



Servlet技术

Allen Long

Email: allen@huihoo.com

<http://www.huihoo.com>

2004-04



JAVA

内容安排



- HTTP协议
- Servlet生命周期
- Servlet开发
- Servlet部署
- 会话管理
- Servlet高级话题



HTTP协议概述



- 请求
- 相应
- 头部信息
- GET和POST



Web应用的发展历史



- CGI
- FastCGI
- 扩展API (NSAPI ISAPI)
- ASP
- Servlet
- JSP



Servlet优势



- 比CGI脚本快，因为Servlet采用了不同的处理模式。
- Servlet使用的标准API为大多数Web Server所接受。
- 因为是Java语言开发的，所以拥有Java的所有优点，包括易于开发和平台无关等。
- 可以方便地访问大量的Java类库资源。



Servlet与Servlet引擎



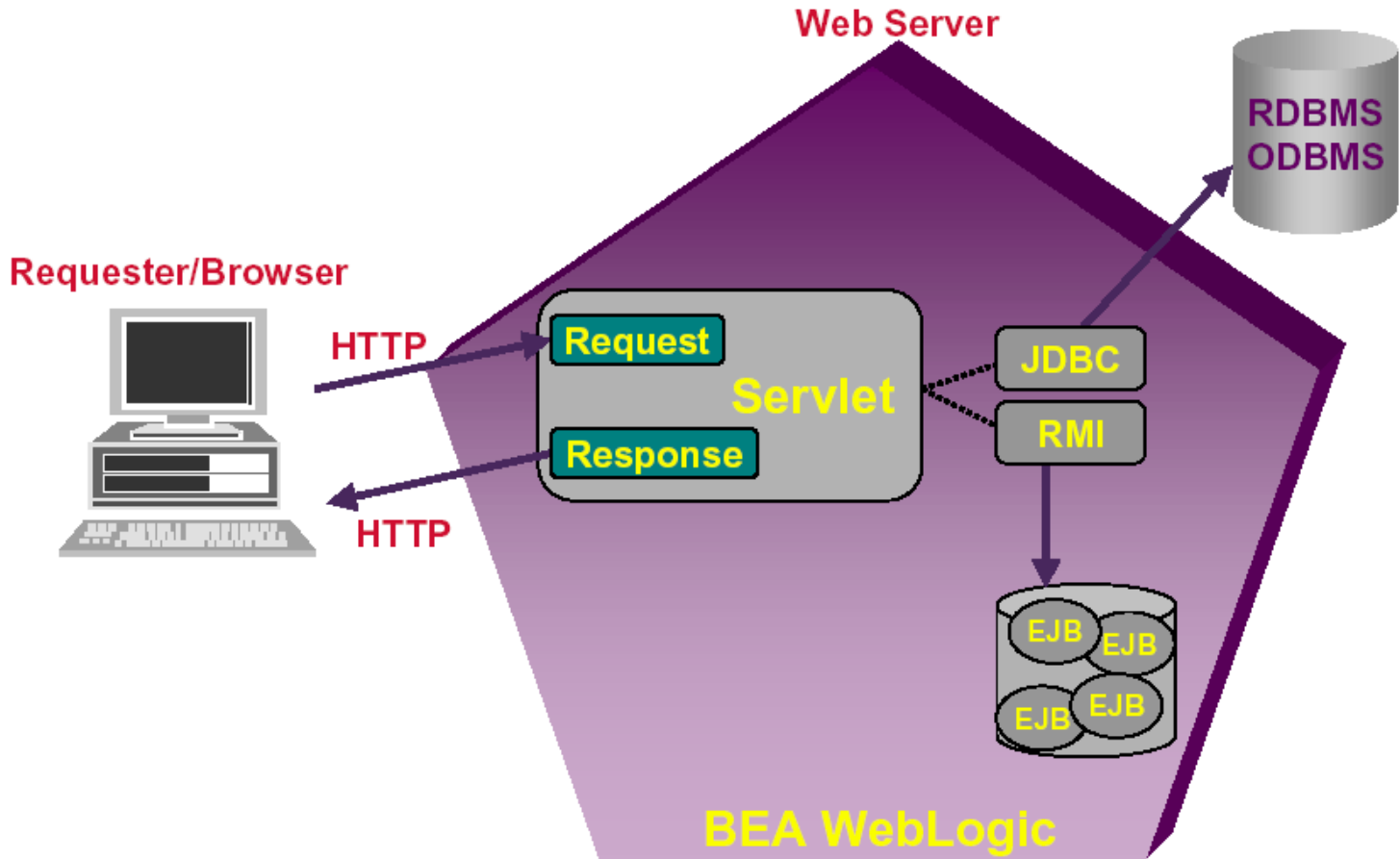
- Servlet Engine提供network Service, 响应MIME request, 运行Servlet Container。



- Servlet Engine 负责实例化和加载Servlet, 这个过程可以在Servlet Engine 加载时执行, 可以在Servlet 响应请求时执行, 也可以在两者之间的任何时候执行。



Servlet Access Access Model



JAVA

Sample Servlet



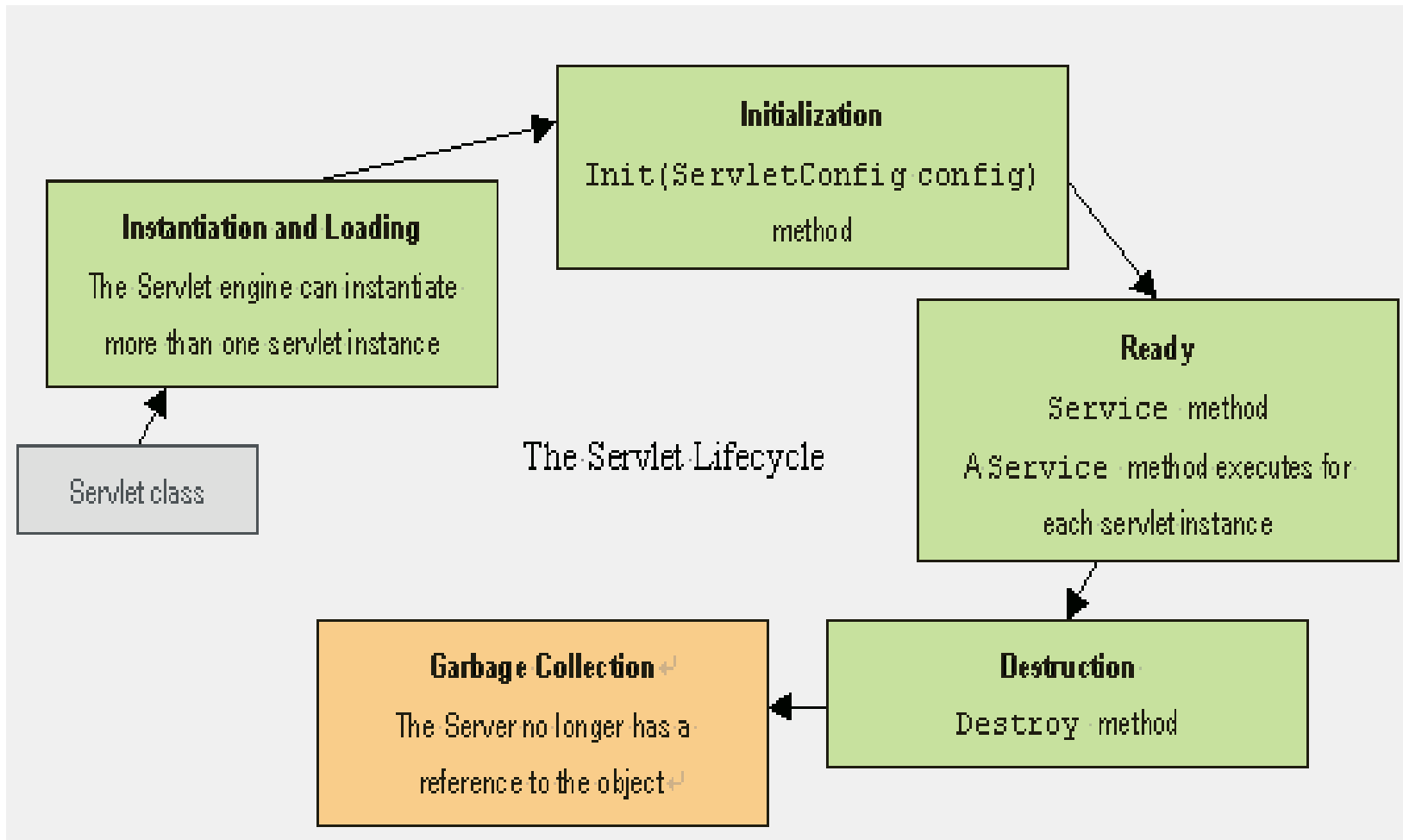
```
public class SimpleServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        // 首先设置头部
        res.setContentType("text/html");

        // 用 writer方法返回响应数据
        PrintWriter out = res.getWriter();
        out.println("<HEAD><TITLE> SimpleServlet</TITLE></HEAD><BODY>");
        out.println("<h1> SimpleServlet Output </h1>");
        out.println("<P>This is output is from SimpleServlet.");
        out.println("</BODY>");
        out.close();
    }
}
```



