

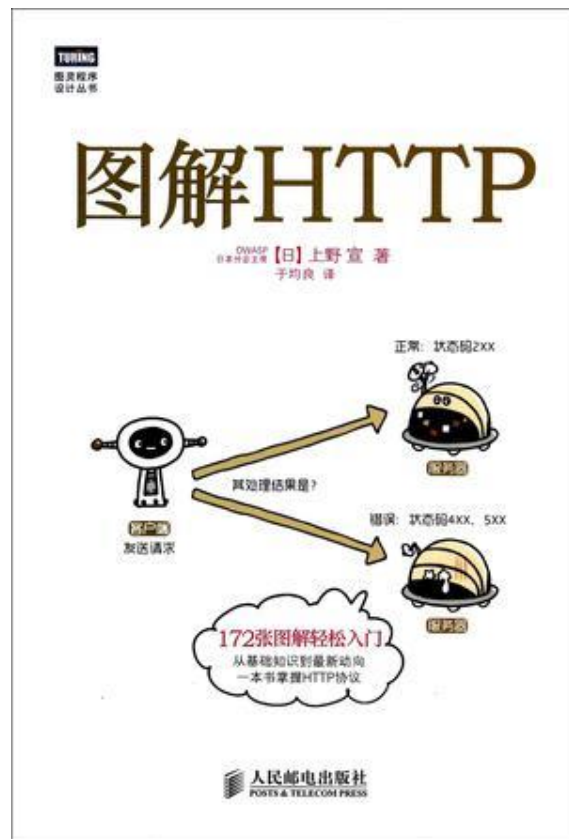
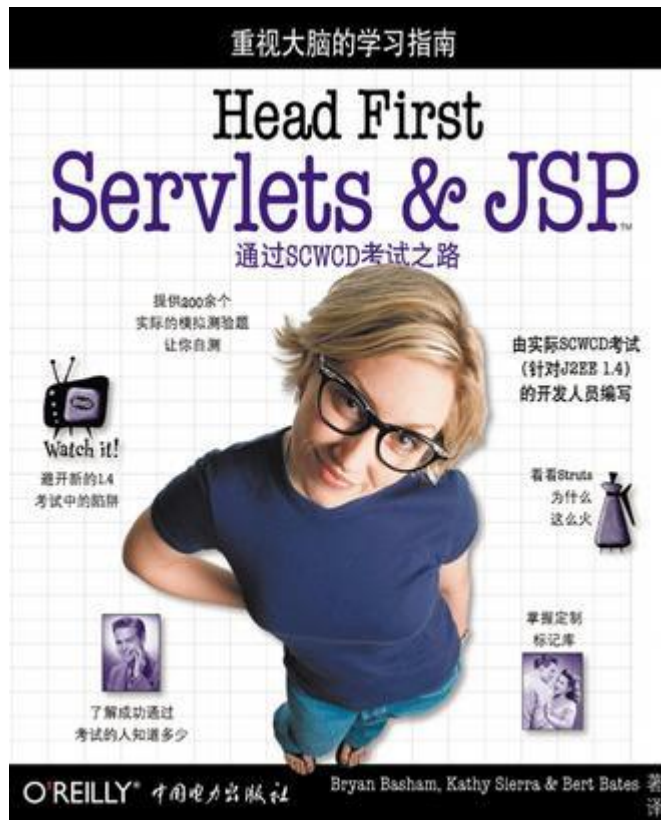


JSP/Servlet

凯盛软件

推荐书籍

凯盛软件





Google 搜索

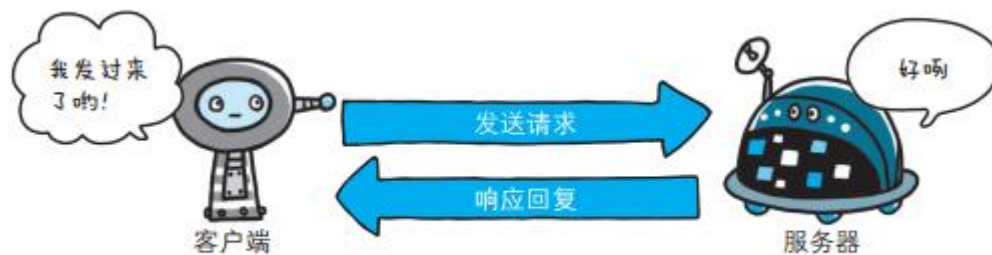
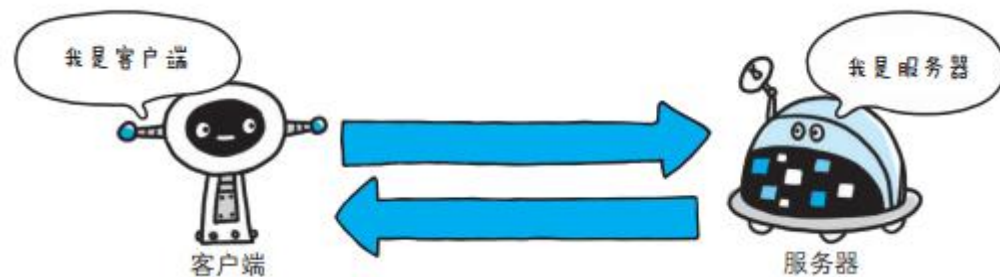
手气不错

Google.com.hk 使用下列语言：[中文（繁體）](#) [English](#)

Http下午茶 : <http://www.kancloud.cn/kancloud/tealeaf-http/43837>

HTTP协议用于客户端和服务端之间的通讯

凯盛软件

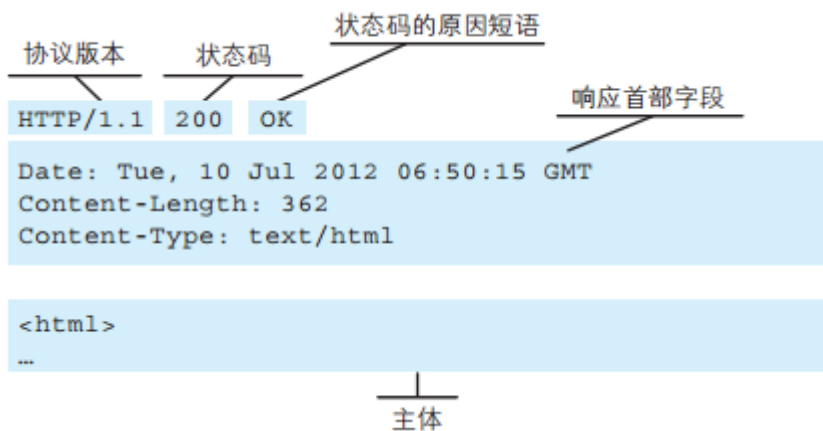


图：请求必定由客户端发出，而服务器端回复响应

```
GET /index.html HTTP/1.1
Host: www.kaishengit.com
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6,es;q=0.4,ja;q=0.2,sk;q=0.2,zh-TW;q=0.2,it;q=0.2,de;q=0.2,gl;q=0.2
```



```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 11 Mar 2016 07:13:13 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Last-Modified: Tue, 08 Mar 2016 02:41:55 GMT
Via: (null)
Content-Encoding: gzip
```

