JavaWeb简介

> ## 本节目标
>
> 1.了解Web编程模型 2.理解HTTP协议 3.了解动态Web技术

1.Web简介

1.1 Web应用

Web程序是一种可以通过浏览器访问的应用程序。Web应用的一个最大好处是用户只需要有浏览器即可访问应用程序，无需安装其它软件。我们经常访问的电商网站，如：淘宝、京东、门户网站、新浪、网易等都属于Web应用程序。

1.2 B/S模型

B/S模型也称之为浏览器-服务器模型，对应的是C/S模型即就是客户端-服务器模型。

B/S模型的优点：

- 具有分布式特点，可以随时随地进行查询、浏览等业务处理。
- 业务扩展简单方便，通过增加网页即可增加服务功能。
- 维护简单方便，只需要改变网页，即可实现所有用户的同步更新。
- 开发简单，共享性强

1.3 HTTP协议

1.3.1 什么是HTTP协议

HTTP: HyperText Transfer Protocol 超文本传输协议，其在RFC2616中定义，是当今互联网上应用最为广泛的一种网络协议。

1.3.2 HTTP协议的特点

- Web应用程序的基础
- 基于TCP/IP协议的应用层协议
- HTTP协议的应用缺省端口是80
- HTTP协议特点是：无状态、无连接（HTTP/1.1版本，即当前版本已经支持长连接）

> 备注：推荐书籍《图解HTTP》 日·上野宣 著 于均良 译

2.动态Web

Web服务器只能向客户提供静态资源，但是我们所见到的网站都不可能是静态网页，不然也不会那么多姿多彩。动态Web技术就显的特别重要了。

2.1 Web服务器

Web服务器是“通过HTTP协议处理请求的计算机系统”，擅长提供静态的Web页面，而**不做动态内容**和**不在服务器上保持数据**，而此时如果我们需要一个动态的页面就需要一些辅助程序来完成。

常用的主流Web服务器有：

- Nginx
- Apache HTTP Server
- Apache Tomcat 侧重点在Web容器

2.1 动态技术

- CGI (Common Gateway Interface）：公共网关接口
- PHP (HyperText Preprocessor）：超文本处理器
- ASP (Active Server Page)：活动服务器页面
- JSP (Java Server Pages)：Java服务器页面，其是简化的Servlet设计

3. Servlet技术

3.1 什么是Servlet

Servlet（Server Applet）是Java Servlet的简称，称为小服务程序，即用Java编写的服务器端程序；主要功能在于交互式地浏览和修改数据，生成动态Web内容。

狭义的Servlet是指Java语言实现的一个接口，广义的Servlet是指任何实现了Servlet接口的类。

一般情况下，我们将Servlet理解为前者。Servlet运行于支持Java的应用服务器中。从原理上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于HTTP协议的Web服务器。

3.2 Servlet与CGI对比

3.2.1 两者对比

- 与传统的CGI和许多其它类似CGI的技术相比，Java Servlet**具有更高的效率**，**更容易使用**，**功能更强大**，**具有更好的可移植性**，**更节省投资**。在未来的技术发展过程中，Servlet有可能彻底取代CGI。
- 在传统的CGI中，每个请求都要启动一个新的**进程**，如果CGI程序本身的执行时间较短，启动进程所需要的开销很可能反而超过实际执行时间。而在Servlet中，每个请求由一个轻量级的Java线程处理。
- 在传统CGI中，如果有N个并发请求同时访问同一个CGI程序，则该CGI程序的代码在内存中重复装载了N次；而对于Servlet，处理请求的是N个线程，只需要一份Servlet类代码。在性能优化方面，Servlet也比CGI有着更多的选择。

3.2.2 Servlet的优势

- **方便**

Servlet提供了大量的实用工具例程，例如自动地解析和解码HTML表单数据、读取和设置HTTP头、处理Cookie、跟踪会话状态等。

- **功能强大**

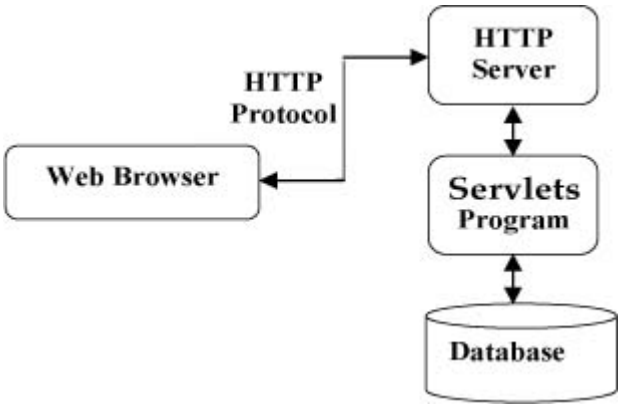
在Servlet中，许多使用传统CGI程序很难完成的任务都可以轻松地完成。例如，Servlet能够直接与Web服务器交互，而普通的CGI程序不能。Servlet还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

