Web应用开发
之
Servlet技术模型

赵小敏
浙江工业大学计算机科学与技术学院

本节内容

- Web应用程序及结构
- 部署描述文件
- 注解
- ServletConfig和ServletContext接口

2.5 Web应用程序及结构

- 2.5.1 Web应用程序
- 2.5.2 应用服务器
- 2.5.3 Web应用程序的结构

2.5.1 Web应用程序

- **Web应用程序**是一种可以通过Web访问的应用程序。
- 一个Web应用程序是由完成特定任务的各种Web组件（Web Components）构成的并通过Web将服务展示给外界。

2.5.2 应用服务器

- Web应用程序驻留在应用服务器（Application Server）上。
- 应用服务器为 Web 应用程序提供一种简单的和可管理的对系统资源的访问机制。它也提供低级的服务，如HTTP协议的实现和数据库连接管理。
- Servlet 容器仅仅是应用服务器的一部分。

2.5.2　应用服务器

- 市场上可以得到多种应用服务器，其中包括
- Apache 的Tomcat
- JBoss
- Oracle的WebLogic
- IBM 的WebSphere
- 国产应用服务器：东方通TongWeb、中创软件InforWeb、金蝶Apusic

2.5.3 Web应用程序的结构

- Web应用程序具有严格定义的目录结构。
- 一个Web应用程序的所有资源被保存在一个结构化的目录中，目录结构是按照资源和文件的位置严格定义的。
- Tomcat安装目录的webapps目录是所有Web应用程序的默认根目录。

1. 理解文档根目录

- 每个Web应用程序都有一个文档根目录（document root），它是应用程序所在的目录。
- 如果要访问html目录中的/hello.html 文件，应该使用下面的URL。
- http://localhost:8080/web2020/html/hello.html

web2020
- ├ css (存放样式文件)
- ├ html (存放HTML文件)
- ├ images (存放GIF、JPEG或PNG文件)
- ├ js (存放JavaScript脚本文件)
- ├ jsp (存放JSP文件)
- ├ index.html (默认的欢迎文件)
- ├WEB-INF
- ├ classes（类文件目录）
  - ├ com.demo.LoginServlet.class
  - ├ lib（库文件目录）
    - ├ *.jar(fastjson-1.2.9.jar,mytaglib.jar)
  - ├ web.xml（部署描述文件）

2. 理解WEB-INF目录

- 每个Web应用程序在它的根目录中都有一个WEB-INF目录。
- 该目录中主要存放供服务器访问的资源。
- 该目录主要包含三个内容。

  1）classes目录

  2）lib目录

  3）web.xml文件

3. Web归档文件

- 一个Web应用程序包含许多文件，可以将这些文件打包成一个扩展名为.war的Web归档文件中，一般称为**WAR文件**。
- 可以直接把一个WAR文件放到Tomcat的webapps目录中，Tomcat会自动把该文件的内容释放到webapps目录中并创建一个与WAR文件同名的应用程序。

4. 默认的Web应用程序

- 除用户创建的Web应用程序外，Tomcat服务器还维护一个默认的Web应用程序。*<tomcat-install>***webapps\ROOT**目录被设置为默认的Web应用程序的文档根目录。
- 它与其他的Web应用程序类似，只不过访问它的资源不需要指定应用程序的名称或上下文路径。

5、web应用程序的部署

- 导出war文件
- 配置tomcat
- 启动tomcat

2.6 部署描述文件

- Web应用程序中包含多种组件，有些组件可使用注解配置，有些组件需使用部署描述文件配置。
- 部署描述文件（Deployment Descriptor，简称**DD**）可用来初始化Web应用程序的组件,每个web应用程序都有一个web.xml文件

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>chapter02</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
    <servlet>
        <servlet-name>HelloServlet2</servlet-name>
        <servlet-class>HelloServlet2</servlet-class>
   </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet2</servlet-name>
        <url-pattern>/hello.do</url-pattern>
    </servlet-mapping>
</web-app>
```

2.6.1 DD文件的定义

- 为了保证跨Web容器的可移植性，部署描述文件的文档类型定义（Document Type Definition，**DTD**）规定了XML文档的语法和标签的规则，这些规则包括一系列的元素和实体的声明。
- 这些元素和实体声明都包含在<web-app>元素中

在<web-app>中定义的元素

元素名	说明
description	对应用程序的简短描述
display-name	定义应用程序的显示名称
context-param	定义应用程序的初始化参数
servlet	定义Servlet
servlet-mapping	定义Servlet映射
welcome-file-list	定义应用程序的欢迎文件
session-config	定义会话时间
listener	定义监听器类
filter	定义过滤器
filter-mapping	定义过滤器映射
error-page	定义错误处理页面
security-constraint	定义Web应用程序的安全约束
mime-mapping	定义常用文件扩展名的MIME类型

2.6.2 <servlet >元素

- <servlet>元素为Web应用程序定义一个Servlet，该元素的DTD定义如下。