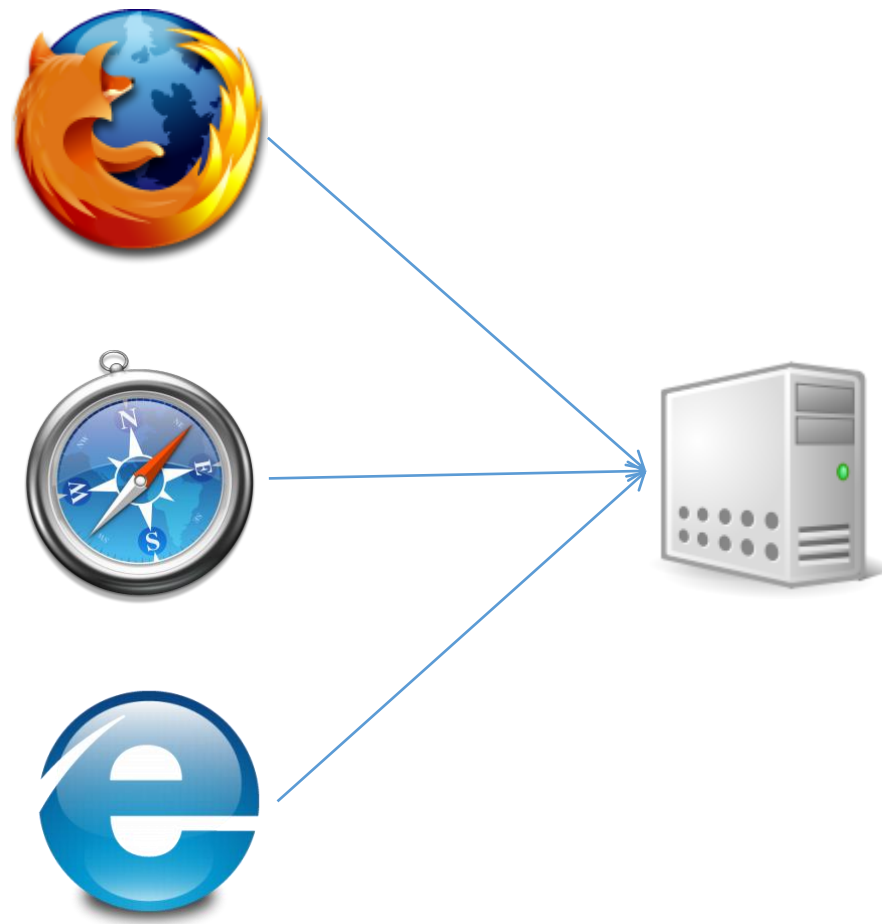


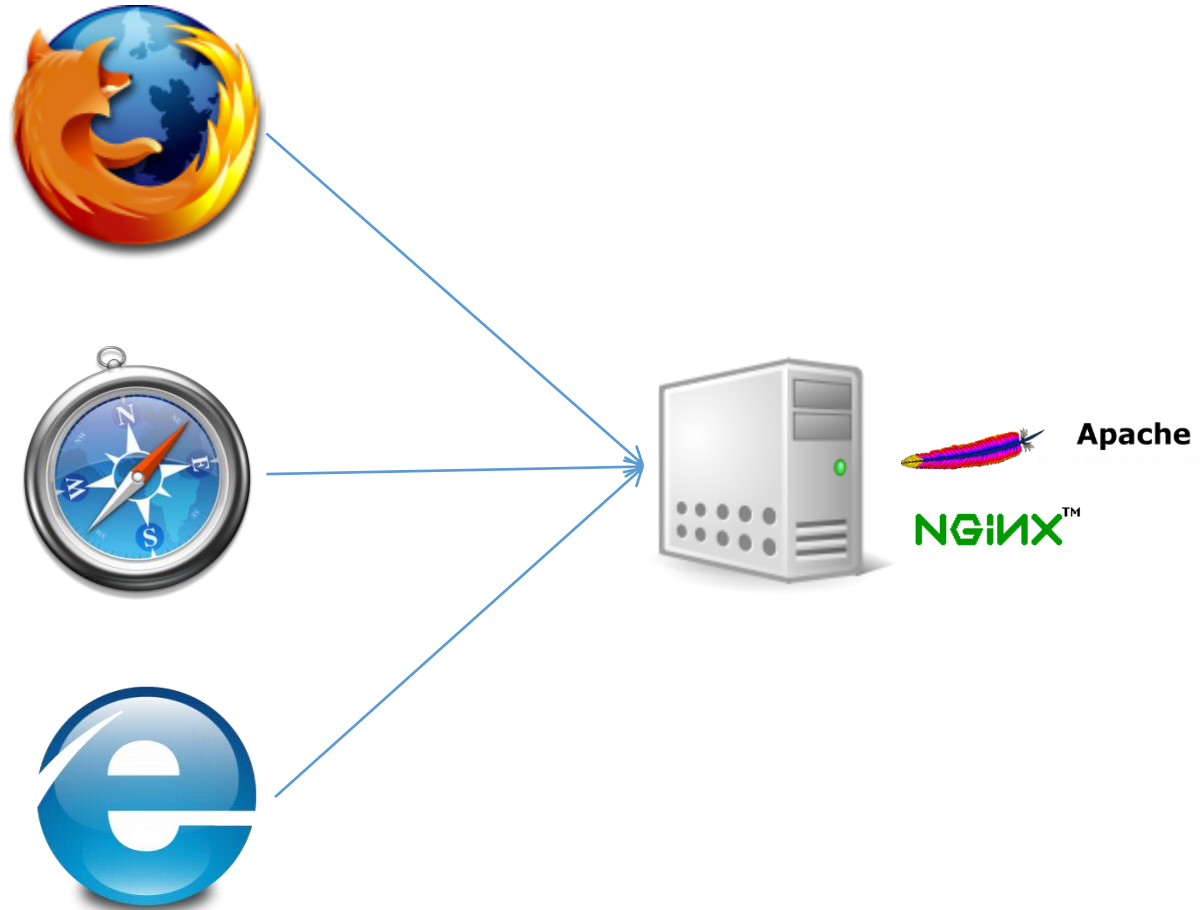
服务器 客户端

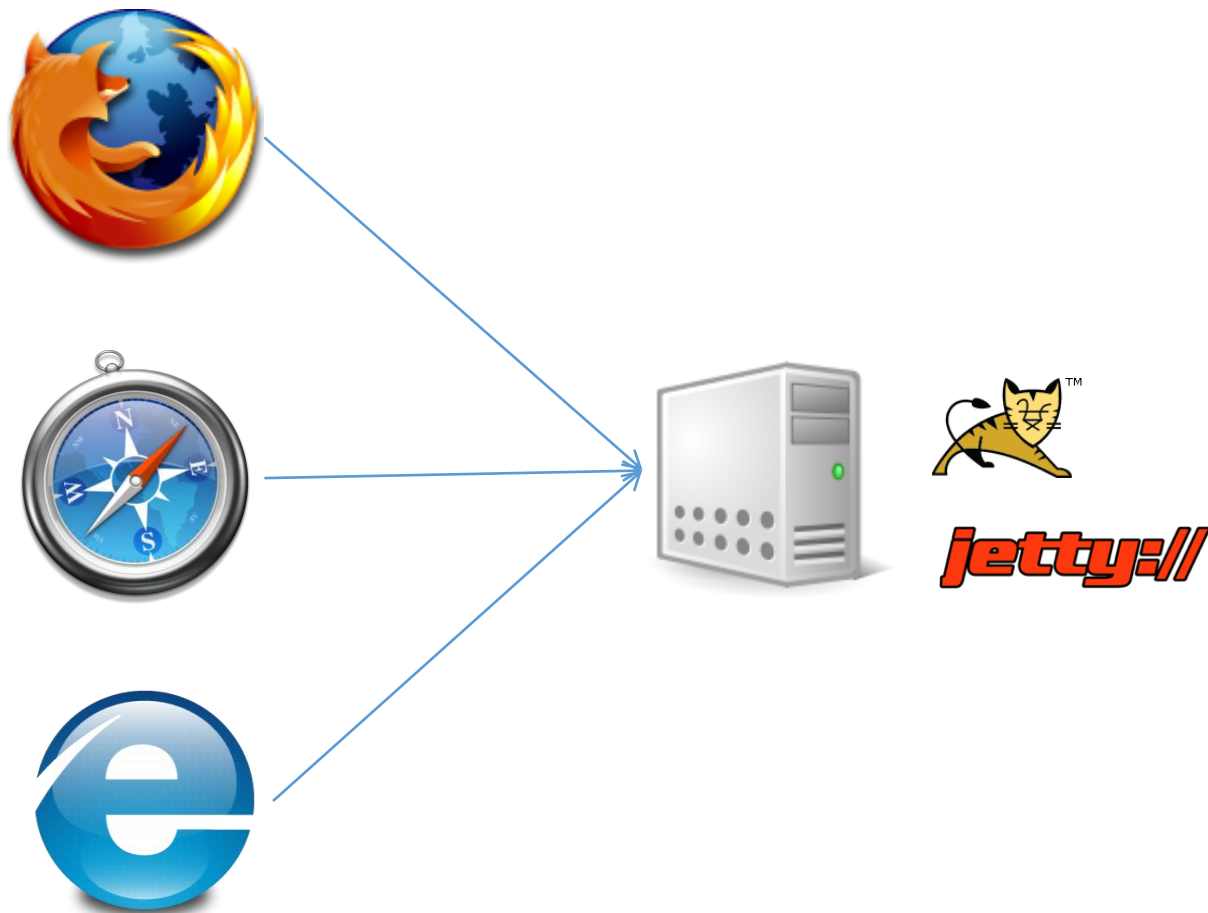
凯盛软件

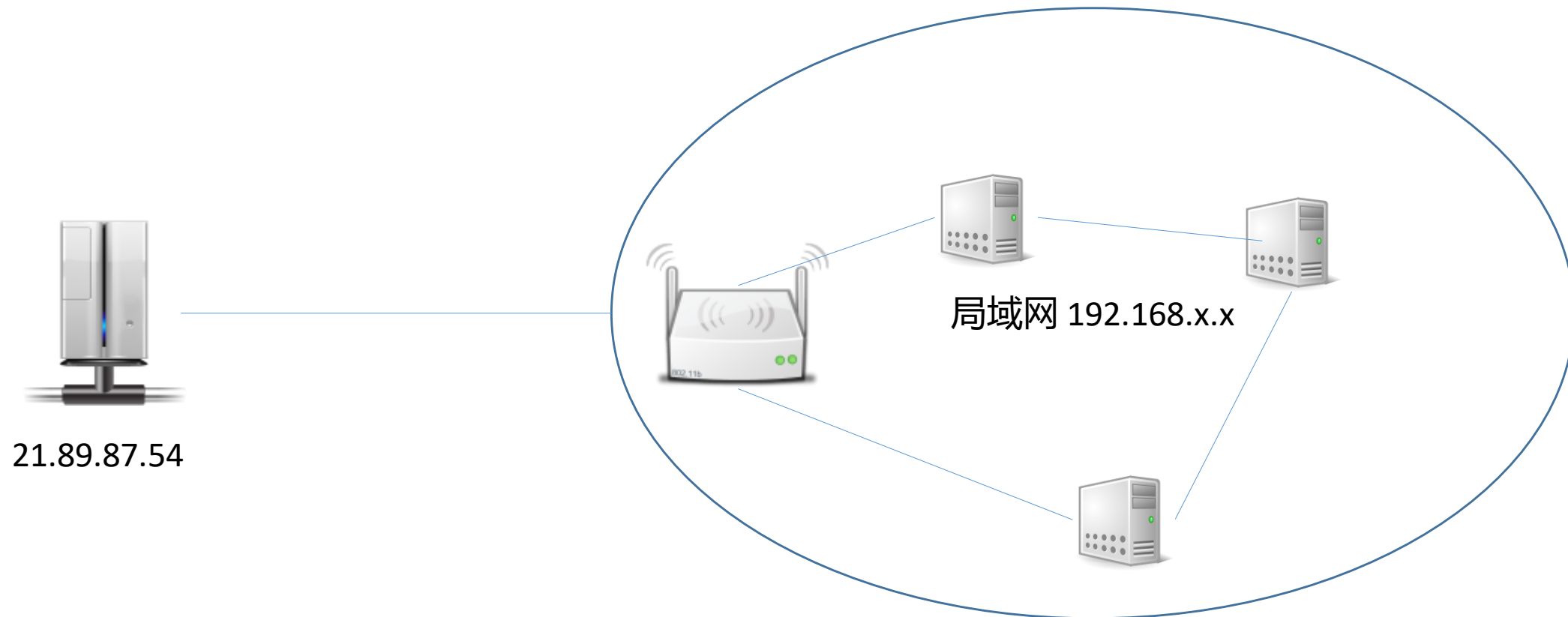
服务器：提供可访问资源的机器

客户端：可以进行HTTP访问的设备







