传智播客 www.itcast.cn —高级软件人才实作培训专家！

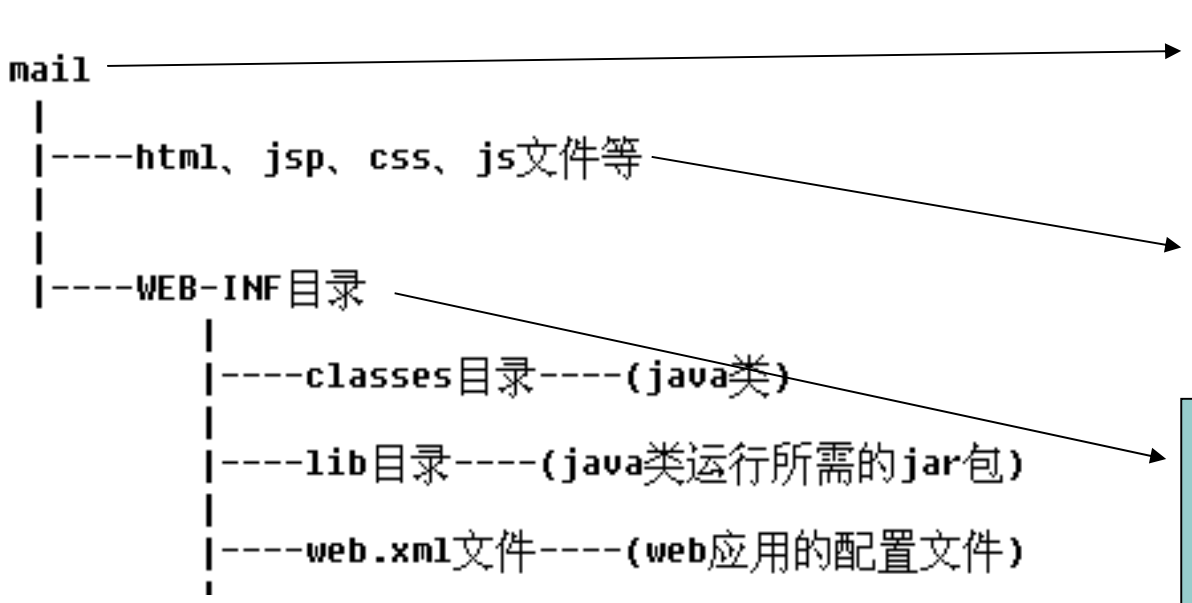
Servlet

传智·董平

Servlet简介

- Servlet是sun公司提供的一门用于开发动态web资源的技术。
- Sun公司在其API中提供了一个servlet接口，用户若想开发一个动态web资源(即开发一个Java程序向浏览器输出数据)，需要完成以下2个步骤：
  - 编写一个Java类，实现servlet接口。
  - 把开发好的Java类部署到web服务器中。
- 快速入门，用servlet向浏览器输出“hello servlet”。
  - 阅读Servlet API，解决两个问题：
  - 输出hello servlet的java代码应该写在servlet的哪个方法内？
  - 如何向IE浏览器输出数据？

Servlet在web应用中的位置



Web应用所在目录

这些文件一般存在web应用根目录下，**根目录下的文件外界可以直接访问**

java类、jar包、web应用的配置文件存在这个目录下，**该目录下的文件外界无法非法直接访问，由web服务器负责调用**

提示：按照一种约定俗成的称呼习惯，通常我们也把实现了servlet接口的java程序，称之为Servlet。

第一个Servlet的编写

动手练习

完成目标：利用Servlet向客户端输出”HelloSerlvet”