```
<!ELEMENT servlet (description?, icon?,
  display-name?, servlet-name,
  (servlet-class | jsp-file), init-param*,
load-on-startup?, security-role-ref*)>
```

1. <servlet-name>元素

- 该元素用来定义Servlet名称，该元素是必选项。定义的名称在DD文件中应该唯一。
- 可以通过ServletConfig的getServletName()方法检索Servlet名。

2. <servlet-class>元素

- 该元素指定Servlet类的完整名称，即需要带包的名称，例如com.demo.HelloServlet。
- 容器将使用该类创建Servlet实例。
- Servlet类以及它所依赖的所有类都应该在Web应用程序的类路径中。
- WEB-INF目录中的classes目录和lib目录中的jar文件被自动添加到容器的类路径中，因此如果把类放到这两个地方就不需要设置类路径。

3. <init-param>元素

- 该元素定义向Servlet传递的初始化参数。
- 在一个<servlet>元素中可以定义任意多个<init-param>元素。每个<init-param>元素必须有且仅有一组<param-name>和<param-value>子元素。
- Servlet可以通过ServletConfig接口的getInitParameter( )方法检索初始化参数。

4. <load-on-startup>元素

- <load-on-startup>元素指定是否在Web应用程序启动时载入该Servlet。
- 该元素的值是一个整数。如果没有指定该元素或其内容为一个负数，容器将根据需要决定何时装入Servlet。如果其内容为一个正数，则在Web应用程序启动时载入该Servlet。
- 对不同的Servlet，可以指定不同的值，这可以控制容器装入这些Servlet的顺序，值小的先装入。

2.6.3 <servlet-mapping >元素

- <servlet-mapping>元素定义一个映射，它指定哪个URL模式被该Servlet处理。
- 容器使用这些映射根据实际的URL访问合适的Servlet。
- <servlet-mapping>元素的DTD定义：

<!ELEMENT servlet-mapping (servlet-name, url-pattern)>

2.6.4 <welcome-file-list>元素

- 通常在浏览器的地址栏中输入一个路径名称，而没有指定特定的文件，也能访问到一个页面，这个页面就是欢迎页面，文件名通常为index.html或index.jsp。
- 在Tomcat中，如果访问的URL是目录，并且没有特定的Servlet与这个URL模式匹配，那么它将在该目录中首先查找index.html文件，如果找不到将查找index.jsp文件，如果找到上述文件，将该文件返回给客户。如果找不到（包括目录也找不到），将向客户发送404错误信息。

2.7 @WebServlet和@WebInitParam注解

- 在Servlet 3.0中可以使用@WebServlet注解而不需要在web.xml文件中定义Servlet。
- 该注解属于javax.servlet.annotation包，在定义Servlet时应使用下列语句导入：

import javax.servlet.annotation.WebServlet;

```
@WebServlet(name = "genericServlet",urlPatterns={"/generic-servlet" })
@WebServlet("/generic-servlet")
```

@WebServlet注解的常用元素

元素名	类 型	说 明
name	String	指定Servlet名称，等价于web.xml中的<servlet-name>元素。如果没有显式指定，则使用Servlet的完全限定名作为名称
urlPatterns	String[]	指定一组Servlet的URL映射模式，该元素等价于web.xml文件中的<url-pattern>元素
value	String[]	该元素等价于urlPatterns元素。两个元素不能同时使用
loadOnStartup	int	指定该Servlet的加载顺序，等价于web.xml文件中的<load-on-startup>元素
initParams	WebInitParam[]	指定Servlet的一组初始化参数，等价于<init-param>元素
asyncSupported	boolean	声明Servlet是否支持异步操作模式，等价于web.xml文件中的<async-supported>元素
description	String	指定该Servlet的描述信息，等价于<description>元素
dispalyName	String	指定该Servlet的显示名称，等价于<display-name>元素

@WebInitParam注解的常用元素

元素名	类 型	说 明
name	String	指定初始化参数名，等价于<param-name>元素
value	String	指定初始化参数值，等价于<param-value>元素
description	String	关于初始化参数的描述，等价于<description>元素