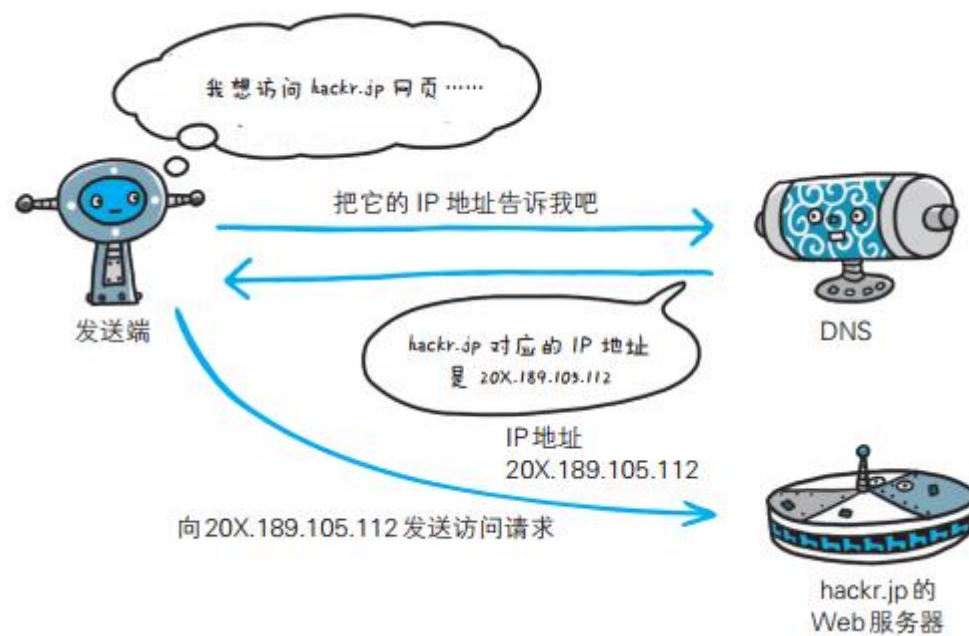


所有的电脑无论在是否联网的情况下都有一个本机IP地址：

127.0.0.1

也可以通过

localhost来访问



http://user:pass@www.example.jp:80/dir/index.htm?uid=1#ch1

协议
方案名

登录信息
(认证)