- **可移植性好**

Servlet用Java编写，Servlet API具有完善的标准。因此为企业级应用编写的Servlet无需任何实质上的改动即可移植到Apache、Microsoft IIS等Web服务器，几乎所有的主流服务器都直接或通过插件支持Servlet。

3.3 Servlet架构

下图展示了 Servlet 在 Web 应用程序中的位置。



3.4 Servlet的任务

- 读取客户端（浏览器）发送的显式的数据。比如：网页上的 HTML 表单，自定义的 HTTP 客户端程序的表单。
- 读取客户端（浏览器）发送的隐式的 HTTP 请求数据。比如：cookies、媒体类型，浏览器能理解的压缩格式等。
- 处理数据并生成结果。这个过程可能需要访问数据库；执行业务层逻辑；调用 Web 服务，或者直接计算得出对应的响应数据。
- 发送显式的数据（即文档）到客户端（浏览器）。该文档的格式可以是多种多样的，包括文本文件（HTML或XML）、二进制文件（GIF图像）、Excel等。
- 发送隐式的HTTP响应到客户端（浏览器）。这包括告诉浏览器或其它客户端被返回的文档类型（例如HTML），设置cookies和缓存参数，以及其它类似的任务。

3.5 Servlet规范

Java Servlet是运行在带有支持 Java Servlet 规范的解释器的web容器上的 Java 类。Servlet 可以使用 `javax.servlet` 和 `javax.servlet.http` 包创建，它是 JavaEE的标准组成部分，JavaEE是支持大型开发项目的Java类库的扩展版本。

> 备注：本课程内容基于Java Servlet 3.1规范

4. Web容器

4.1 什么是容器

Servlet是没有main()方法，它受控于另外一个Java应用，这个Java应用成为容器。

常见的Java Web容器有：

- Tomcat
- Jetty : Eclipse Jetty提供Web服务和Servlet容器

4.2 容器能提供什么

容器是用来管理和运行Servlet，那么容器都能提供那些功能呢?

- 通信支持

  利用容器提供的方法，能够轻松让Servlet与Web服务器对话，无需自己建立ServerSocket，监听端口，创建流等操作。容器知道自己与Web服务器之间的协议，所以Servlet不必担心Web服务器（如：Apache，Nginx）与 自己的Web之间的API，只需要考虑实现业务即可。

- 生命周期管理

  容器控制着Servlet的生与死。它会负责加载类，实例化和初始化Servlet，调用Servlet的方法，并使Servlet实例能够被垃圾回收。有了容器的控制，开发者就不需要太多的考虑资源管理。

- 多线程支持

  容器会自动地为它接收到的每个Servlet请求创建一个新的Java线程。针对客户端请求，如果Servlet已经运行完相应的HTTP服务方法，该线程就会结束。这些并不是说不用考虑线程安全性，还是会遇到同步问题。不过，由于服务器创建和管理多线程来处理多个请求，这些都可以让开发者省去不少工作。

- 声明方式实现安全

  利用容器，可以使用XML部署描述文件来配置安全性（Servlet 3.0开始可以使用注解描述），而不必将其硬编码写到Servlet或者其它类中。

- JSP支持

  容器可以将JSP代码翻译成Java代码。

4.3 Tomcat介绍

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（现在已属于Oracle）和其它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。

4.4 Tomcat安装

- 官方网站
- 下载安装包(版本 8.5.32)
- 解压下载的压缩包至本地磁盘，即可完成安装

4.5 Tomcat目录说明

下面表格是Tomcat容器安装目录下的每个目录的说明：

目录	说明
/bin	存放各种操作系统下用于启动和停止Tomcat的命令文件
/conf	存放Tomcat服务器各种配置文件
/lib	存放Tomcat服务器所需的依赖Jar文件
/logs	存放Tomcat的日志文件
/temp	存放Tomcat运行时的临时文件
/webapps	发布Web应用时，默认会将Web应用的发布到此目录中
/work	Tomcat把由JSP成的Servlet放于此目录下

5. 项目示例

5.1 准备项目

- 创建Maven项目

  在pom.xml文件中设置打包类型为: `war` 表示**Web application Archive**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.bittech.javaweb</groupId>
    <artifactId>javaweb-case</artifactId>
    <version>1.0.0</version>
    <!-- 设置打包类型 -->
    <packaging>war</packaging>
</project>
```

- 添加依赖

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
```

- 创建webapp目录

  `src/main/webapp`

- 创建WEB-INF目录

  `src/main/webapp/WEB-INF`

- 创建web.xml文件

  JavaWeb应用程序的描述文件，可以从Tomcat的安装目录的conf/web.xml获取样例；

  下面是web.xml的内容，文件位于：`src/main/webapp/WEB-INF/web.xml`。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                      http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1">

