《JavaEE 实验手册》

Java 教研室

版 本 1.3

文档提供：Java 教研室 孙丽萍

修 改 记 录

修改时间	修改人	修改内容
2009.8.25	刘战洪	文档创建
2011.7.1	孟双英	修改
2012.8.27	张立飞	修改
2016.3.7	孙丽萍	修改

目录

第三章 Servlet 模型（二）

一、内容概述

本章的教学内容是掌握 Servlet 的生命周期，也就是 Servlet 在 Web 服务器中被加载、实例化、初始化、执行以及销毁的整个过程。还要熟悉与 Servlet 相关的一些接口与类型的作用并且能够熟练的运用到自己的程序里。

二、实验一 掌握 Servlet 的初始化过程

2.1 实验目的

通过程序验证 Servlet 的初始化，并能牢固掌握 Servlet 的初始化过程。

2.2 准备

在进行开发之前，要保证开发工具的正确安装与配置。

2.3 实验步骤

步骤一：

创建“Dynamic Web Project”，项目名称为 01_init。

步骤二：

由 Eclipse 创建 Servlet 及其配置，Servlet 类的名称为 InitAndDestroy。

步骤三：

因为我们不关心客户端是使用 GET 还是 POST 方式发送的请求，所以对于这两种请求方式都进行同样的操作，可以通过在 doPost 方法中直接调用 doGet 方法来实现，doPost 的代码如下所示

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
       throws ServletException, IOException {
    this.doPost(request, response);
}
```

步骤四：

重写父类(GenericServlet)的 init 方法，用来验证 Servlet 什么时候被初始化。修改后的 init 方法的代码如下：

```
@Override
public void init(ServletConfig config) throws ServletException {
    System.out.println("InitAndDestroy Servlet is init...");
}
```

步骤五：

运行该项目，观察 Eclipse 控制台的输出信息(只有 Tomcat 启动信息)。

步骤六：

在浏览器中输入访问地址：

http://localhost:8080/01_init/InitAndDestroy

这时观察 Eclipse 控制台的输出信息，如*图 2-1*

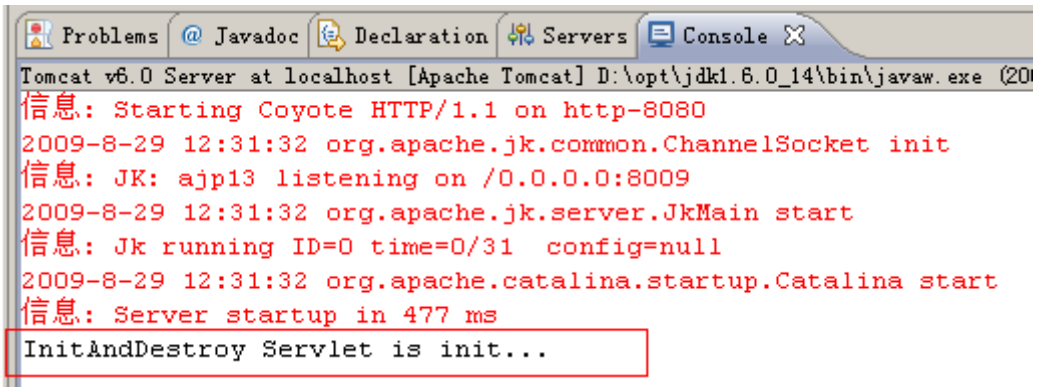

图 2-1

根据该结果，得出结论：该 Servlet 在第一次被访问的时候进行初始化。

步骤七：

另外启动一个浏览器(如 FireFox)，访问同一个 Servlet，控制台没有变化，说明 Servlet 只被初始化了一次。

步骤八：

停止 Tomcat，修改 web.xml 中该 Servlet 的描述，增加<load-on-startup>标签内容，修改后的 Servlet 描述如下：