JAVA

Servlet 生命周期



初始化



Servlet Engine 加载好Servlet 后，必须要初始化它。在初始化阶段，`init()`方法被调用。这个方法在`javax.servlet.Servlet`接口中定义。`init()`方法以一个Servlet 配置文件（`ServletConfig` 型）为参数。Servlet configuration 对象由Servlet Engine 实现，可以让Servlet 从中读取一些name-value对的参数值。`ServletConfig` 对象还可以让Servlet access 一个Servlet Context对象。



Service



Servlet 被初始化以后，就处于能响应请求的就绪状态。每个对Servlet 的请求由一个Servlet Request 对象代表。Servlet 给客户端的响应由一个Servlet Response对象代表。当客户端有一个请求时，Servlet Engine 将ServletRequest 和ServletResponse对象都转发给Servlet，这两个对象以参数的形式传给Service方法。这个方法由javax.servlet.Servlet定义、并由具体的Servlet 实现。

ServletRequest接口可以让Servlet 获取客户端请求中的参数。如form data, request信息, 协议类型等等。Servlet 可以从ServletInputStream流中读取request 数据。ServletResponse接口允许Servlet设置response headers和status codes 。



Destroy



当服务器卸载一个servlet，它将调用servlet的destroy方法。这个destroy方法是与初始化方法相反，同时从内存中释放servlet。





Servlet开发



JAVA

Package javax.servlet.http



Class Hierarchy

oclass java.lang.[Object](#)

oclass javax.servlet.http.[Cookie](#) (implements java.lang.[Cloneable](#))

oclass java.util.[EventObject](#) (implements java.io.[Serializable](#))

oclass javax.servlet.http.[HttpSessionEvent](#)

oclass javax.servlet.http.[HttpSessionBindingEvent](#)

oclass javax.servlet.[GenericServlet](#) (implements java.io.[Serializable](#), javax.servlet.[Servlet](#), javax.servlet.[ServletConfig](#))

oclass javax.servlet.http.[HttpServlet](#) (implements java.io.[Serializable](#))

oclass javax.servlet.http.[HttpUtils](#)

oclass javax.servlet.[ServletRequestWrapper](#) (implements javax.servlet.[ServletRequest](#))

oclass javax.servlet.http.[HttpServletRequestWrapper](#) (implements javax.servlet.http.[HttpServletRequest](#))

oclass javax.servlet.[ServletResponseWrapper](#) (implements javax.servlet.[ServletResponse](#))

oclass javax.servlet.http.[HttpServletRequestWrapper](#) (implements javax.servlet.http.[HttpServletRequest](#))

Interface Hierarchy

ointerface java.util.[EventListener](#)

ointerface javax.servlet.http.[HttpSessionActivationListener](#)

ointerface javax.servlet.http.[HttpSessionAttributeListener](#)

ointerface javax.servlet.http.[HttpSessionBindingListener](#)

ointerface javax.servlet.http.[HttpSessionListener](#)

ointerface javax.servlet.http.[HttpSession](#)

ointerface javax.servlet.http.[HttpSessionContext](#)

ointerface javax.servlet.[ServletRequest](#)

ointerface javax.servlet.http.[HttpServletRequest](#)

ointerface javax.servlet.[ServletResponse](#)

ointerface javax.servlet.http.[HttpServletRequest](#)



HTTP Servlet Sample



- Extend HttpServlet
- The Service method is call by the servlet container for each request





Sample



HttpServletRequest



- 获得CGI变量
- `getParameterNames()` 在HTML页上,返回一个的枚举的参数
- `getParameterValues(String name)` 返回multi-valued参数的值
- `getParameter(String name)` 返回一个指定的命名的参数的值



HttpServletResponse



- 向客户提供通讯的通道
- 允许servlet返回内容并且/或者是错误
- 设置内容头（类型，长度.....）
- 重定向服务器以返回一个指的URL



Handling Requests Handling Requests



- Servlets will retrieve data from:
 - initialization parameters (ServletConfig interface)
 - servlet parameters (ServletRequest interface) that come from
 - from query string OR
 - TML forms
 - **cookies**



ServletConfig接口



- The ServletConfig interface describes the configuration of a single servlet.

The ServletConfig Interface:

```
Public interface ServletConfig {  
    public ServletContext getServletContext();  
    public String getInitParameter(String name);  
    public Enumeration getInitParameterNames();  
}
```





An Example



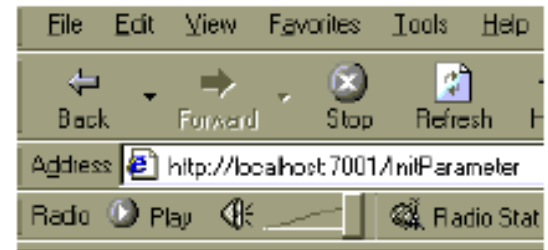
- Let's write a servlet that:
 - prints out a statement NUM_PRINT times
 - pulls NUM_PRINT from an initialization parameter



JAVA

```
<servlet>
...
  <init-param>
    <param-name>NUM_PRINT</param-name>
    <param-value>7</param-value>
  </init-param>
</servlet>
```

 web.xml



This is a statement
This is a statement
This is a statement
This is a statement
This is a statement
This is a statement
This is a statement

URLs



- A URL pointing to a Servlet can be split into portions:
 - protocol, server, port
 - Web Application name
 - extra path information(Servlet jsp html)
 - the query string (containing parameters)



`http://localhost:7001/AServlet/extra/info?val1=cool&val2=not%20cool`



JAVA

Query Strings



- **Use a query string to pass parameters from the browser to the web server.**

Query String Rules:

- 1、 The query string starts with a question mark ‘?’
- 2、 Use name = value for each name/value pair
- 3 、 Separate each name/value pair with an ‘&’
- 4、 Use %20 for each space
- 5、 Use %33 for each percent sign





Sample



HTML Forms



- You can use HTML forms (instead of query strings) to send parameters to a servlet
- One parameter is passed for each form field

HTML Form Syntax:

```
<FORM method = [GET|POST] action = [URLofServletToExecute]>  
    <INPUT type = [text|textarea|select|hidden|many others]  
        name = [FieldName]  
        value = [FieldValue]>
```

...

```
<INPUT type = SUBMIT>
```

```
</FORM>
```



Sample



Last:

First:

Phone:

Email:

HTML Form Syntax:

```
<FORM ACTION="http://localhost:7001/phoneBook" METHOD="POST">
Last:  <INPUT TYPE="TEXT" NAME="last" SIZE="25" VALUE="Kosta"><BR>
First: <INPUT TYPE="TEXT" NAME="first" SIZE="25" VALUE="Charles"><BR>
Phone: <INPUT TYPE="TEXT" NAME="areaCode" SIZE="6" VALUE="123">
      <INPUT TYPE="TEXT" NAME="phone" SIZE="17" VALUE="4567890"><BR>
Email: <INPUT TYPE="TEXT" NAME="email" SIZE="25"
      VALUE="ckosta@bea.com"><BR>
      <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
      <INPUT TYPE="RESET" NAME="Reset" VALUE="Reset">
</FORM>
```

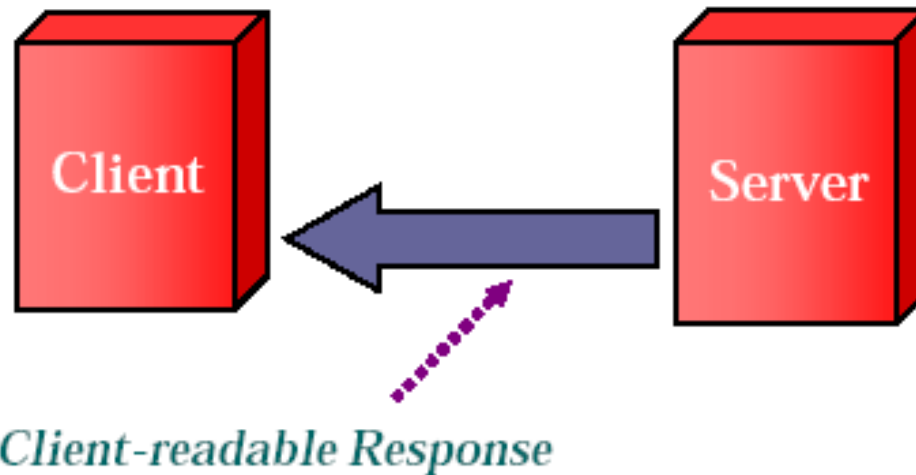


JAVA

Response Structure



- A servlet uses an instance of `HttpServletResponse` to send data to the client.



JAVA



HttpServletResponse 接口



```
Public interface HttpServletResponse extends ServletResponse {  
    // adding cookies to your response  
    public void addCookie(Cookie);  
    // adding name/value pairs to the header  
    public boolean containsHeader(String);  
    public void setDateHeader(String, long);  
    public void setHeader(String, String);  
    public void setIntHeader(String, int);  
    // encoding response URLs  
    public String encodeRedirectURL(String);  
    public String encodeURL(String);  
    public void sendRedirect(String);  
    ...  
}
```



JAVA

An Example



- Let's write a servlet that:
 - writes HTML with the `PrintWriter` object instead of `OutputStream`
 - intermixes Java variables with HTML output



Response Example

Generated HTML Page:

```
<HTML>
  <HEAD><TITLE>
    Response Example
  </TITLE></HEAD>

  <BODY>
    <H1>Response Example</H1>
  </BODY>
</HTML>
```



JAVA

Status/Error Messages



- **The HttpServletResponse interface has many constants declared for status/errors:**

A Sampling of HTTP Constants:

```
public static final SC_OK;                // status code 200
public static final SC_NOT_FOUND;        // status code 404
public static final SC_BAD_REQUEST;     // status code 400
public static final SC_FORBIDDEN;      // status code 403
...
```

These Values Can Be Used With:

```
public void sendError(int);
public void sendError(int, String);
public void sendStatus(int);
public void sendStatus(int, String);
```



JAVA



状态保存

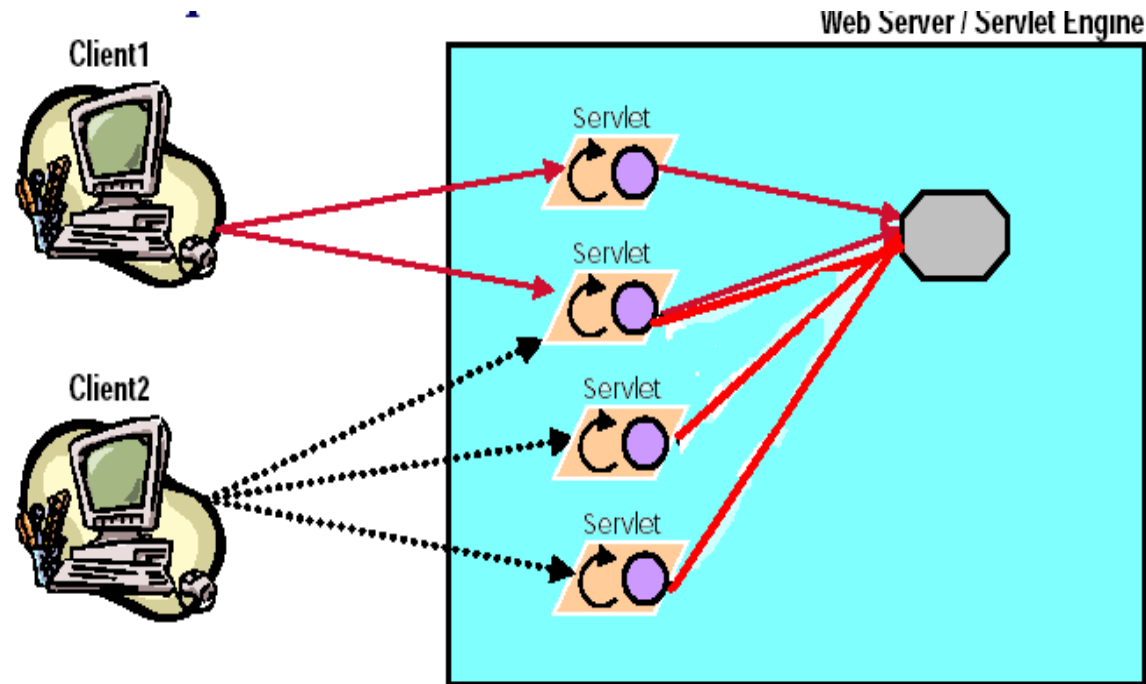
- Application (ServletContext)
- Session (HttpSession)
- Request ()
- Page (JSP)



Servlet间信息共享



- 可以实现多Servlet之间的信息共享（状态，资源等）
- 实际项目的应用
 - 聊天室
 - 系统静态数据



JAVA

设置属性



```
public class SpecialSetter extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/plain");
        PrintWriter out = res.getWriter();

        ServletContext context = getServletContext();
        context.setAttribute("com.costena.special.burrito", "Pollo Adobado");
        context.setAttribute("com.costena.special.day", new Date());

        out.println("The burrito special has been set.");
    }
}
```