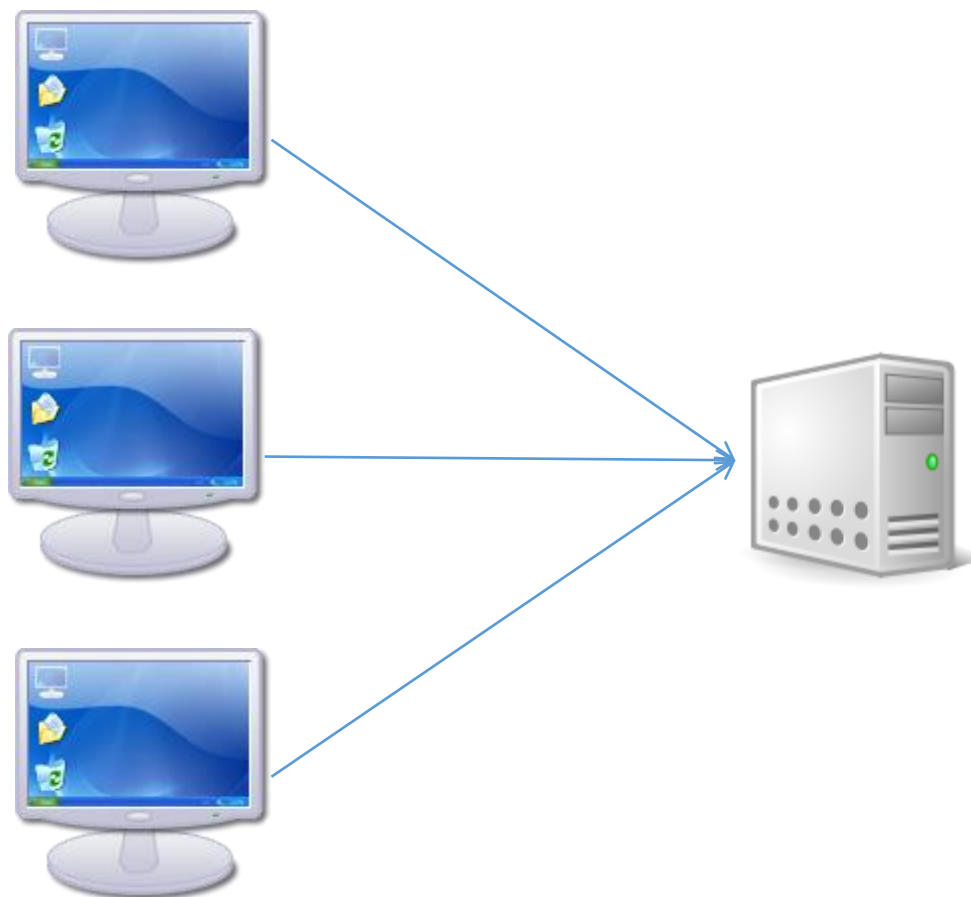
服务器地址

服务器端口号

带层次的文件路径

查询字符串

片段标识符






```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>我的Web项目</title>
  </head>

  <body>
    Hello , 这是JSP页面 !
  </body>
</html>
```