```
@WebServlet(name="ConfigDemoServlet",urlPatterns ={"/config-demo" },
    initParams = {
        @WebInitParam(name = "email", value = "hacker@163.com"),
        @WebInitParam(name = "telephone", value = "8899123") })
```

2.8 ServletConfig和ServletContext接口

1、 ServletConfig接口

- 在Servlet初始化时，容器将调用init(ServletConfig)方法，并为其传递一个ServletConfig对象，该对象称为Servlet配置对象，使用该对象可以获得Servlet初始化参数、Servlet名称、ServletContext对象等。
- 要得到ServletConfig接口对象有两种方法：

(1)覆盖Servlet的init(ServletConfig config)方法，然后把容器创建的ServletConfig对象保存到一个成员变量中

(2)在Servlet中直接使用getServletConfig()方法获得ServletConfig对象

ServletConfig接口示例

```
@WebServlet(name = "ConfigDemoServlet", urlPatterns = { "/config-demo" },
   initParams = {
      @WebInitParam(name = "email", value = "hacker@163.com"),
      @WebInitParam(name = "telephone", value = "8899123") })
public class ConfigDemoServlet extends HttpServlet {
    String servletName = null;
    ServletConfig config = null;
    String email = null;
    String telephone = null;
    public void init(ServletConfig config) {
        this.config = config;
        servletName = config.getServletName();
        email = config.getInitParameter("email");
        telephone = config.getInitParameter("telephone");
    }

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

2、ServletContext接口

- Web容器在启动时会加载每个Web应用程序，并为每个Web应用程序创建一个唯一的ServletContext实例对象，该对象一般称为Servlet上下文对象。
- Servlet可以用javax.servlet.ServletContext对象来获得Web应用程序的初始化参数或Servlet容器的版本等信息，它也可以被Servlet用来与其他的Servlet共享数据

2、ServletContext接口

- 有两种方法得到ServletContext引用：

（1）直接调用getServletContext()方法

```
ServletContext context = getServletContext();
```

（2）先得到ServletConfig引用，再调用它的getServletContext()方法

```
ServletContext context = getServletConfig().getServletContext();
```

获取应用程序的初始化参数

- ServletContext对象是在Web应用程序装载时初始化的。可以使用下面两个方法检索Servlet上下文初始化参数：
- public String getInitParameter(String name)：返回指定参数名的字符串参数值，如果参数不存在则返回null。
- public Enumeration getInitParameterNames()：返回一个包含所有初始化参数名的Enumeration对象。

获取应用程序的初始化参数

- 应用程序初始化参数应该在web.xml文件中使用<context-param>元素定义，而不能通过注解定义。

```
<context-param>
    <param-name>adminEmail</param-name>
    <param-value>webmaster@163.com</param-value>
</context-param>
```

- 在Servlet中可以使用下面代码检索adminEmail参数值。

```
ServletContext context = getServletContext();
String email = context.getInitParameter("adminEmail");
```

通过ServletContext对象获得资源

- public URL getResource(String path)：返回由给定路径指定的资源的URL对象。
- public InputStream getResourceAsStream(String path)：如果想从资源上获得一个InputStream对象，这是一个简洁的方法，它等价于getResource(path).openStream()。
- public String getRealPath(String path)：返回给定的相对路径的绝对路径。
- 示例：FileDownloadServlet.java

登录日志

- 使用ServletContext接口的log()方法可以将指定的消息写到服务器的日志文件中，该方法有下面两种格式。
- public void log(String msg)：参数msg为写到日志文件中的消息。
- public void log(String msg, Throwable throwable)：将msg指定的消息和异常的栈跟踪信息写入日志文件。

用RequestDispatcher实现请求转发

- 使用ServletContext接口的下列两个方法也可以获得RequestDispatcher对象，实现请求转发。
- RequestDispatcher getRequestDispatcher(String path)：参数path表示资源路径，它必须以“/”开头，表示相对于Web应用的文档根目录。
- RequestDispatcher getNamedDispatcher(String name)：参数name为一个命名的Servlet对象。Servlet和JSP页面都可以通过Web应用程序的DD文件指定名称。

使用ServletContext对象存储数据

- 使用ServletContext对象也可以存储数据，该对象也是一个**作用域对象**，它的作用域是整个应用程序。在ServletContext接口中也定义了4个处理属性的方法.
- public void setAttribute(String name, Object object)：将给定名称的属性值对象绑定到上下文对象上。