获得属性



```
public class SpecialGetter extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        ServletContext context = getServletContext();
        String burrito = (String)
            context.getAttribute("com.costena.special.burrito");
        Date day = (Date)
            context.getAttribute("com.costena.special.day");

        DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM);
        String today = df.format(day);

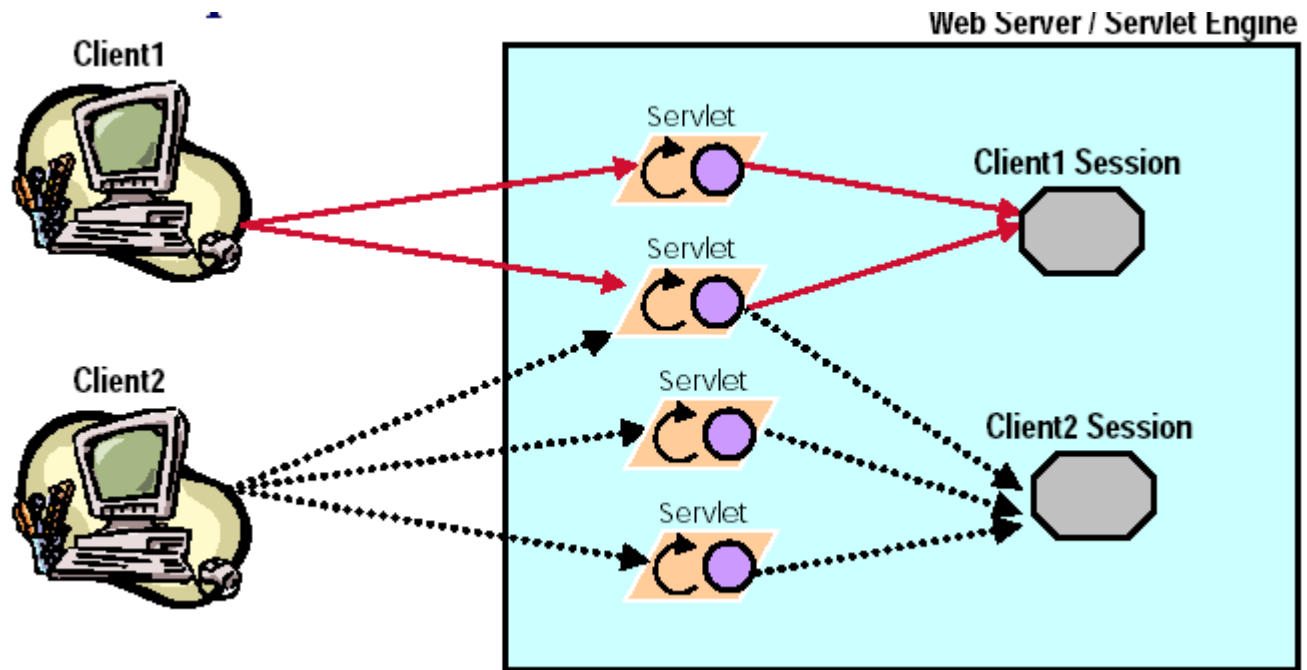
        out.println("Our burrito special today (" + today + ") is: " + burrito);
    }
}
```



共享Session



- A session for a client is shared among multiple servlets.



Web中会话跟踪



- 隐藏表单
- URL重写
- 使用Cookie
- HttpSession





取得 HttpSession



- The HttpSession object is the servlet's view of the session

访问一个Session:

```
public HttpSession getSession(boolean createNew);
```

参数 createNew:

true – 若存在session就返回，否则创建一个新的session

false -若存在session就返回，否则返回 null

如何开始一个新Session:

```
HttpSession session = request.getSession(true);
```



HttpSession接口



You can store session-specific data in an HttpSession object, as name/value pairs.

设置/获得属性的方法:

```
Object getAttribute(String name)
Enumeration getAttributeName()
void setAttribute(String name, Object value)
void removeAttribute(String name)
```

管理Session生命周期的方法:

```
String getId()
boolean isNew()
void invalidate() // kills this session
long getCreationTime()
long getLastAccessTime()
int getMaxInactiveInterval() // timeout period
void setMaxInactiveInterval()
```



JAVA

HttpSession例子



```
public class SessionSnoop extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        // Get the current session object, create one if necessary
        HttpSession session = req.getSession();
        Integer count = (Integer)session.getAttribute("snoop.count");
        if (count == null)
            count = new Integer(1);
        else
            count = new Integer(count.intValue() + 1);
        session.setAttribute("snoop.count", count);
        out.println("<HTML><HEAD><TITLE>SessionSnoop</TITLE></HEAD>");
        out.println("<BODY><H1>Session Snoop</H1>");
        out.println("<H3>Here is your saved session data:</H3>");
        Enumeration enum = session.getAttributeNames();
        while (enum.hasMoreElements()) {
            String name = (String) enum.nextElement();
            out.println(name + ": " + session.getAttribute(name) + "<BR>");
        }
        out.println("</BODY></HTML>");
    }
}
```





Session生命周期



- 会话生命周期的概述
- 正确选择超时值
 - 用户环境
 - 服务器的性能
- web.xml设置

```
<session-config>  
  <session-timeout>30</session-timeout>  
</session-config>
```
- `public void setMaxInactiveInterval(int interval)`
- `public int getMaxInactiveInterval()`



JAVA



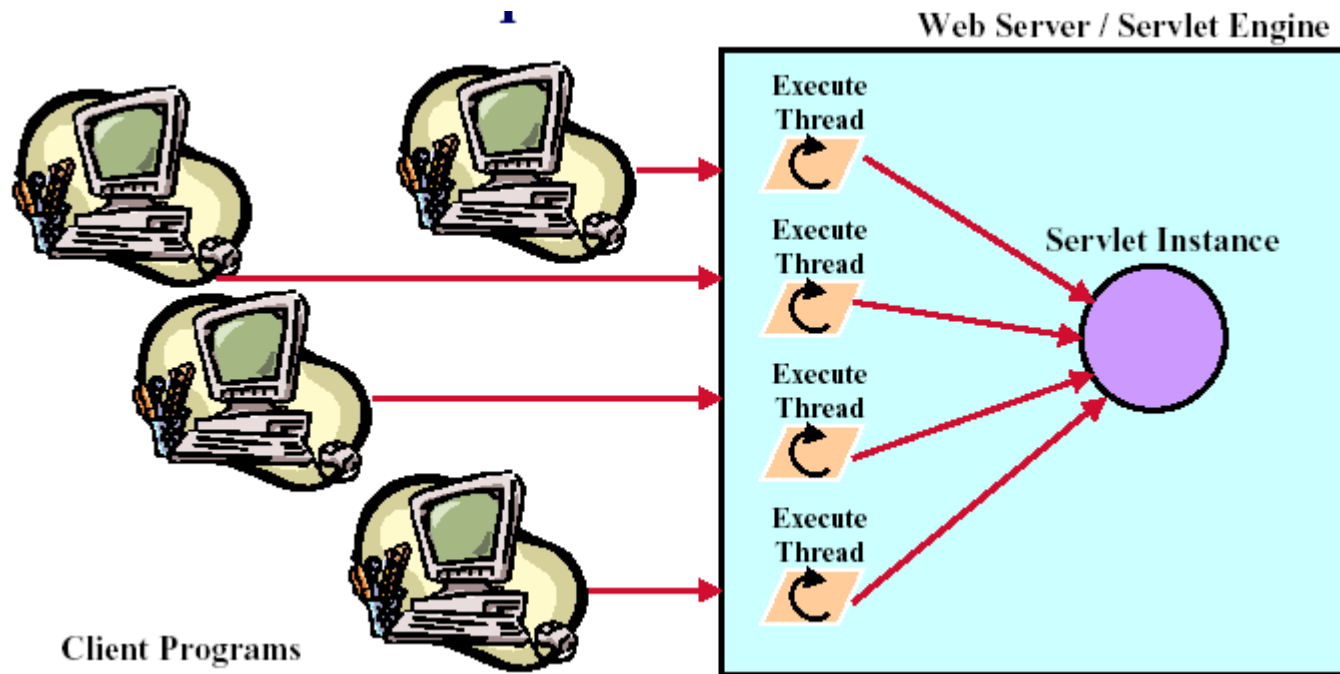
Servlets and Threads



多线程



- A servlet must be able to handle multiple, concurrent requests.



