Java 的不同之处。着重讲解了基本语法、函数、数组、面向对象技术、Web 消息事件、DOM 应用、类与 UI、定时器、系统对话框等具体内容。

第 10 章介绍 Ajax 知识。主要介绍局部刷新技术，XMLHttpRequest 对象，Ajax 返回 HTML 页面和 XML 数据的处理方法，特殊字符的 URI 参数编码技术，级联 Ajax 技术，类在 Ajax 中的应用。

本书内容循序渐进，采取实例驱动讲授方式，所有示例复制下来编译后就可以运行，许多题目是笔者多年 Web 编程经验的总结，实用性较强。示例前因后果都做了必要的说明，对一些稍难的题目，对其设计思想也做了相应的论述，帮助读者加深理解。

本书第 3、5、6、8、10 章由金百东编写，第 1、2、4、7、9 章由刘德山编写。因本书程序较多，故全书变量均用正体。

由于作者水平有限，时间紧迫，书中难免有疏漏之处，恳请广大读者批评指正。

作　者

2016 年 1 月

目录

第 1 章 JSP 介 绍

1.1 JSP 简介

JSP 是 Java Server Pages 的缩写，是由 Sun 公司倡导、多家公司参与，于 1999 年推出的一种动态网页技术标准。JSP 是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术，利用这一技术可以建立安全的、跨平台的先进动态网站，这项技术还在不断地被更新和优化。

需要强调的是：要想真正地掌握 JSP 技术，必须要有较好的 Java 语言基础。

1.2 开发环境

本书采用的开发环境是：JDK7＋Eclipse＋Tomcat7.0。Tomcat 是一个免费的开源的 Servlet 容器，它是 Apache 基金会的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有了 Sun 的参与和支持，最新的 Servlet 和 JSP 规范总能在 Tomcat 中得到体现。本节主要讲述 Eclipse 环境下配置 Tomcat 的步骤。