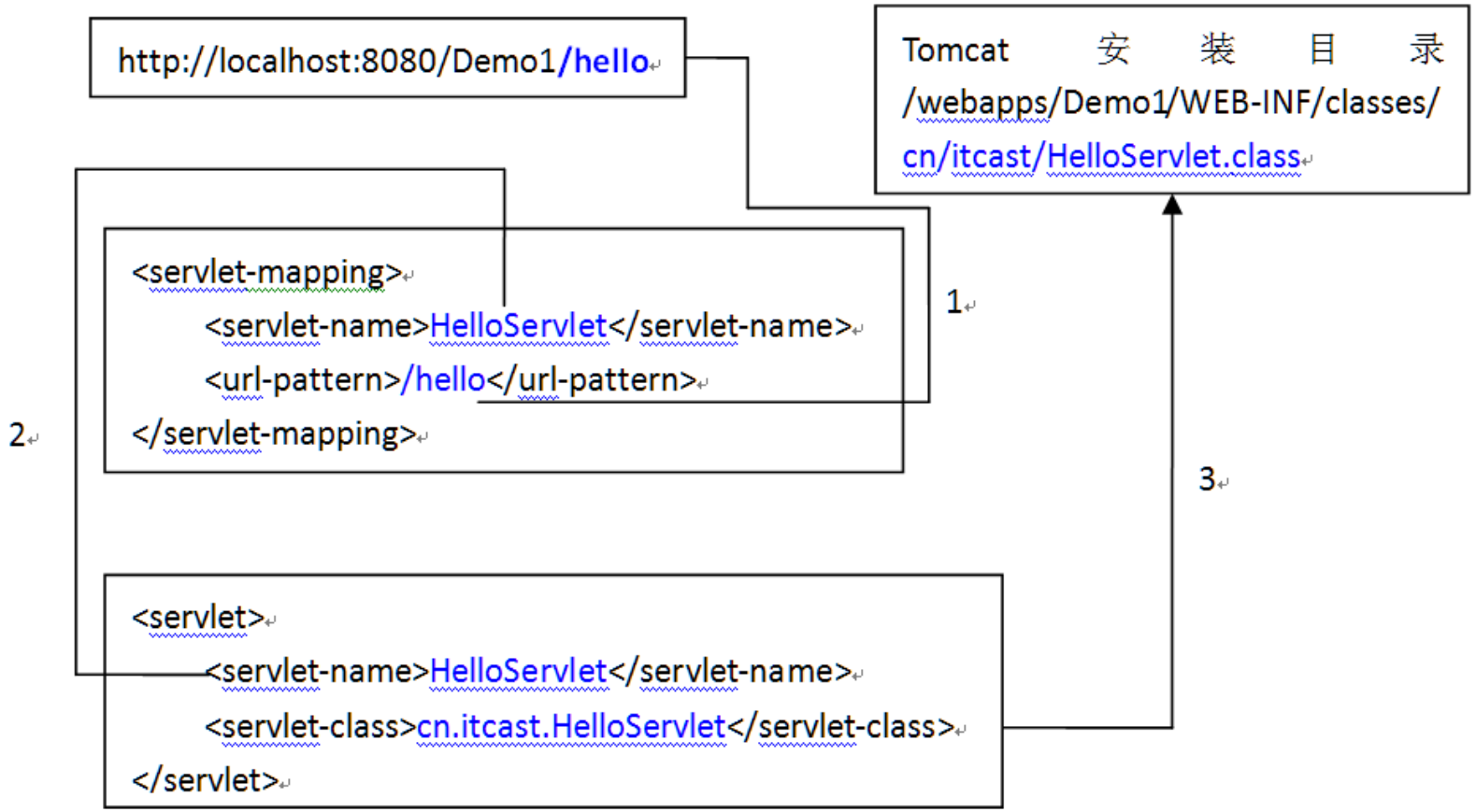
运行常见错误，见备注

执行过程

1. 客户端发出请求http://localhost:8080/Demo1/abc
2. 根据web.xml文件的配置，找到<url-pattern>子元素的值“/hello”的<servlet-mapping>元素

- 读取<servlet-mapping>元素的<servlet-name>子元素的值，由此确定Servlet的名字为”HelloServlet”
- 找到<servlet-name>值为HelloServlet的<servlet>元素
- 读取<servlet>元素的<servlet-class>子元素的值，由此确定Servlet的类名为cn.itcast.HelloServlet。
- 到Tomcat安装目录/webapps/Demo1/WEB-INF/classes/cn/itcast目录下查找到HelloServlet.class文件