Servlet中的数据同步



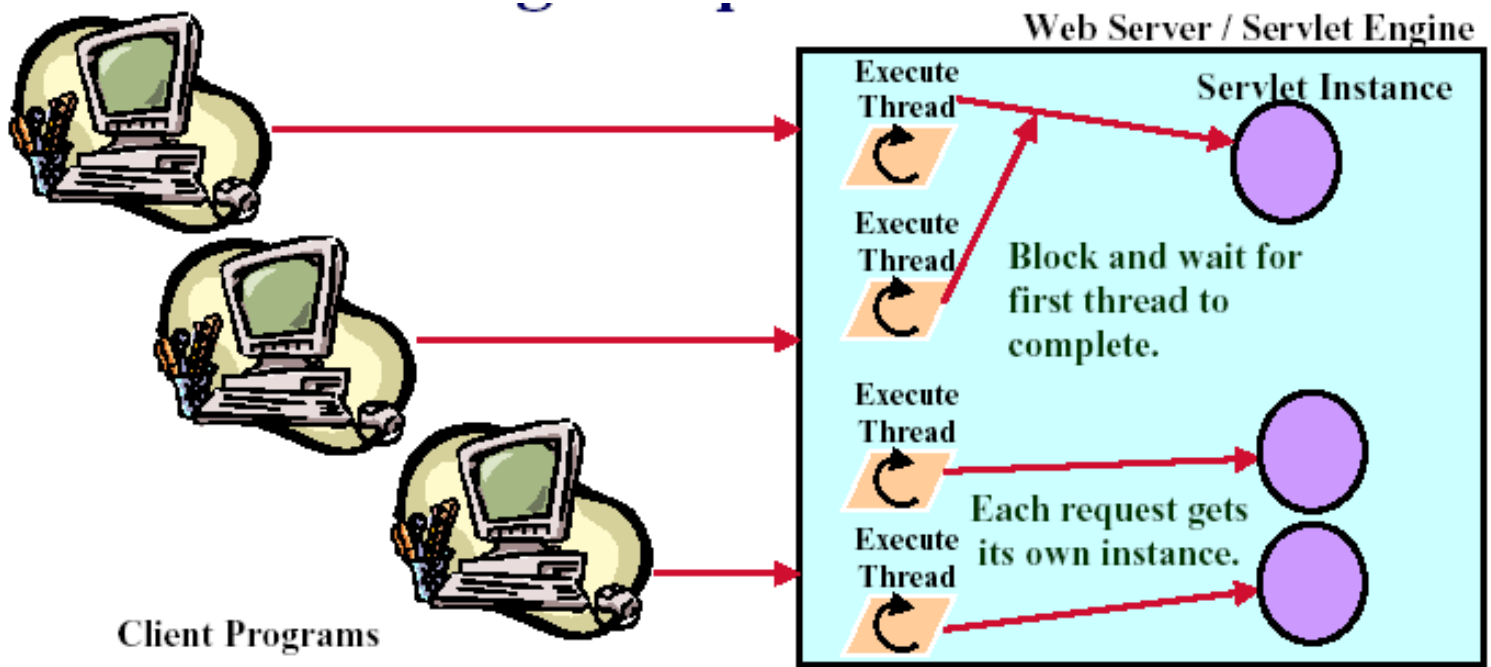
- Synchronized实现
- singelThreadModel



单线程



Servlets that implement the `SingleThreadModel` interface will only execute a single request thread.



JAVA

一个计数器的实现



```
public class SimpleCounter extends HttpServlet {  
  
    int count = 0;  
  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");  
        PrintWriter out = res.getWriter();  
        count++;  
        out.println("Since loading, this servlet has been accessed " +  
            count + " times.");  
    }  
}
```



线程安全



- 。同步体只允许一个单线程进入块(block)

Synchronized Block 语法：

```
synchronized (AnyObject) {  
    ...  
}
```

- 。The thread that can obtain **AnyObject**'s mutex flag gets to enter the block



例子



```
Import java.io.*;  
Import javax.servlet.*;  
Import javax.servlet.http.*;
```

```
Public class MultipleInstanceServlet extends HttpServlet {  
    Object lock = null;  
    public void init(ServletConfig cg) {  
        lock = new Object();  
    }  
    Public void service(HttpServletRequest request,  
        HttpServletResponse response) throws IOException,  
        ServletException {  
        synchronized(lock) {  
            ...  
        }  
        // regular code  
    }  
}
```



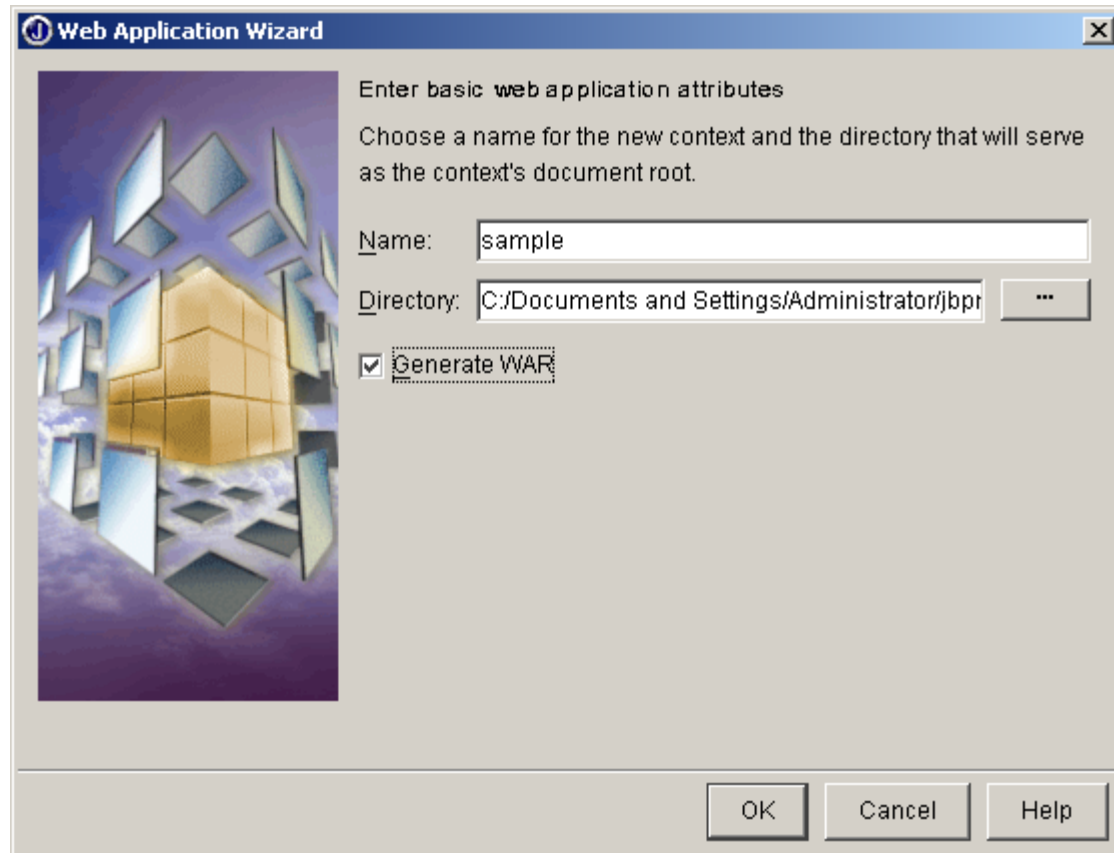
使用JBuilder开发Servlet



- Creating the project
- Creating the WebApp
- Creating the Servlet with the Servlet wizard
- Adding code to the Servlet
- Compiling and running the Servlet



Create WebApp



Creating the servlet with the Servlet wizard-1



Servlet Wizard - Step 1 of 4

Choose servlet name and type

Specify the class name of the servlet, its webapp, and the type of servlet to create.