(1) 下载并安装 Tomcat7.0，假设安装目录是 D:/Tomcat7，其余均是默认安装即可。

(2) 打开 Eclipse，在菜单中选择 Window→Preferences，出现图 1-1 所示的对话框。

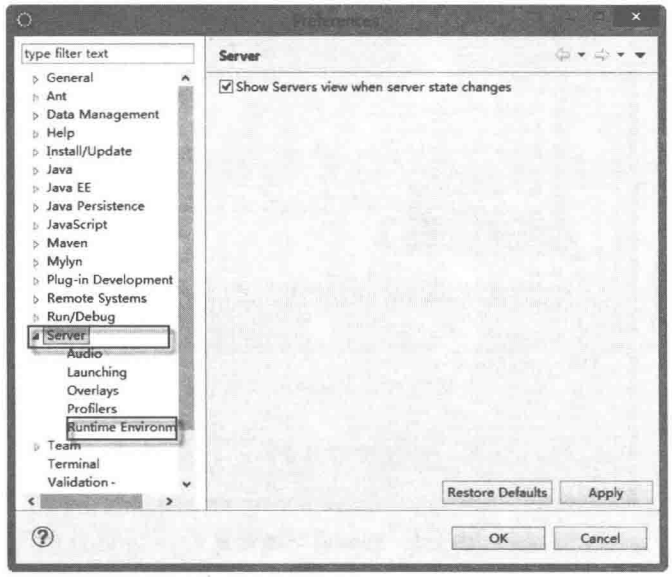

图 1-1 Tomcat 环境配置 1

(3) 选择 Server→Runtime Environment 项后，出现图 1-2 所示的对话框。

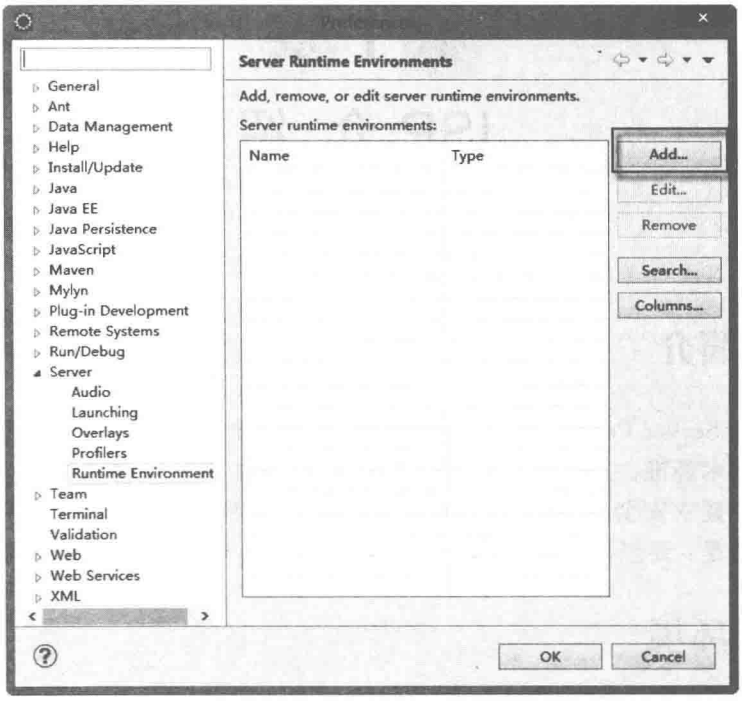

图 1-2 Tomcat 环境配置 2

(4) 单击 Add 按钮后，出现图 1-3 所示的对话框。

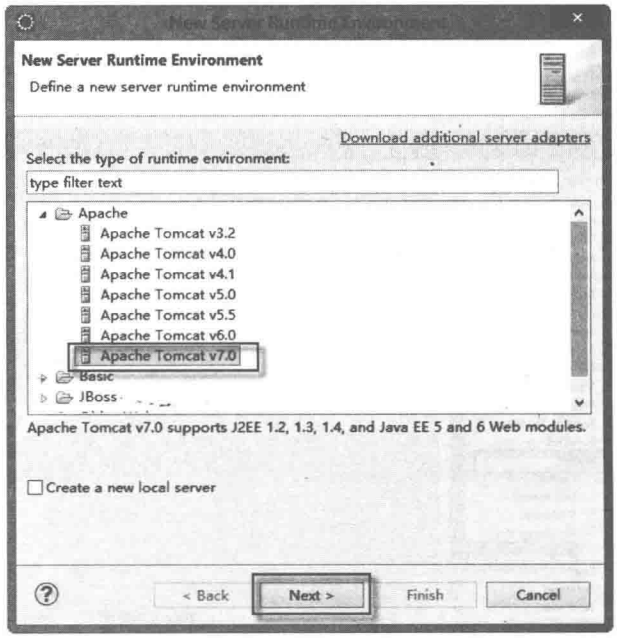

图 1-3 Tomcat 环境配置 3

选择 Apache Tomcat v7.0，然后单击 Next 按钮，出现图 1-4 所示的对话框。

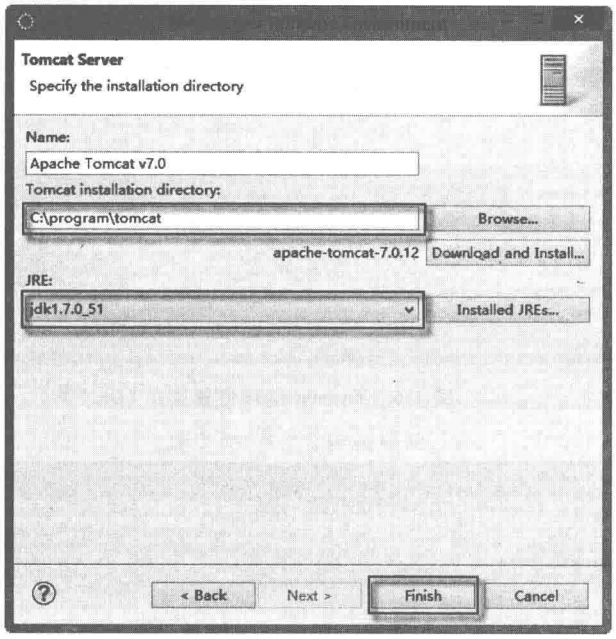

图 1-4　Tomcat 环境配置 4

首先单击 Browse 按钮，设置 Tomcat 安装目录；然后从下拉菜单中选择 JRE，最后单击 Finish 按钮即可。

那么，如何在 Eclipse 下运行 Tomcat 呢？步骤如下所示。