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>我的Web项目</title>
  </head>

  <body>
    <%
      for(int i = 0;i < 10;i++) {
    %>
    Hello , 这是JSP页面 ! <br/>
    <%} %>
  </body>
</html>
```

welcome file

凯盛软件

/WEB-INF/web.xml

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

HTTP状态码

凯盛软件

- 404

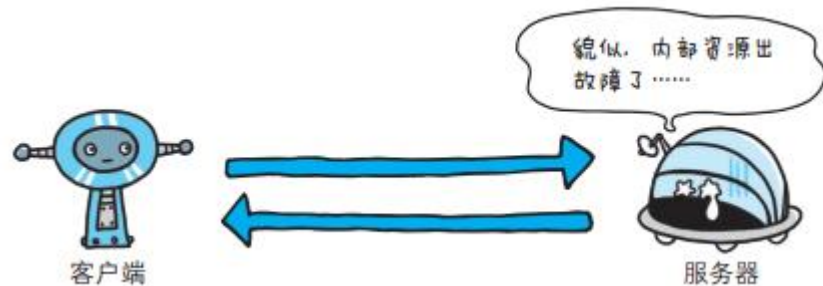
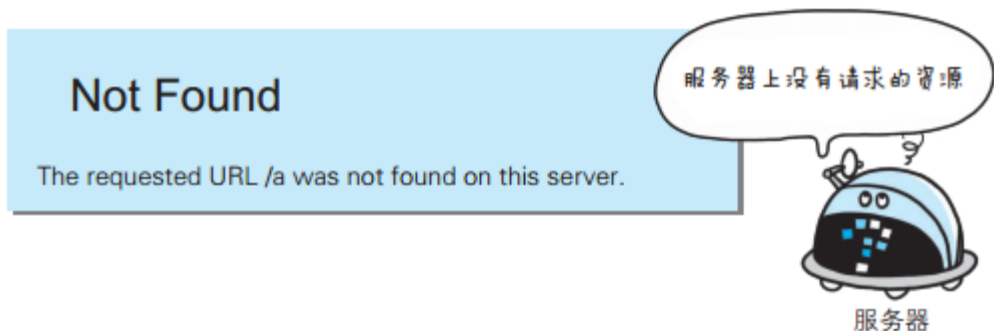
当请求的资源在服务器中找不到时，返回404

- 500

服务器运行出错时(例如JSP中10/0)，返回500

- 200

服务器一切正常，返回200

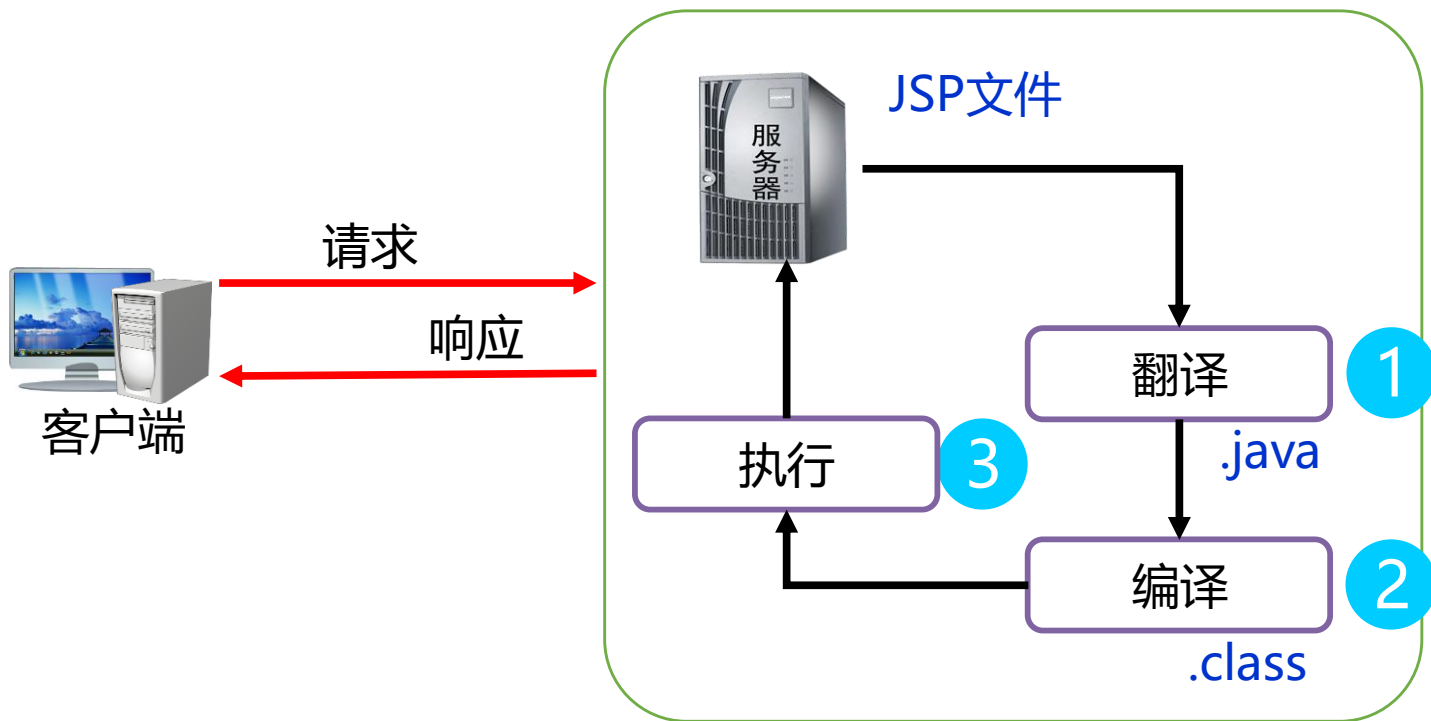


<http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>



第一次执行

凯盛软件





注意：如果对JSP文件进行了修改，Web容器会重新对JSP文件进行翻译和编译

- 修改端口号

```
<Connector port="8080" protocol="HTTP/1.1"
```

```
    connectionTimeout="20000"
```

```
    redirectPort="8443" />
```

- 配置虚拟目录

```
<Engine name="Catalina" defaultHost="localhost">
```

```
    <Host name="localhost" autoDeploy="true">
```

```
        <Context docBase="E:\\Workspaces\\test\\WebRoot" path="" reloadable="false" />
```

```
    </Host>
```

```
</Engine>
```


- 静态内容
- 指令
- 表达式
- scriptlet
- 声明
- 注释


获取表单的内容

- index.jsp

```
<form action="login.jsp" method="post">  
    用户名:<input type="text" name="username"/><br/>  
    密码:<input type="password" name="userpwd"/><br/>  
    <input type="submit" value="进入系统"/>  
</form>
```

- login.jsp

```
<%  
String name = (String)request.getParameter("username");  
String pwd = (String)request.getParameter("userpwd");  
%>
```



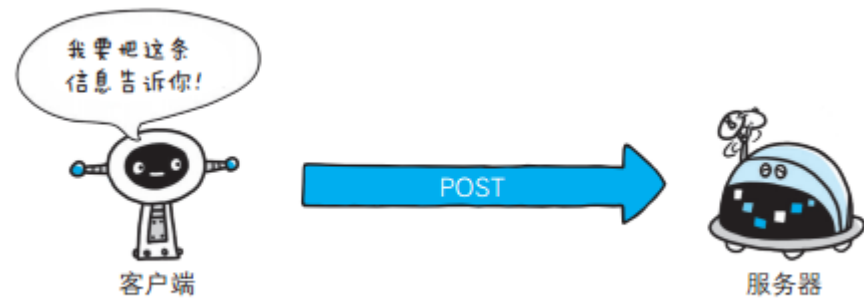
Get和Post

凯盛软件

GET：获取资源



POST：传输实体主体



页面跳转

- login.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String name = (String)request.getParameter("username");
String pwd = (String)request.getParameter("userpwd");


if("tom".equals(name) && "123".equals(pwd)) {
    response.sendRedirect("home.jsp");
} else {
    response.sendRedirect("index.jsp");
}
%>
```

- login.jsp

```
if("tom".equals(name) && "123".equals(pwd)) {  
    response.sendRedirect("home.jsp");  
} else {  
    response.sendRedirect("index.jsp?error=10001");  
}
```

- index.jsp

```
<%  
    String error = request.getParameter("error"); //获取URL中的值  
    if("10001".equals(error)) {  
        <div style="color:red">用户名或密码错误</div>  
    } %>
```



请求转发和跨页面传值

- login.jsp

```
if("tom".equals(name) && "123".equals(pwd)) {  
    request.setAttribute("loginName", name); //设置跨页面传值  
    RequestDispatcher rd = request.getRequestDispatcher("home.jsp");  
    rd.forward(request, response); //进行请求转发跳转  
} else {  
    response.sendRedirect("index.jsp?error=10001");  
}
```

- home.jsp

```
<body>  
    <%  
        String name = (String)request.getAttribute("loginName");  
    %>  
    Welcome, <%=name %>  
</body>
```

- 重定向跳转
 - 地址栏改变
 - 不能跨页面传值
- 请求转发跳转
 - 地址栏不变
 - 可以跨页面传值

MVC模型指的是模型-视图-控制器 (**Model-View-Controller**)。MVC模型有助于将应用程序分割为三个组件，使得程序设计更加的容易，并使各个组件之间的耦合度降到最低。MVC中三个组件分别为：模型 (Model)、视图 (View) 和控制器 (Controller)。