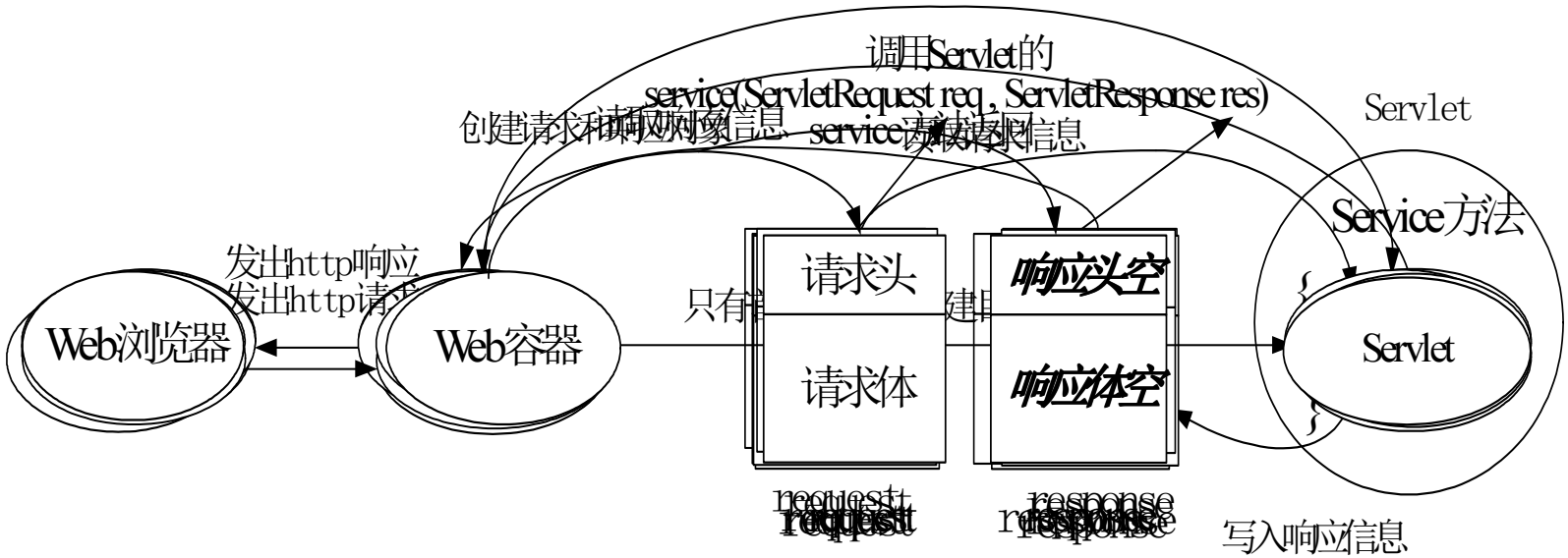
执行过程图解


http://localhost:8080/Demo1/hello
Tomcat 安 装 目 录
/webapps/Demo1/WEB-INF/classes/
cn/itcast/HelloServlet.class
1
<servlet-mapping>
<servlet-name>HelloServlet</servlet-name>
<url-pattern>/hello</url-pattern>
</servlet-mapping>
2
3
<servlet>
<servlet-name>HelloServlet</servlet-name>
<servlet-class>cn.itcast.HelloServlet</servlet-class>
</servlet>

Tip： Servlet的运行过程

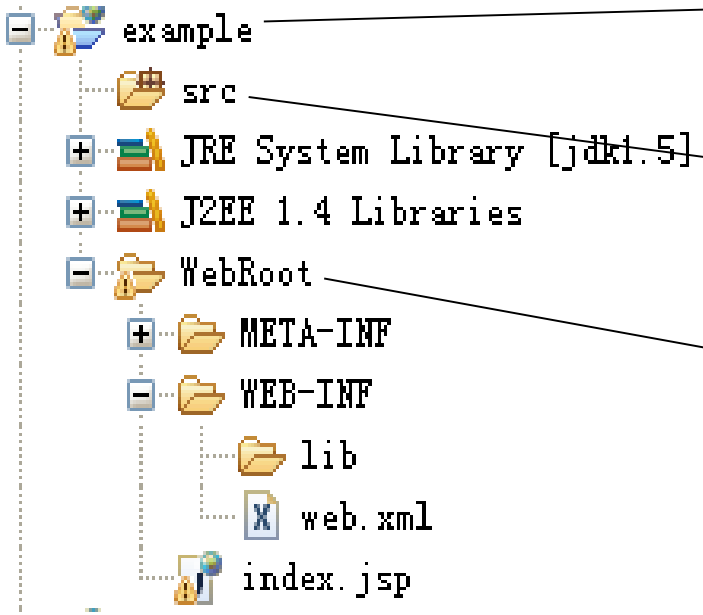
- Servlet程序是由WEB服务器调用，web服务器收到客户端的Servlet访问请求后：
  - ① Web服务器首先检查是否已经装载并创建了该Servlet的实例对象。如果是，则直接执行第④步，否则，执行第②步。
  - ② 装载并创建该Servlet的一个实例对象。
  - ③ 调用Servlet实例对象的init()方法。
  - ❿ 创建一个用于封装HTTP请求消息的HttpServletRequest对象和一个代表HTTP响应消息的HttpServletResponse对象，然后调用Servlet的service()方法并将请求和响应对象作为参数传递进去。
  - ❿ WEB应用程序被停止或重新启动之前，Servlet引擎将卸载Servlet，并在卸载之前调用Servlet的destroy()方法。