```
<servlet>
  <description></description>
  <display-name>InitAndDestroy</display-name>
  <servlet-name>InitAndDestroy</servlet-name>
  <servlet-class>onest.dev.InitAndDestroy</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>InitAndDestroy</servlet-name>
  <url-pattern>/InitAndDestroy</url-pattern>
</servlet-mapping>
```

步骤九：

启动 Tomcat，不进行任何页面的访问，观察 Eclipse 控制台的输出，其中会有该 Servlet 被初始化的记录，如*图 2-2*

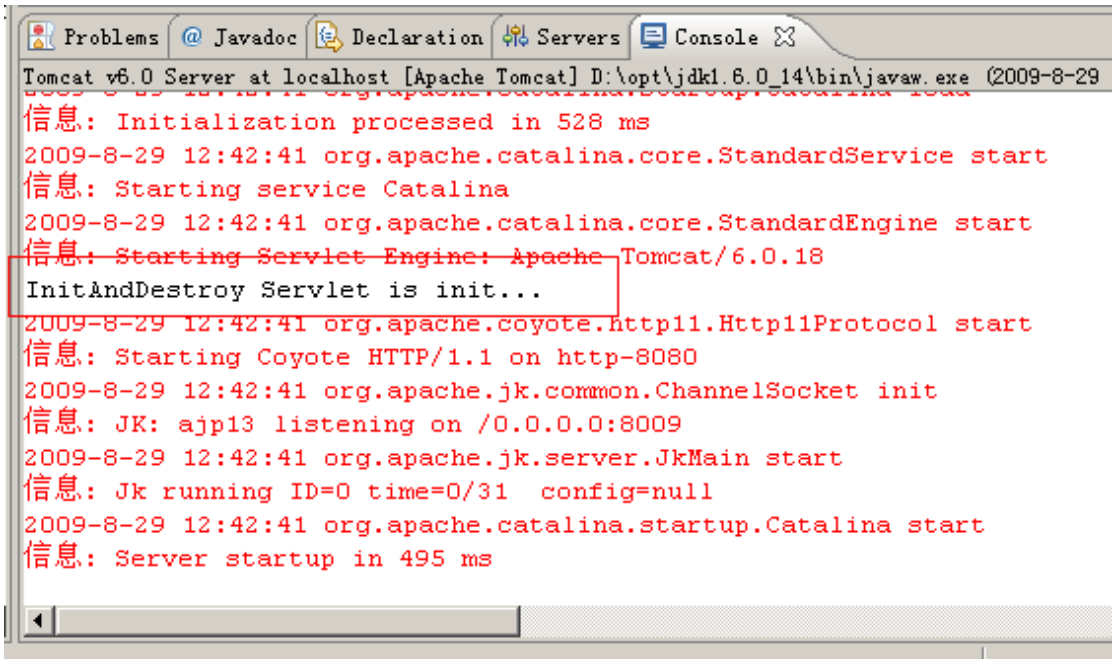

图 2-2

实验结果证明当 Servlet 的描述中增加<load-on-startup>标签后，在 Tomcat 启动的时候就会初始化该 Servlet。

2.4 实验结论

通过试验得出结论：Servlet 在第一次被访问的时候被初始化，或者通过修改 web.xml 相关 Servlet 的配置项，达到在 Servlet 容器启动的时候就对相关 Servlet 进行初始化的目的。

2.5 扩展

编写代码验证 Servlet 被销毁的过程。

三、实验二 掌握 ServletConfig 和 ServletContext 接口

3.1 实验目的

通过实验掌握 ServletConfig 和 ServletContext 接口的应用。

3.2 准备

该实验是在上一实验(实验一)的基础上完成。

3.3 实验步骤

步骤一：

修改 web.xml 中 InitAndDestroy 类的 Servlet 描述，修改后的 Servlet 描述如下：

```
<servlet>
    <display-name>InitAndDestroy</display-name>
    <servlet-name>InitAndDestroy</servlet-name>
    <servlet-class>onest.dev.InitAndDestroy</servlet-class>
    <init-param>
        <param-name>init.test</param-name>
        <param-value>hahahahaha</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>InitAndDestroy</servlet-name>
    <url-pattern>/InitAndDestroy</url-pattern>
</servlet-mapping>
```

要注意<init-param>标签的组成以及其所在位置。

步骤二：

将 InitAndDestroy 类的 init 方法修改成如下内容：