- 模型

模型代表应用程序的数据以及用于访问控制和修改这些数据的业务规则

- 视图

视图用来组织模型的内容。视图对象使用对象模型的查询方法以获得数据，并将数据显示给用户

- 控制器

控制器定义了应用程序的行为。它负责对来自视图的用户请求进行解析，并把这些请求映射成相应的行为，这些行为由模型负责实现



Models

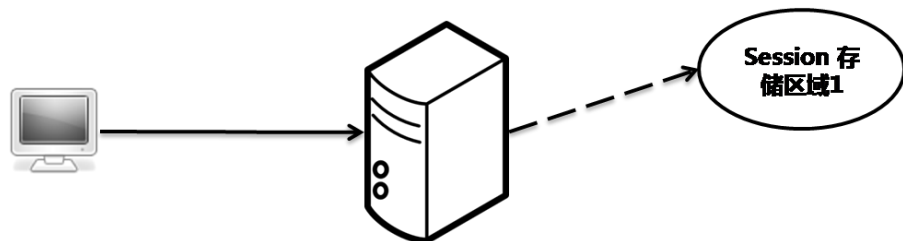


Views

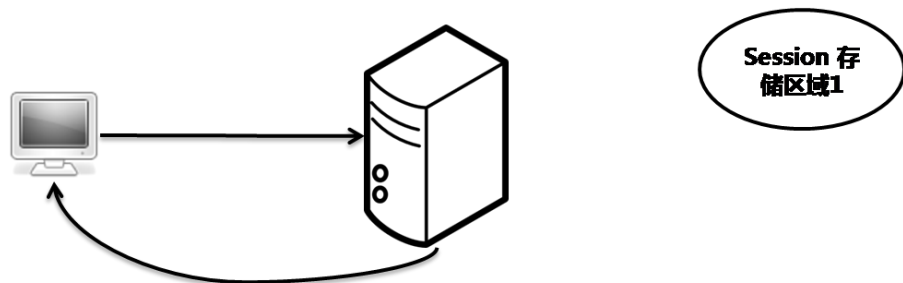


Controllers

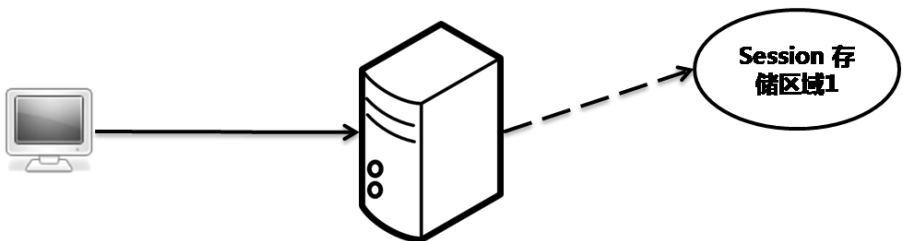




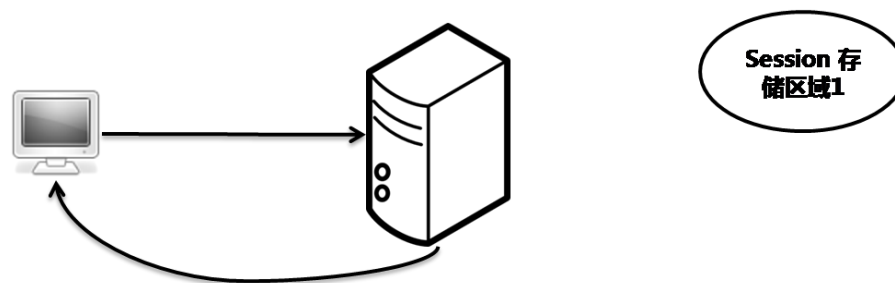
登录银行系统将用户名等资料存储到session1中



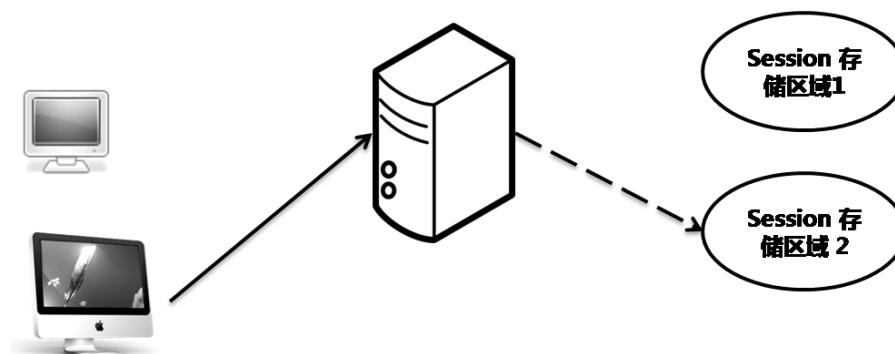
服务器给客户端做出响应提示登录成功



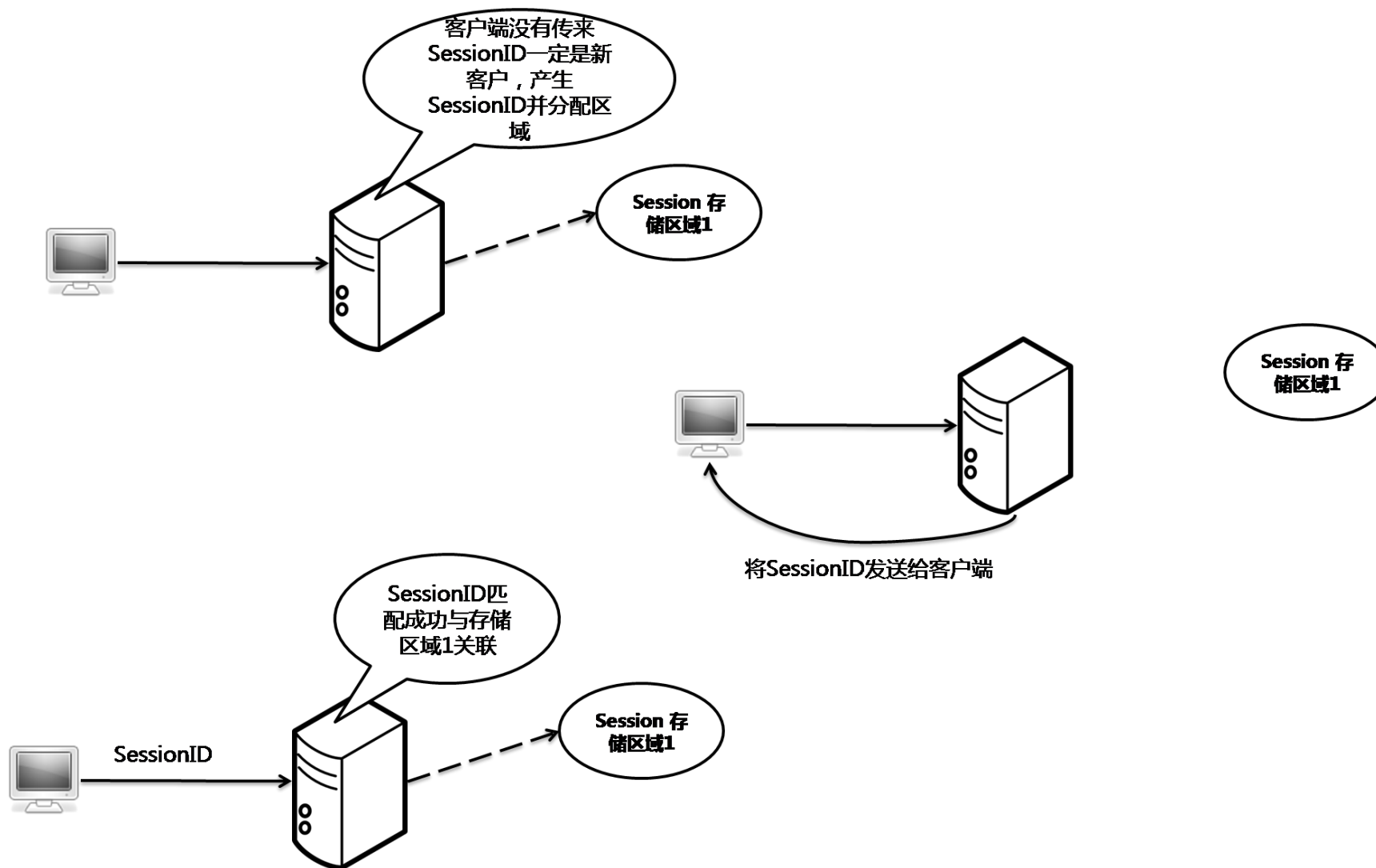
客户端点击支付发出请求，服务器根据session1知道是同一用户



服务器给客户端做出响应提示支付成功

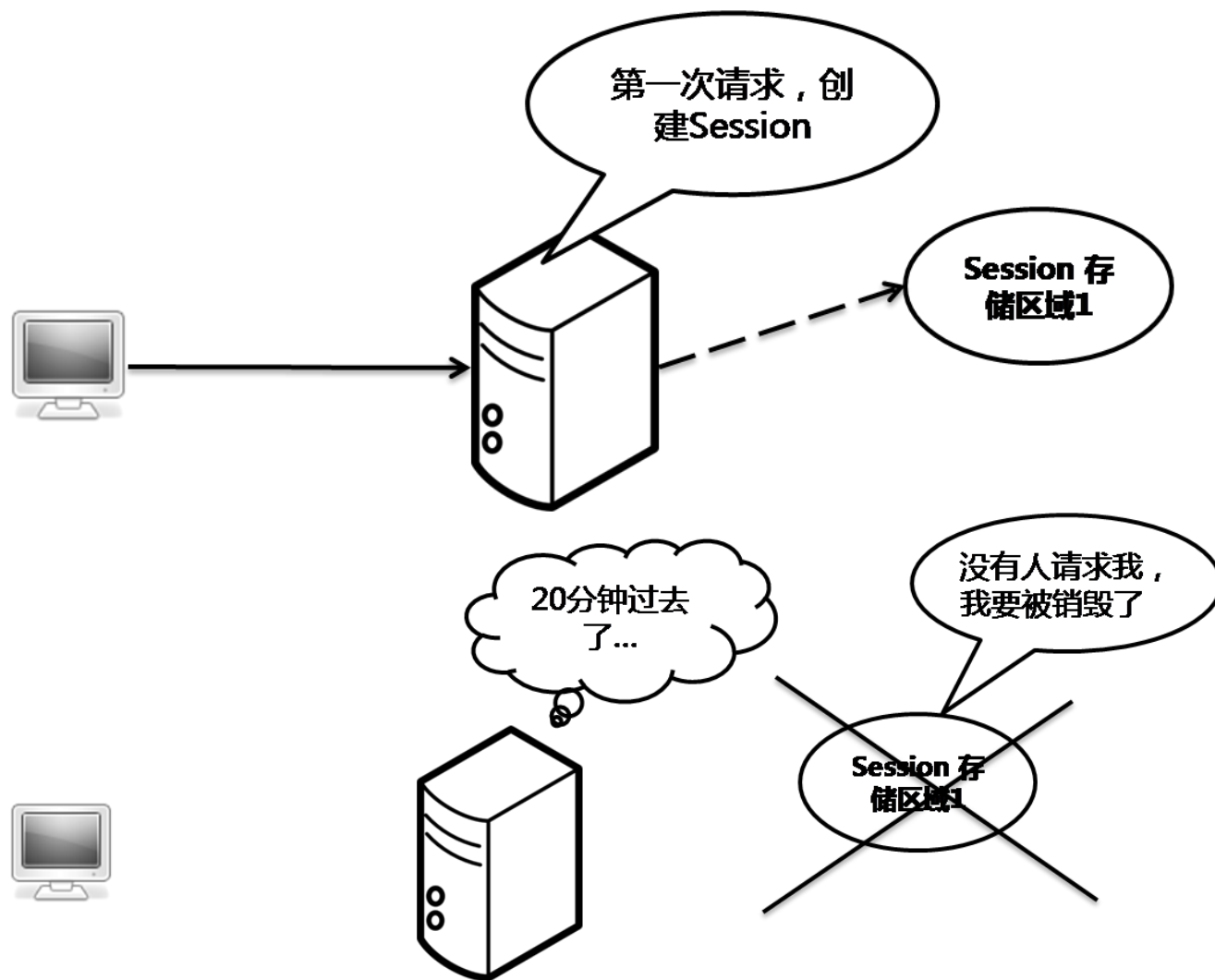


新来的一个客户端，服务器把它的信息放到session2中



方法名	返回值类型	定义
getId()	String	获得当前Session的SessionID
setAttribute(String key,String value)	void	向Session空间中存储对象。
getAttribute(String key)	Object	从当前的Session空间中获取key对应的对象。