Package:

Class:

Generate header comments

Single Thread Model

WebApp:

Standard servlet

Filter servlet

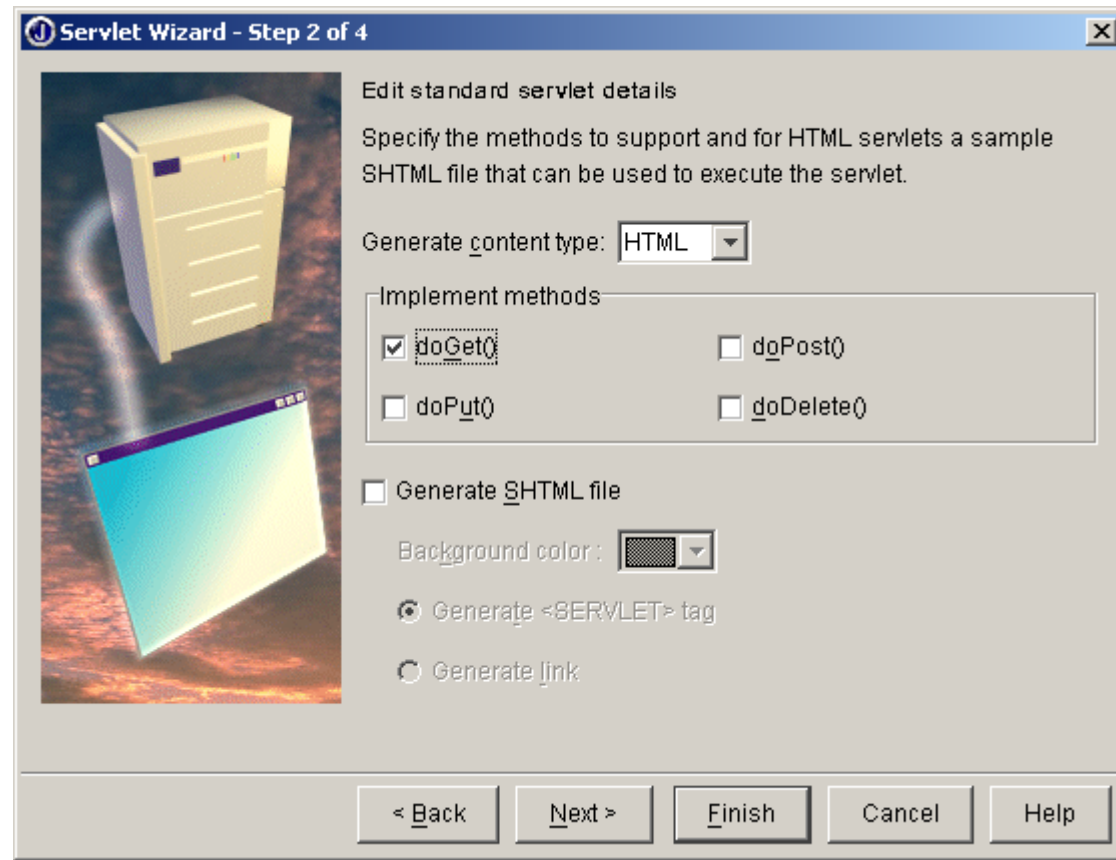
Listener servlet

< Back Next > Finish Cancel Help

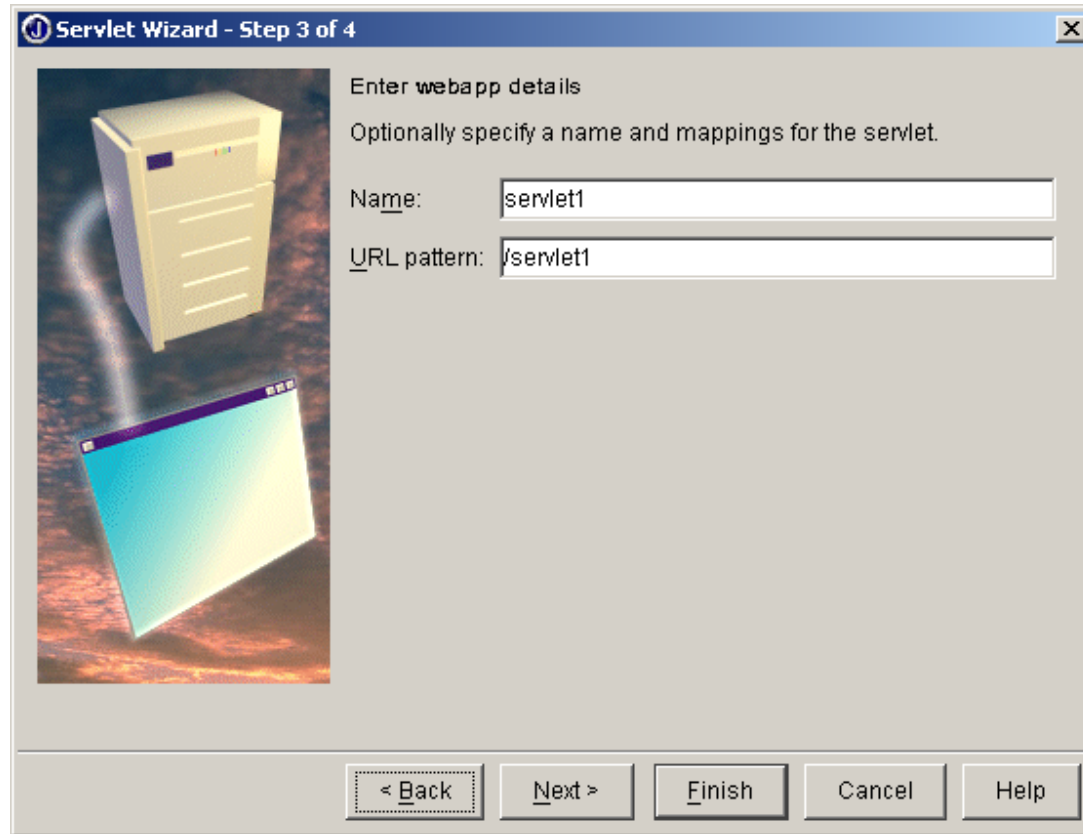
Creating the servlet with the Servlet wizard-2