```
@Override
public void init(ServletConfig config) throws ServletException {
    System.out.println("InitAndDestroy Servlet is init...");
    System.out.println(config.getInitParameter("init.test"));
}
```

步骤三：

运行该项目，控制台输出如*图 3-1*

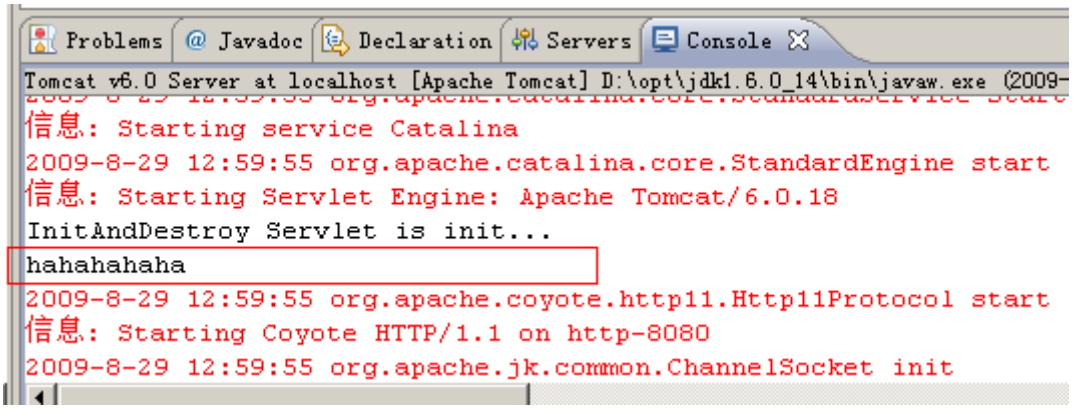

图 3-1

步骤四：

修改 web.xml，在根节点(<web-app>节点)中增加<context-param>标签，修改后的内容如下：

```
    <context-param>
        <param-name>context.text</param-name>
        <param-value>Hi,this ContextParam 's Value</param-value>
    </context-param>
```

注意该标签的组成及位置。

步骤五：

将 InitAndDestroy 类的 init 方法修改成如下内容：

```
@Override
public void init(ServletConfig config) throws ServletException {
    System.out.println("InitAndDestroy Servlet is init...");
    System.out.println(config.getInitParameter("init.test"));
    ServletContext servletContext = config.getServletContext();
    System.out.println(servletContext.getInitParameter("context.te
xt"));
}
```

步骤三：

停止 Tomcat 并重新运行该项目，控制台输出如*图 3-2*

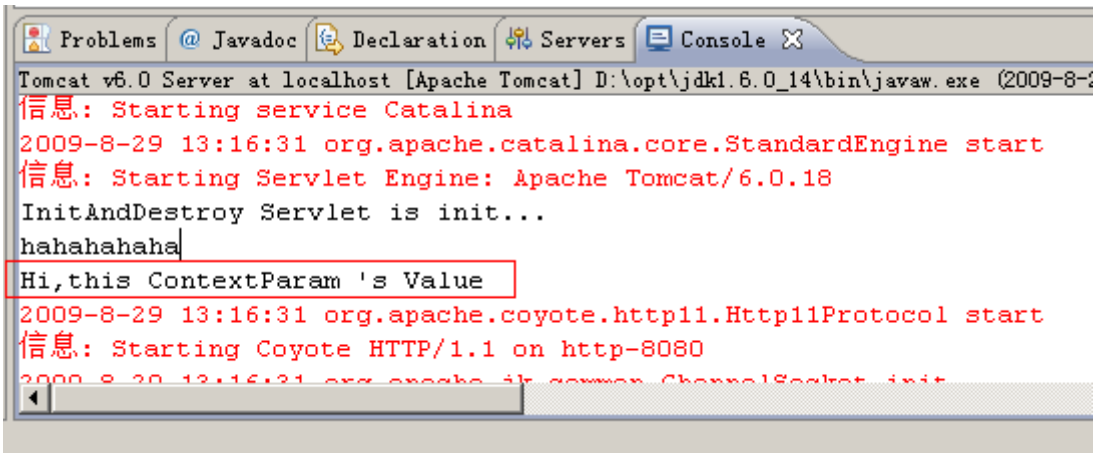

图 3-2

在控制台输出了<param-value>的值。

3.4 实验结论

可以通过 ServletConfig 接口的实例获得与 Servlet 相关的配置信息，通过 ServletContext 接口的实例获取整个 Web 应用的配置信息。

四、实验三 编写网站的访问记数器

4.1 实验目的

能利用 ServletContext 接口的功能实现现实的需求，编写网站的访问记数器。

4.2 功能描述

当有客户端(一个或者多个)访问同一网站时，访问同一个或者不同个网页(Servlet)都将访问了多少次的次数记录下来。

4.3 概要设计

需要在 Web 应用程序的全局环境中保存一个数值，当有 Servlet 被访问时，将该数据加一。

要在全局环境中保存数据，可以利用 ServletContext 接口，在 Web 服务器(Tomcat)启动时，将 ServletContext 的实例增加一个属性并初始值为 0，在其他的 Servlet 中，被访问到的话就将该属性的值增一。

4.4 编码实现

步骤一：

创建动态网站项目 01_context，新建初始化计数器的 Servlet 类 InitServlet。并且在该 Servlet 的 init 方法中进行 ServletContext 对象属性的设置与初始化，init 方法的代码如下：