(1) 依次选择 Window→Show View→Servers，则出现 Servers 选项卡，如图 1-5 所示。

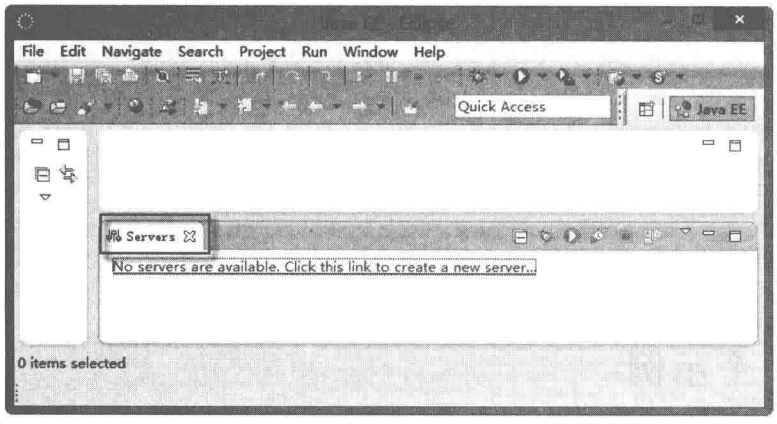

图 1-5　Tomcat 运行设置 1

(2) 在 Servers 选项卡空白处右击，出现浮动菜单，依次选择 New→Server 项后，按要求完成操作，如图 1-6 所示。

单击“方块”所示的位置可启动或停止 Tomcat 服务器的运行，运行成功的控制台界面如图 1-7 所示。

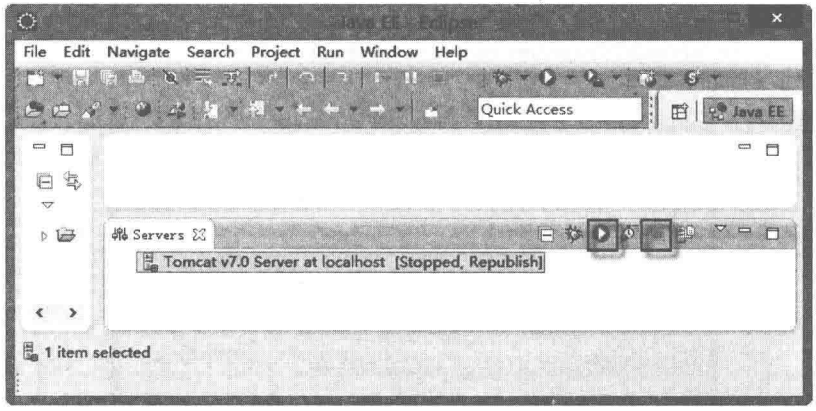

图 1-6 Tomcat 运行设置 2

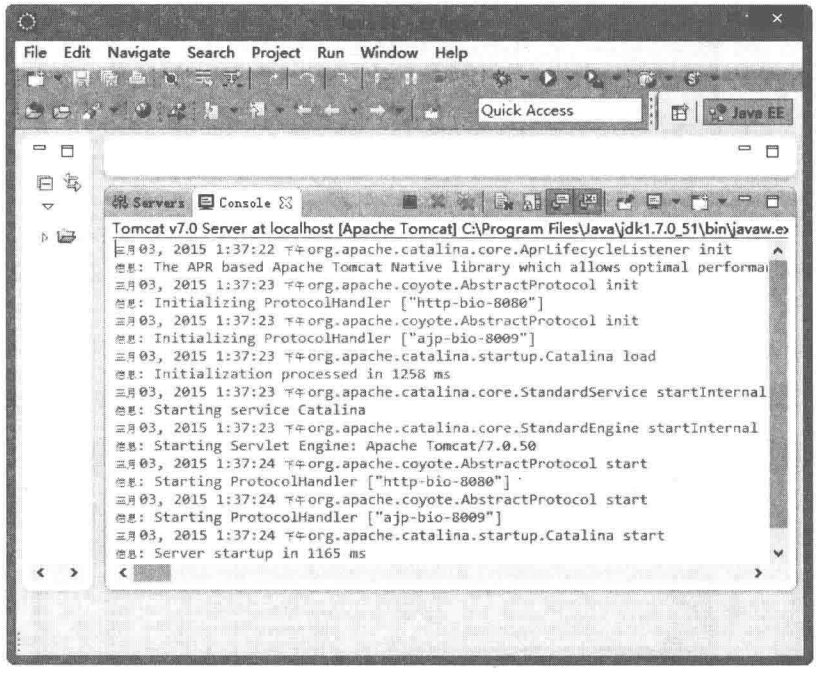

图 1-7 Tomcat 运行图

1.3 第 1 个示例

【例 1-1】 “Hello world”程序。

与建立 Java Project 工程步骤相似，建立 Dynamic Web Project 工程，工程名称为 chap1，如图 1-8 所示。

选中 WebContent 并右击，在浮动菜单中依次选择 New→JSP File，将文件命名为 hello.jsp。在该文件编辑区输入最简单的 JSP 代码，如图 1-9 所示。