使用ServletContext对象存储数据

- public Object getAttribute(String name)：返回绑定到上下文对象上的给定名称的属性值，如果没有该属性，则返回null。
- public Enumeration getAttributeNames()：返回绑定到上下文对象上的所有属性名的Enumeration对象。
- public void removeAttribute(String name)：从上下文对象中删除指定名称的属性。

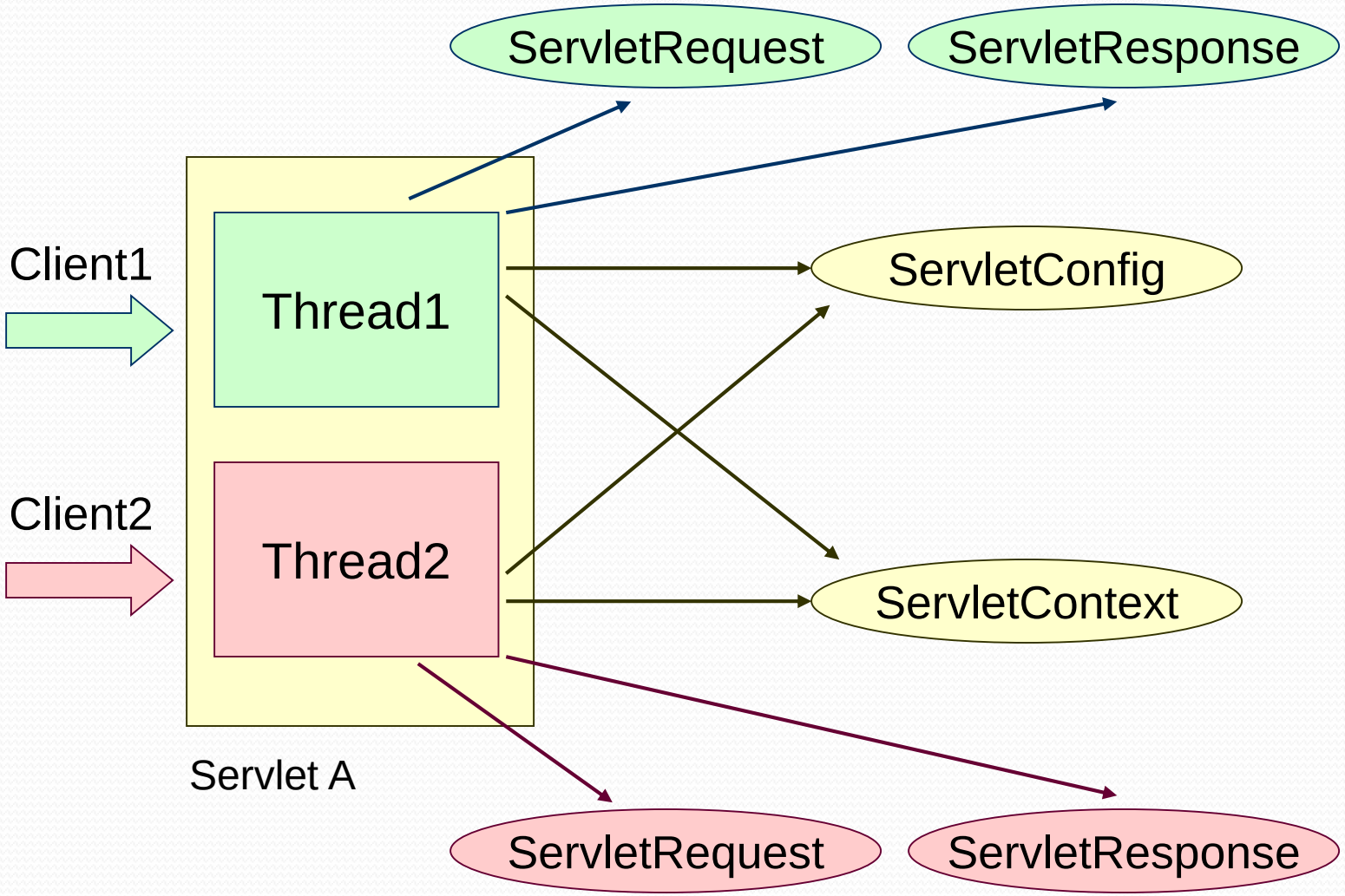
检索Servlet容器的信息

- getServerInfo()方法返回Servlet所运行的容器的名称和版本。
- getMajorVersion()和getMinorVersion()方法可以返回容器所支持的Servlet API的主版本号和次版本号。
- getServletContextName()方法返回与该ServletContext对应的Web应用程序名称，它是在web.xml中使用<display-name>元素定义的名称。