```
@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    ServletContext servletContext = config.getServletContext();
    servletContext.setAttribute("requestNumber", 0);
}
```

步骤二：

为了保证该值在 Web 服务器(Tomcat)启动时就被初始化，需要修改 web.xml 中该 Servlet 的启动配置，修改后的内容如下

```
<servlet>
  <description></description>
  <display-name>InitServlet</display-name>
  <servlet-name>InitServlet</servlet-name>
  <servlet-class>onest.dev.InitServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

步骤三：

编写被访问的 Servlet 类 ServletA 和 ServletB，在其 service 方法(doGet 和 doPost 两个方法)可以读取记数器的值，并将计数器值增一。关键代码如下：

```
protected void doGet(HttpServletRequest request
        , HttpServletResponse response)
       throws ServletException, IOException {
    ServletContext servletContext = this.getServletContext();
    Object number = servletContext.getAttribute("requestNumber");
    int i = 0;
    if (number == null)
        servletContext.setAttribute("requestNumber", 0);
    else {
        i = new Integer(number.toString());
        i++;
        servletContext.setAttribute("requestNumber", i);
    }
    PrintWriter writer = response.getWriter();
    writer.println("I'm ServletA, the request number is " + i);
}
```

步骤四：

运行该项目，分别多次访问 ServletA 与 ServletB，观察输出结果。如*图 4-1*

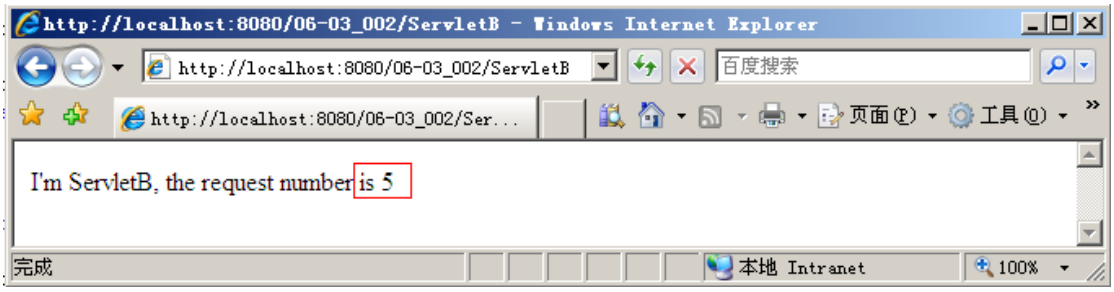

图 4-1

4.4 实验结论

ServletContext 是在 Web 服务器启动时就创建了一个该接口的实例，在整个 Web 应用中，有且只有一个 ServletContext 的实例，可以利用该实例的

get/setAttribute 方法保存全局范围的数据。

五、实验四 展示图片到客户端的浏览器

5.1 实验目的

Servlet 相关知识的应用，将服务器的 D 盘根目录下的 soft.jpg 展示到客户端的浏览器

5.2 功能描述

读取出服务器 D 盘根目录下的 soft.jpg 文件展示给访问该网站的客户。

5.3 概要设计

设计一

当有用户访问网站的某页面(Servlet)时，在该 Servlet 的执行方法中将服务器 D 盘根目录下的 soft.jpg 图片文件读入服务器的内存中。

设计二

将内存中的图片信息以流的方式输出给客户端浏览器。浏览器接收到流的信息需要以图片的格式进行展示。

设计三

在完成输出后，要将流进行关闭操作，释放其所占资源。

5.4 编码实现

步骤一：

使用 Eclipse 创建动态网站项目，并起名称。

[由学生完成]

步骤二：

创建 Servlet 类：ImageServlet，并实现读取图片文件、设置输出格式、将图片信息输出到客户端的代码。

关键部分代码如下：

```
protected void doGet(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {

    response.setContentType("image/jpg");
    ServletOutputStream outputStream = response.getOutputStream();
    File imageFile = new File("D:/slpfile/soft.jpg");
    InputStream fileInputStream = new FileInputStream(imageFile);
    int length = (int) imageFile.length();
    byte[] b = new byte[length];
    fileInputStream.read(b);
    outputStream.write(b);
    fileInputStream.close();
    outputStream.flush();
    outputStream.close();
}
```

其他代码及设置

[由学生完成]

步骤三：

测试

[由学生完成]

注意：客户端浏览器需要选择使用 FireFox。

5.5 实验结论

通过此实验，让同学掌握使用 Servlet 处理响应时，设置 ContentType 属性，以用来输出非文本数据。