单击“方块”指示的按钮，则执行结果如图 1-10 所示。

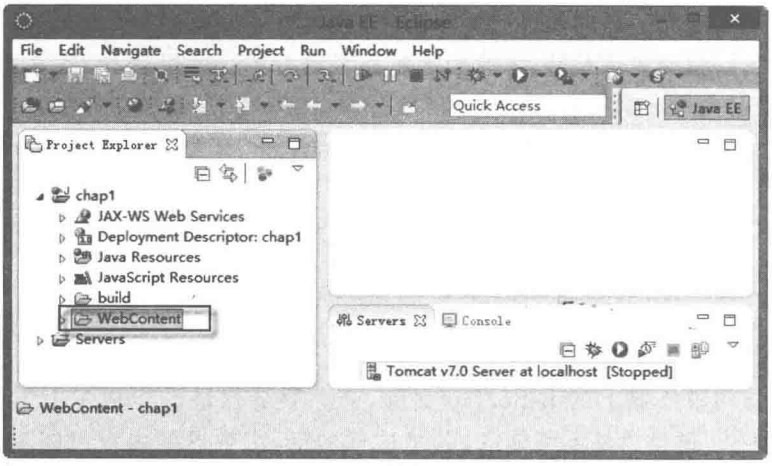

图 1-8　建立 JSP 工程图

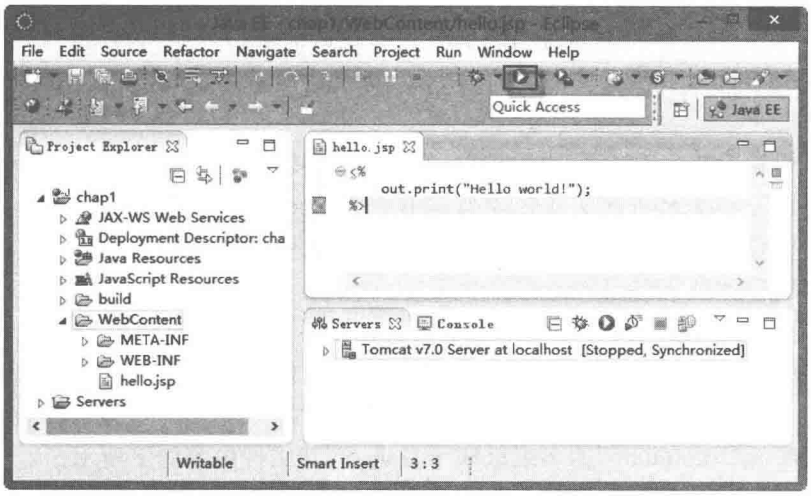

图 1-9　JSP 代码编辑区

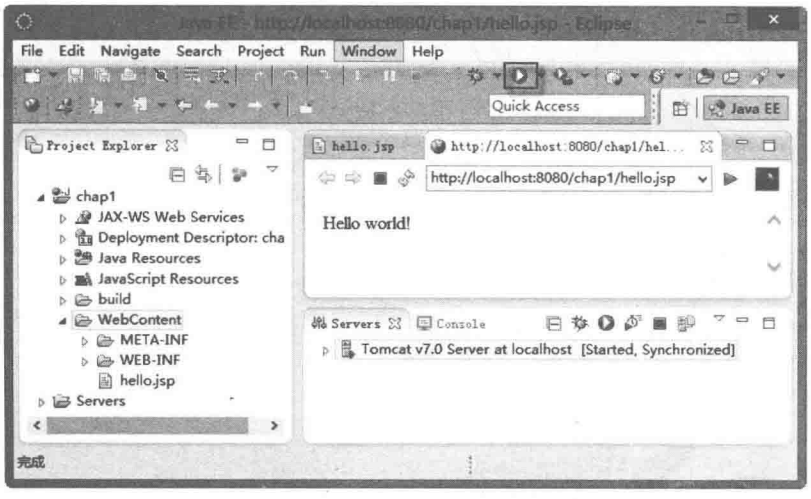

图 1-10　JSP 代码执行图

注意：若要执行某 JSP 文件，一定将鼠标焦点切换到该文件编辑区域，再单击“方块”内的执行按钮。当按钮响应运行后，首先判断服务器运行否。若未运行，则启动 Tomcat 服务器，之后运行 JSP 文件；若已运行，则直接运行该 JSP 文件。也就是说 Tomcat 服务器在 Web 应用中仅启动一次，而 JSP 文件可运行多次。当然，本文仅列出了运行 JSP 文件的一种方式，还有其他方式，就不一一列举了。

1.4 JSP 运行流程

Web 应用涉及客户端(浏览器)和服务器端，以 1.3 节的示例为依据，假设在某一时刻，有成千上万人访问服务器上的 hello.jsp 文件，会在服务器端产生成千上万个与之匹配的 hello 内存对象吗？很显然是不可能的，若是这样的话，服务器的内存可能不久就会自行崩溃了。因此只能是这样的情况：hello 对象在服务器端仅创建一次，为多个客户共享。据此可得 JSP 文件运行详细流程如下：

(1) 客户通过浏览器向服务器端的 JSP 页面发送请求。

(2) JSP 引擎检查 JSP 文件对应的 Servlet 源代码是否存在，若不存在转向第(4)步，否则执行下一步。

(3) JSP 引擎检查 JSP 页面是否修改，若未修改，转向第(7)步，否则执行下一步。

(4) JSP 引擎将 JSP 页面文件转译为 Servlet 源代码(相应的.java 代码)。

(5) JSP 引擎将 Servlet 源代码编译为相应的字节码(.class 代码)。

(6) JSP 引擎加载字节码到内存。

(7) 字节码处理客户请求，并将结果返回给客户。

也就是说：必须根据.jsp 文件获得对应的.java 文件。以 1.3 节中的 hello.jsp 为例，其对应的源文件为 hello_jsp.java(即 XXX.jsp 对应 XXX_JSP.java)。该文件是可以查到的，在“工作空间目录/.metadata”的多级级联子目录下，其文件内容如下所示。

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
public final class hello_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {
  private static final javax.servlet.jsp.JspFactory _jspxFactory=
          javax.servlet.jsp.JspFactory.getDefaultFactory();
  private static java.util.Map<java.lang.String,java.lang.Long> _jspx_
dependants;
  private javax.el.ExpressionFactory _el_expressionfactory;
  private org.apache.tomcat.InstanceManager _jsp_instancemanager;
  public java.util.Map<java.lang.String,java.lang.Long>getDependants() {
    return _jspx_dependants;
  }
  public void _jspInit() {