invalidate()	void	强制Session过期
removeAttribute(String key)	void	从当前的Session空间中删除key对应的对象。
getCreationTime()	long	获得当前Session创建的时间
getLastAccessedTime()	long	获得客户端最后一次请求服务器的时间。
setMaxInactiveInterval(int interval)	void	设置Session的最大请求间隔时间,单位为秒
getMaxInactiveInterval()	int	获得Session的最大请求间隔时间，单位为秒
isNew()	boolean	判断一个Session是不是一个新的Session

- 过期（默认30分钟）
 - 调用getMaxInactiveInterval()方法设置
 - 在web.xml中设置
- 调用invalidate()方法强制其死亡
- Web容器关闭或重启



继承自javax.servlet.http.HttpServlet

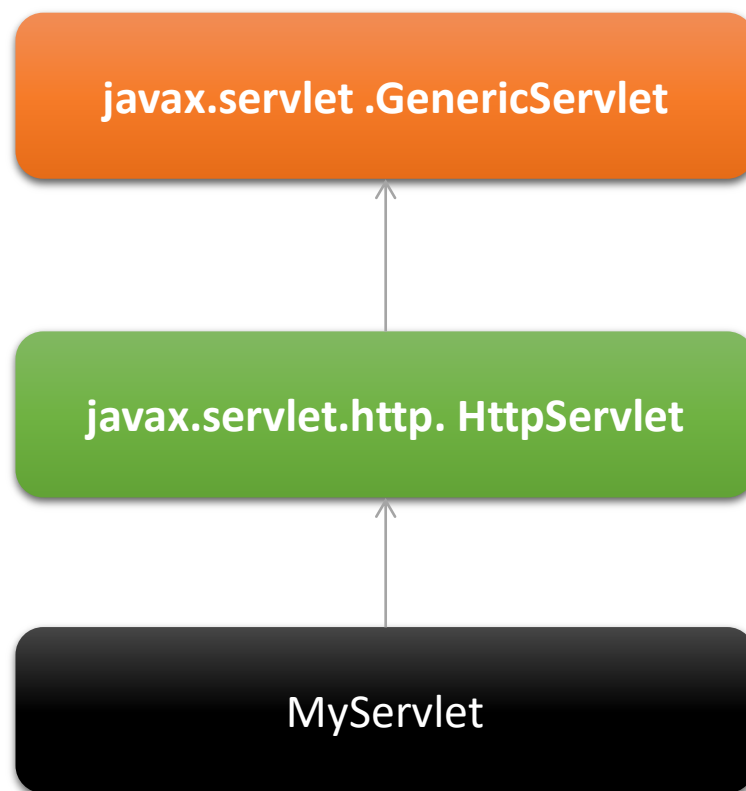
```
public class AppServlet extends HttpServlet {  
  
    private static final long serialVersionUID = 1L;  
  
    @Override  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
    }  
  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
    }  
  
}
```