</web-app>
```

5.2 准备内容

- 创建静态内容：HTML页面(如：index.html)

  静态页面可以归档到webapp目录下，就可以通过URL地址访问到页面

- 创建动态内容：Servlet类并配置Servlet

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class IndexServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("text/html; charset=UTF-8");
        PrintWriter writer = resp.getWriter();
        writer.append("<html>")
                .append("<head>")
                .append("</head>")
                .append("<body>")
                .append("<h1>")
                .append("Hello Java Web Power by Servlet")
                .append(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new
Date()))
                .append("</h1>")
                .append("</body>");
    }
}
```

web.xml中配置Servlet

```
<!-- Servlet信息 -->
<servlet>
```

```
    <!-- Servlet部署的名称-内部的秘密名称 -->
    <servlet-name>IndexServlet</servlet-name>
    <!-- Servlet真正的类名 -->
    <servlet-class>com.bittech.javaweb.web.IndexServlet</servlet-class>
</servlet>
<!-- Servlet映射 -->
<servlet-mapping>
    <!-- Servlet部署的名称 -->
    <servlet-name>IndexServlet</servlet-name>
    <!-- 客户端访问的URL名称 -->
    <url-pattern>/index</url-pattern>
</servlet-mapping>
```

web.xml文件称之为Web应用程序的的描述文件，比如：Servlet，欢迎页面，错误处理，以及Filter，Listener等等，JavaWeb中的资源配置，都可以通过在该文件中进行配置描述，容器才可以识别到各类资源。自Java Servlet API 3.0开发，描述信息可以通过注解来完成，本课程中主要以配置为主，注解为辅。

5.3 打包部署

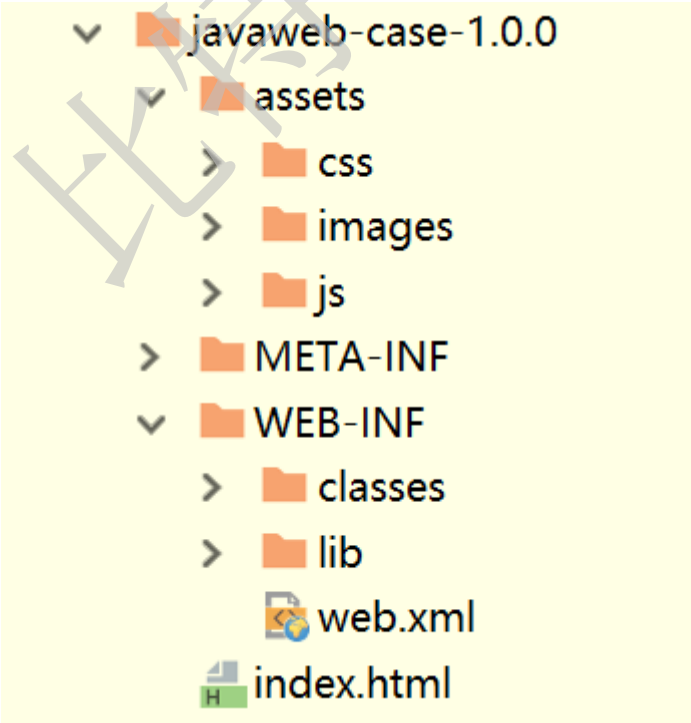
- 打包

  通过maven命令进行打包 `mvn package`

- 部署

  将生成的war包如：`javaweb-case-1.0.0.war` 修改名称为 `case.war` 复制到 `${TOMCAT_HOME}/webapps` 目录下

- 应用目录结构



  - assets：静态资源文件，比如：css，javascript ,images
  - META-INF：应用元信息
  - WEB-INF：**应用内部能够访问的资源**

        - classes：Java类编译之后的classes文件
        - lib：应用依赖的jar包
        - web.xml ： 应用部署描述文件
    - index.html：应用外部能访问的静态文件

5.4 启动访问

- 进入到Tomcat安装目录的bin目录

  Windows环境执行`startup.bat`启动Tomcat容器

  Linux环境执行`startup.sh`启动Tomcat容器

- 打开浏览器访问

  静态页面：`http://localhost:8080/case/index.html`

  动态页面：`http://localhost:8080/case/index`

总结

知识块	知识点	分类	掌握程度
HTTP协议	1.HTTP协议特点	思想型	掌握
Servlet	1.Servlet特点 2.Servlet规范 3.Servlet与CGI对象	思想型	掌握
Web容器	1.Tomcat安装部署配置 2. 理解Web容器	实战型	掌握
Web应用	1.创建Web项目	实战型	掌握

6.扩展

- Tomcat的服务器的端口号修改
- Tomcat的管理界面演示，角色和用户配置