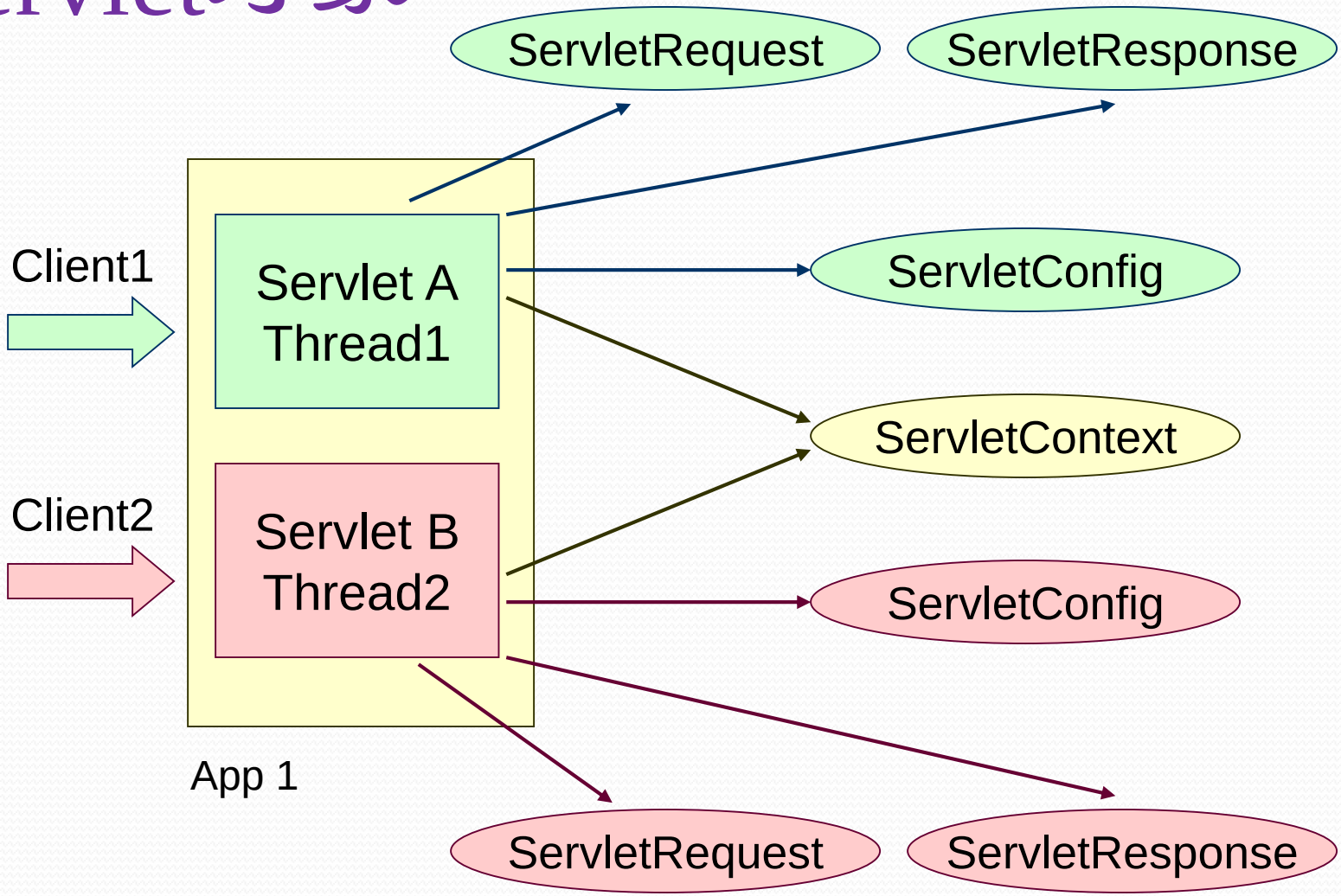
ServletConfig和ServletContext对象区别

- ServletConfig：获取当前Servlet信息，也可获取初始化信息，每产生一个Servlet对象会产生一个ServletConfig对象
- ServletContext：获取所有Servlet的Web应用设置

Servlet对象


ServletRequest
ServletResponse
Client1
Thread1
ServletConfig
Client2
Thread2
ServletContext
Servlet A
ServletRequest
ServletResponse

Servlet对象


ServletRequest
ServletResponse
Client1
Servlet A
Thread1
ServletConfig
ServletContext
Client2
Servlet B
Thread2
ServletConfig
App 1
ServletRequest
ServletResponse

Thank You !