JAVA



Creating the servlet with the Servlet wizard-3



Web Application

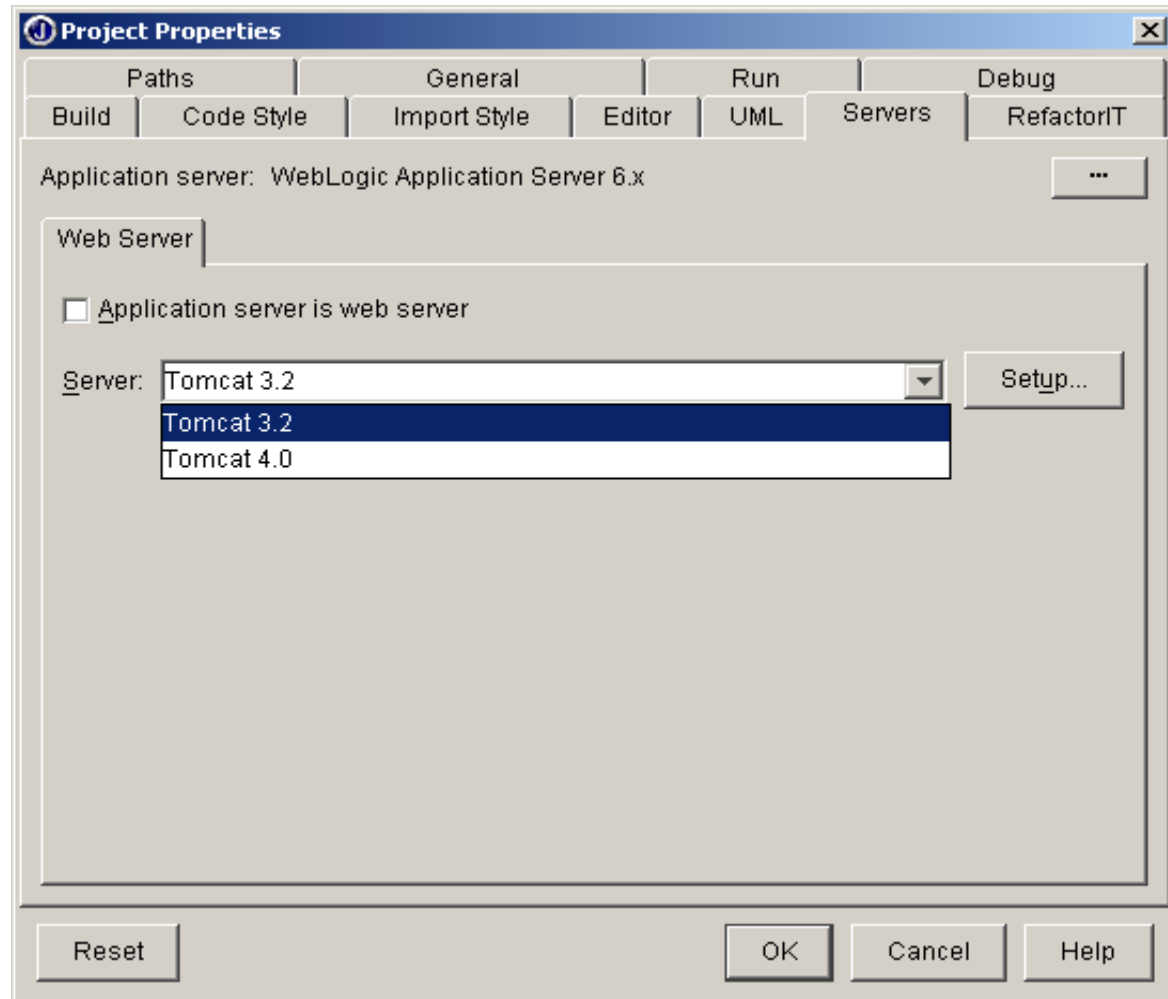


The screenshot shows the JBuilder IDE interface. The title bar reads "JBuilder - C:/Documents and Settings/Administrator/jbproject/untitled10/webproject/WEB-INF/web.xml". The menu bar includes File, Edit, Search, View, Project, Run, Team, Wizards, Tools, Window, and Help. The toolbar contains various icons for file operations and development. The main workspace is titled "Web Application 'sample'" and contains several fields: "Large icon:" and "Small icon:" with selection buttons; "Display name:" and "Description:" text boxes; "Session timeout:" text box; and a "Distributable" checkbox. Below these is a "Welcome files" area with a large empty box and a list of buttons: "Add", "Remove", "Move to Top", "Move Up", "Move Down", and "Move to Bottom". The left sidebar shows a project tree with folders like "untitled10.jpx", "junit.samples.money", "untitled10", "jsp", "sample", and "Deployment descriptors". The "WebApp Deployment Descriptor" tree is expanded, showing categories like "Context Parameters", "Servlets", "Tag Libraries", "MIME Types", "Error Pages", "Environment Entries", "EJB References", "Resource Manager Connector", "Login", and "Security". The bottom status bar shows "WebApp DD Editor" and tabs for "Source", "Transform View", and "History".



JAVA

Web Server Setup



介绍Tomcat



Tomcat是jakarta项目中的一个重要的子项目，其被JavaWorld杂志的编辑选为2001年度最具创新的java产品(Most Innovative Java Product)，同时它又是sun公司官方推荐的servlet和jsp容器(具体可以见<http://java.sun.com/products/jsp/tomcat/>)，因此其越来越多的受到软件公司和开发人员的喜爱。servlet和jsp的最新规范都可以在tomcat的新版本中得到实现。



Web Application Archive (WAR)



Servlet API V2.2 规范引入了一个新的概念帮助用户来发布用户的应用程序— Web Application Archive (WAR)。一个WAR文件包含了一个 Web 应用程序中所含有的元素 (Servlet代码, HTML和JSP页面等) 和应用程序描述文件来指定如何建立 Web 应用程序中所含有的元素, 我们可以将一个WAR文件的内容分为三类:

- 静态内容如HTML, JSP文件和图象文件
- 执行代码如Java class文件和包含Java库的jar文件
- 描述如何发布 Web 应用程序的发布描述符 (web.xml)



创建WAR文件



创建WAR文件建议流程：

- 建立一个目录以存放所有WAR文件中的内容，暂且称为WAS_ROOT，在WAS_ROOT创建一个名为WEB_INF的目录。
- 按照层次关系在WAS_ROOT下存放静态文件。
- 在WEB-INF/classes目录下按照包的名字放置Java Class文件。
- 在WEB-INF/lib目录下放置JAR文件。
- 在WEB-INF下目录下创建发布描述文件，发布描述文件是XML格式的文件，关于对应的DTD的定义在Servlet API V2.2有详细的描述。
- 创建WAR文件，你可以使用JDK提供的JAR工具。





ServletConfig



- 从配置信息获得参数
`String var0 = request.getParameter("param0");`
- web.xml中servlet参数
`<init-param>
 <param-name>name</param-name>
 <param-value>huihoo</param-value>
</init-param>`



Servlet Mapping技术



- 明确映射
- 路径前缀映射
- 扩展名映射
- 默认映射



Servlet Mapping例子



```
<servlet>
  <servlet-name>hi</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name> hi</servlet-name>
  <url-pattern> /hello.html</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name> hi</servlet-name>
  <url-pattern> *.hello</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name> hi</servlet-name>
  <url-pattern> /hello/* </url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name> hi</servlet-name>
  <url-pattern> /hello.html</url-pattern>
</servlet-mapping>
```



Web 应用程序初始化参数



- 配置

```
<context-param>  
    <param-name>count</param-name>  
    <param-value>10</param-value>  
</context-param>
```

- 读取

```
strName=this.getServletContext().getInitParameter("name");
```

Servlet启动顺序



```
<servlet>  
    <servlet-name>svlInitSystem</servlet-name>  
    <servlet-  
class>app.common.SvlInitSystem</servlet-class>  
    <load-on-startup>1</load-on-startup>  
</servlet>
```



欢迎页面



```
<welcome-file-list>  
  <welcome-file>index.html</welcome-file>  
  <welcome-file>index.jsp</welcome-file>  
</welcome-file-list>
```



weblogic.xml



- description Element
- weblogic-version Element
- security-role-assignment Element
- reference-descriptor Element
- resource-description Element
- ejb-reference-description Element
- session-descriptor Element
- Session Parameter Names and Values
- jsp-descriptor Element
- JSP Parameter Names and Values
- container-descriptor Element
- check-auth-on-forward Element
- redirect-with-absolute-url
- charset-params Element
- input-charset Element
- charset-mapping Element



JAVA

Application Event Listener



- Servlet Event listener are classes that respond to notifications about
 - ServletContext
 - HttpSession



JAVA

- ServletContextListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener

Listener



- ServletContextListener
- ServletContextAttributeListener,
- HttpSessionListener
- HttpSessionAttributeListener



Event



- ServletContextEvent
- ServletContextAttributeEvent
- HttpSessionEvent
- HttpSessionBindingEvent





Sample



配置



```
<listener>  
  <listener-class>app.listener</listener-class>  
</listener>
```