Tip： Servlet的运行过程



在Eclipse中开发Servlet

- 在eclipse中新建一个web project工程，eclipse会自动创建下图所示目录结构：



web工程的名称，该工程部署时，在webapps目录下就会有一个example的web应用

Java程序的开发目录，该目录下编写的所有j程序在部署时，会自动部署到example/web-inf/classes目录下。

webroot对应于web应用的根目录，该目录下的所有子目录和子文件在部署时，会原封不动的发布到web应用目录下。

Servlet的生命周期

Servlet的生命周期(1)

- Servlet是一个供其他Java程序（Servlet引擎）调用的Java类，它不能独立运行，它的运行完全由Servlet引擎来控制和调度。

- 针对客户端的多次Servlet请求，通常情况下，服务器只会创建一个Servlet实例对象，也就是说Servlet实例对象一旦创建，它就会驻留在内存中，为后续的其它请求服务，直至web容器退出，servlet实例对象才会销毁。

- 在Servlet的整个生命周期内，Servlet的init方法只被调用一次。而对一个Servlet的每次访问请求都导致Servlet引擎调用一次servlet的service方法。对于每次访问请求，Servlet引擎都会创建一个新的HttpServletRequest请求对象和一个新的HttpServletResponse响应对象，然后将这两个对象作为参数传递给它调用的Servlet的service()方法，service方法再根据请求方式分别调用doXXX方法。

Servlet的生命周期(2)

- 如果在<servlet>元素中配置了一个<load-on-startup>元素，那么WEB应用程序在启动时，就会装载并创建Servlet的实例对象、以及调用Servlet实例对象的init()方法。

举例：

```
<servlet>
        <servlet-name>invoker</servlet-name>
        <servlet-class>
                org.apache.catalina.servlets.InvokerServlet
        </servlet-class>
        <load-on-startup>2</load-on-startup>
</servlet>
```

- 用途：为web应用写一个InitServlet，这个servlet配置为启动时装载，为整个web应用创建必要的数据库表和数据。

Servlet接口实现类

- Servlet接口SUN公司定义了两个默认实现类，分别为：GenericServlet、HttpServlet。
- HttpServlet指能够处理HTTP请求的servlet，它在原有Servlet接口上添加了一些与HTTP协议处理方法，它比Servlet接口的功能更为强大。因此开发人员在编写Servlet时，通常应继承这个类，而避免直接去实现Servlet接口。
- HttpServlet在实现Servlet接口时，覆写了**service方法**，该方法体内的代码会自动判断用户的请求方式，如为GET请求，则调用HttpServlet的doGet方法，如为Post请求，则调用doPost方法。因此，开发人员在编写Servlet时，通常只需要覆写doGet或doPost方法，而不要去覆写service方法。
- 阅读HttpServlet API文档，看一下servlet-api.jar

Servlet的一些细节(1)

- 由于客户端是通过URL地址访问web服务器中的资源，所以Servlet程序若想被外界访问，必须把servlet程序映射到一个URL地址上，这个工作在web.xml文件中使用<servlet>元素和<servlet-mapping>元素完成。
- <servlet>元素用于注册Servlet，它包含有两个主要的子元素：<servlet-name>和<servlet-class>，分别用于设置Servlet的注册名称和Servlet的完整类名。
- 一个<servlet-mapping>元素用于映射一个已注册的Servlet的一个对外访问路径，它包含有两个子元素：<servlet-name>和<url-pattern>，分别用于指定Servlet的注册名称和Servlet的对外访问路径。例如：

```
<web-app>
    <servlet>
        <servlet-name>AnyName</servlet-name>
        <servlet-class>HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>AnyName</servlet-name>
        <url-pattern>/demo/hello.html</url-pattern>
    </servlet-mapping>
</web-app>
```

Servlet的一些细节(2)

同一个Servlet可以被映射到多个URL上，即多个<servlet-mapping>元素的<servlet-name>子元素的设置值可以是同一个Servlet的注册名。

在Servlet映射到的URL中也可以使用*通配符，但是只能有两种固定的格式：一种格式是“*.扩展名”，另一种格式是以正斜杠（/）开头并以“/*”结尾。

```
<servlet-mapping>
    <servlet-name>
        AnyName
    </servlet-name>
    <url-pattern>
        *.do
    </url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
    <servlet-name>
        AnyName
    </servlet-name>
    <url-pattern>
        /action/*
    </url-pattern>
</servlet-mapping>
```

Servlet的一些细节(3)