    _el_expressionfactory=
```

```
        _ jspxFactory. getJspApplicationContext (getServletConfig ( ).
getServletContext()).getExpressionFactory();
    _jsp_instancemanager=
     org. apache. jasper. runtime. InstanceManagerFactory. getInstanceManager
(getServletConfig());
  }
  public void _jspDestroy() {}
   public void _jspService (final javax. servlet. http. HttpServletRequest
request, final
    javax.servlet.http.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException {
final javax.servlet.jsp.PageContext pageContext;
    javax.servlet.http.HttpSession session=null;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out=null;
    final java.lang.Object page=this;
    javax.servlet.jsp.JspWriter _jspx_out=null;
    javax.servlet.jsp.PageContext _jspx_page_context=null;
    try {
      response.setContentType("text/html");
      pageContext=_jspxFactory.getPageContext(this, request, response,
          null, true, 8192, true);
      _jspx_page_context=pageContext;
      application=pageContext.getServletContext();
      config=pageContext.getServletConfig();
      session=pageContext.getSession();
      out=pageContext.getOut();
      _jspx_out=out;
    out.print("Hello world!");
    } catch (java.lang.Throwable t) {
      if(! (t instanceof javax.servlet.jsp.SkipPageException)){
        out=_jspx_out;
        if(out ! =null && out.getBufferSize() ! =0)
          try { out.clearBuffer(); } catch (java.io.IOException e) {}
        if(_jspx_page_context ! =null) _jspx_page_context.handlePageException
        (t);
        else throw new ServletException(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```