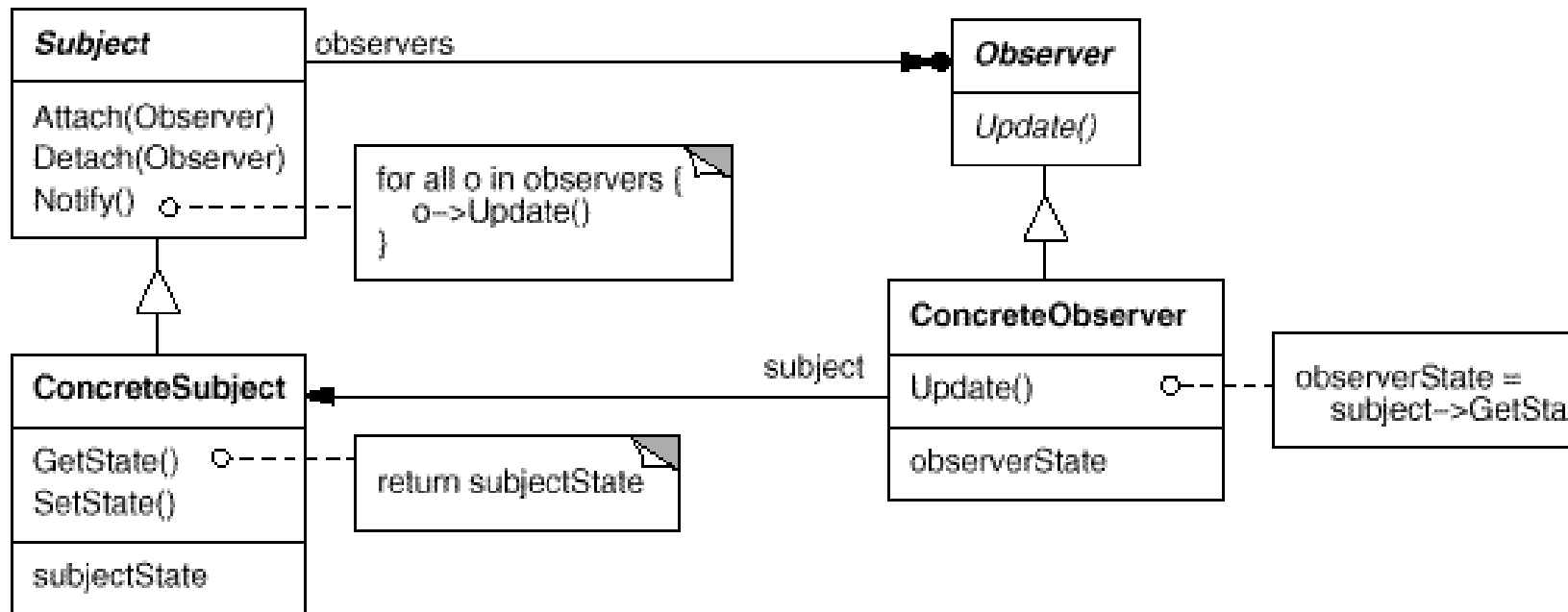


GOF – Observer(观察者)



■ 意图

定义对象间一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。



Servlet Filter



- 被用于:
 - 加密、解密
 - 压缩，解压缩
 - 认证，安全
 - 字符编码
 - Log处理



JAVA

Servlet Filter



- Change request
- Change response



Develop Filter



- Implement `javax.servlet.Filter` interface
 - `init(FilterConfig filterConfig)`
 - `doFilter(ServletRequest request, ServletResponse response, FilterChain filterChain)`
 - `void destroy()`
- Get parameters from
 - `ServletContext`
 - `FilterConfig`



JAVA



Sample



部署



```
<filter>
  <filter-name>servlet3</filter-name>
  <filter-class>untitled7.Servlet3</filter-class>
  <init-param>
    <param-name>param</param-name>
    <param-value />
  </init-param>
</filter>

<filter-mapping>
  <filter-name>servlet3</filter-name>
  <url-pattern>/* </url-pattern>
</filter-mapping>
```



JAVA

Filter Chain Order



Filter execute in the order they are define in the [web.xml](#) file

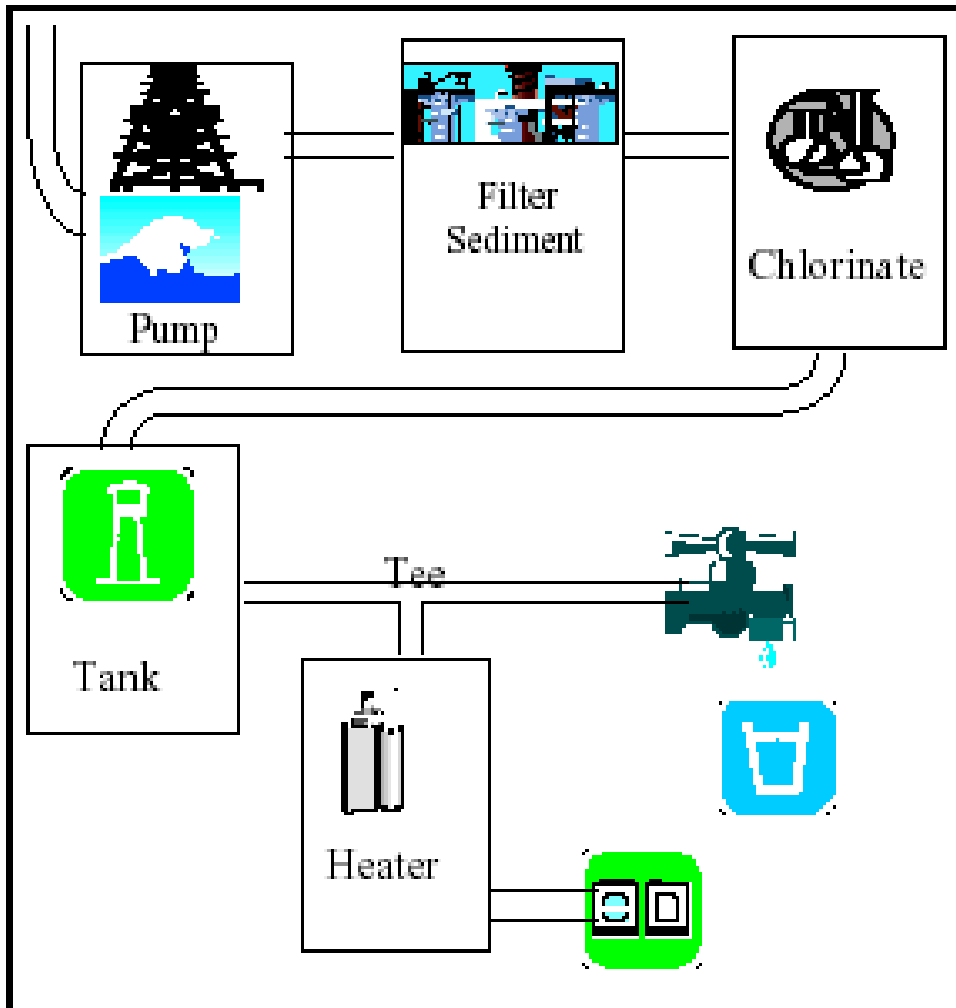
```
<filter>
  <filter-name>authfilter</filter-name>
  <filter-class>untitled7.AuthFilter</filter-class>
</filter>
<filter>
  <filter-name>logfilter</filter-name>
  <filter-class>untitled7.Logfilter</filter-class>
</filter>
<filter>
  <filter-name>servlet3</filter-name>
  <filter-class>untitled7.Servlet3</filter-class>
</filter>
```





管道和过滤器 (Pipes and Filters)

体系结构模式为处理数据流的系统提供了一种结构。每个处理步骤封装在一个过滤器组件中。数据通过相邻过滤器之间的管道传输。重组过滤器可以建立相关系统族。



Public water systems demonstrate *Pipes and Filters*. The water flowing through pipes is the input to a filter. Pipes are also used as the output from a filter. This example shows that name correspondence between the example and pattern can produce a viable example.



高级话题

发送多媒体信息



- 动态生成图表
- 动态生成报表
- 动态生成特殊文件



Sample



```
public class HelloWorldGraphics extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        ServletOutputStream out = res.getOutputStream(); // binary
        output!

        Frame frame = null;
        Graphics g = null;

        try {
            // Create an unshown frame
            frame = new Frame();
            frame.addNotify();

            // Get a graphics region, using the Frame
            Image image = frame.createImage(400, 60);
            g = image.getGraphics();
        }
    }
}
```



JAVA

Sample



```
// Draw "Hello World!" to the off-screen graphics context
```

```
g.setFont(new Font("Serif", Font.ITALIC, 48));  
g.drawString("Hello World!", 10, 50);
```

```
// Encode the off-screen image into a GIF and send it to the client
```

```
res.setContentType("image/gif");
```

```
GifEncoder encoder = new GifEncoder(image, out);  
encoder.encode();
```

```
}
```

```
finally {
```

```
    // Clean up resources
```

```
    if (g != null) g.dispose();
```

```
    if (frame != null) frame.removeNotify();
```

```
}
```

```
}
```

```
}
```



JAVA

其他技术



- Servlet生成SVG图
- 参数的解析
- 数据文件的上传
- Servlet与XML结合



参考资料



- <http://java.sun.com/products/servlet/>
- sun公司的servlet站点

- <http://www.huihoo.com>
国内一个关于中间件的专业站点



结束



谢谢大家！

allen@huihoo.com

<http://www.huihoo.com>