对于如下的一些映射关系：

- Servlet1 映射到 /abc/*
- Servlet2 映射到 /*
- Servlet3 映射到 /abc
- Servlet4 映射到 *.do

问题：

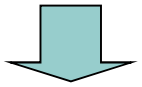
- 当请求URL为“/abc/a.html”，“/abc/*”和“/*”都匹配，哪个servlet响应
  Servlet引擎将调用Servlet1。
- 当请求URL为“/abc”时，“/abc/*”和“/abc”都匹配，哪个servlet响应
  Servlet引擎将调用Servlet3。
- 当请求URL为“/abc/a.do”时，“/abc/*”和“*.do”都匹配，哪个servlet响应
  Servlet引擎将调用Servlet1。
- 当请求URL为“/a.do”时，“/*”和“*.do”都匹配，哪个servlet响应
  Servlet引擎将调用Servlet2。
- 当请求URL为“/xxx/yyy/a.do”时，“/*”和“*.do”都匹配，哪个servlet响应
  Servlet引擎将调用Servlet2。

Servlet的一些细节(4)

- 如果某个Servlet的映射路径仅仅为一个正斜杠（/），那么这个Servlet就成为当前Web应用程序的缺省Servlet。
- 凡是在web.xml文件中找不到匹配的<servlet-mapping>元素的URL，它们的访问请求都将交给缺省Servlet处理，也就是说，缺省Servlet用于处理所有其他Servlet都不处理的访问请求。
- 在<tomcat的安装目录>\conf\web.xml文件中，注册了一个名称为org.apache.catalina.servlets.DefaultServlet的Servlet，并将这个Servlet设置为了缺省Servlet。
- 当访问Tomcat服务器中的某个静态HTML文件和图片时，实际上是在访问这个缺省Servlet。

线程安全

Servlet的线程安全

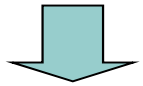


- 当多个客户端并发访问同一个Servlet时，web服务器会为每一个客户端的访问请求创建一个线程，并在这个线程上调用Servlet的service方法，因此service方法内如果访问了同一个资源的话，就有可能引发线程安全问题。
- 如果某个Servlet实现了SingleThreadModel接口，那么Servlet引擎将以单线程模式来调用其service方法。
- SingleThreadModel接口中没有定义任何方法，只要在Servlet类的定义中增加实现SingleThreadModel接口的声明即可。
- 对于实现了SingleThreadModel接口的Servlet，Servlet引擎仍然支持对该Servlet的多线程并发访问，其采用的方式是产生多个Servlet实例对象，并发的每个线程分别调用一个独立的Servlet实例对象。
- 实现SingleThreadModel接口并不能真正解决Servlet的线程安全问题，因为Servlet引擎会创建多个Servlet实例对象，而**真正意义上解决多线程安全问题是指一个Servlet实例对象被多个线程同时调用的问题**。事实上，在Servlet API 2.4中，已经将SingleThreadModel标记为Deprecated（过时的）。

解决并发问题

- 解决并发出现的问题，可以采用以下方式：
- 使用Java同步机制对多线程同步:运行效率低
- 使用SingleThreadModel接口
- 合理决定在Servlet中定义的变量的作用域

ServletConfig对象



- 在Servlet的配置文件中，可以使用一个或多个<init-param>标签为servlet配置一些初始化参数。
- 当servlet配置了初始化参数后，web容器在创建servlet实例对象时，会自动将这些初始化参数封装到ServletConfig对象中，并在调用servlet的init方法时，将ServletConfig对象传递给servlet。进而，程序员通过ServletConfig对象就可以得到当前servlet的初始化参数信息。
- 阅读ServletConfig API，并举例说明该对象的作用：
  - 获得字符集编码
  - 获得数据库连接信息

ServletContext

- WEB容器在启动时，它会为每个WEB应用程序都创建一个对应的ServletContext对象，它代表当前web应用。
- ServletConfig对象中维护了ServletContext对象的引用，开发人员在编写servlet时，可以通过ServletConfig.getServletContext方法获得ServletContext对象。
- 由于一个WEB应用中的所有Servlet共享同一个ServletContext对象，因此Servlet对象之间可以通过ServletContext对象来实现通讯。ServletContext对象通常也被称之为**context域对象**。
- 查看ServletContext API文档，了解ServletContext对象的功能。

ServletContext应用

- 多个Servlet通过ServletContext对象实现数据共享。
- 获取WEB应用的初始化参数。
- 实现Servlet的转发。
- 利用ServletContext对象读取资源文件。
  - .properties文件（属性文件）
  - 得到文件路径(ServletContext.getRealPath(),ServletContext.getReaourceAsStream())
  - 思考：如果一个普通类该如何读取配置文件？？
  - 读取资源文件的三种方式
- 练习：实现文件下载