- doGet() 用于处理get请求
- doPost() 用户处理post请求

- 在Web.xml中配置Servlet

```
<servlet>
    <servlet-name>AppServlet</servlet-name>
    <servlet-class>com.kaishengit.web.AppServlet</servlet-class>
</servlet>
```

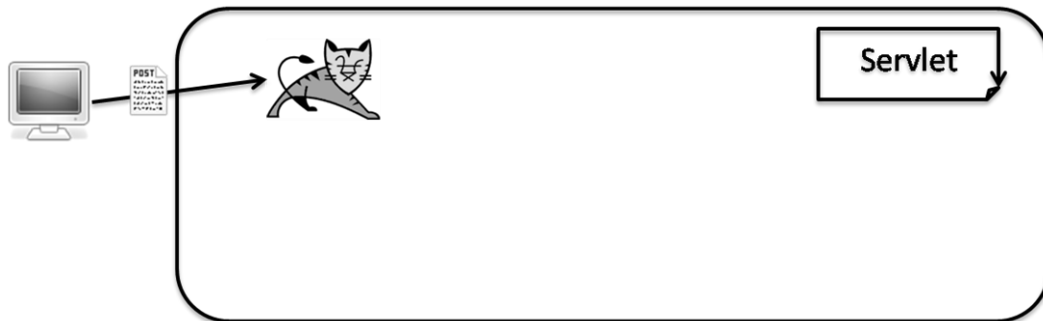
```
<servlet-mapping>
    <servlet-name>AppServlet</servlet-name>
    <url-pattern>/home</url-pattern>
</servlet-mapping>
```



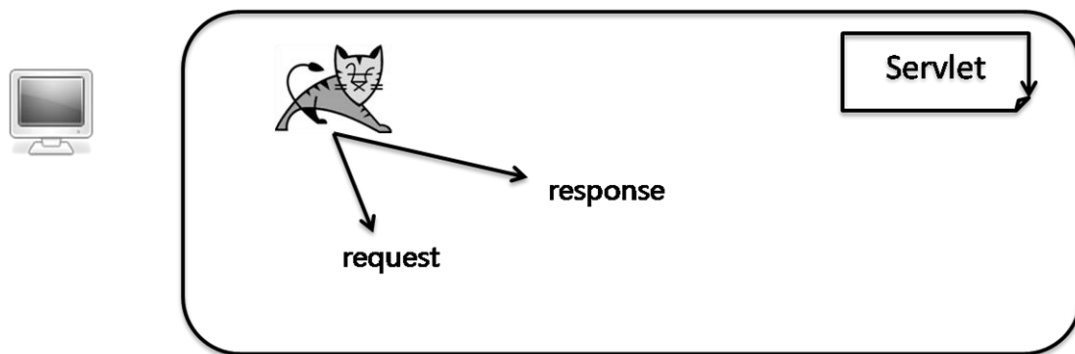
Servlet执行过程

凯盛软件

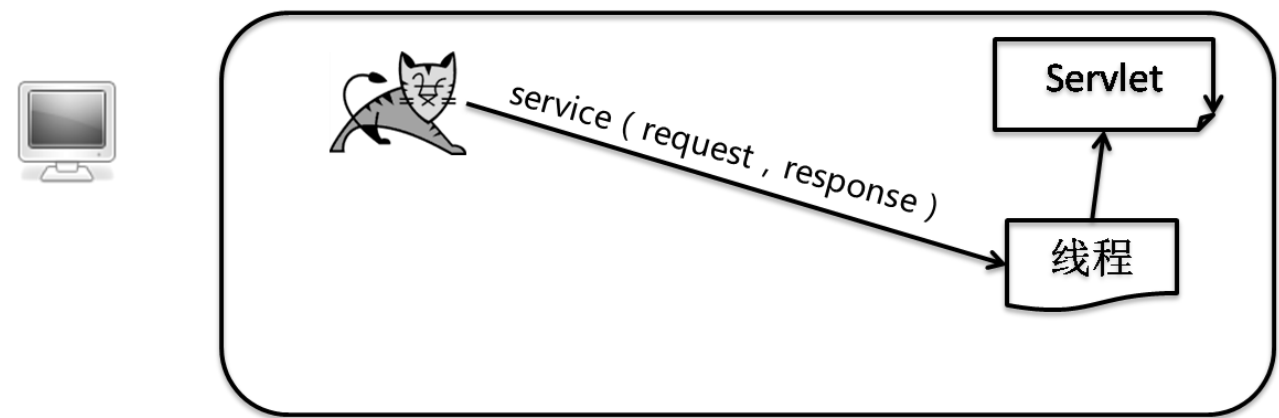
客户端发出post请求



容器产生request和response对象



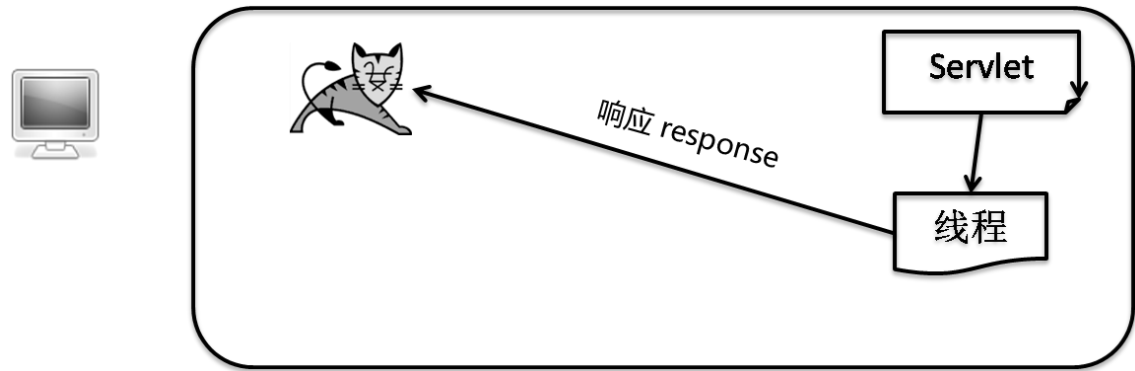
容器根据URL找到合适的Servlet并分配线程进行访问



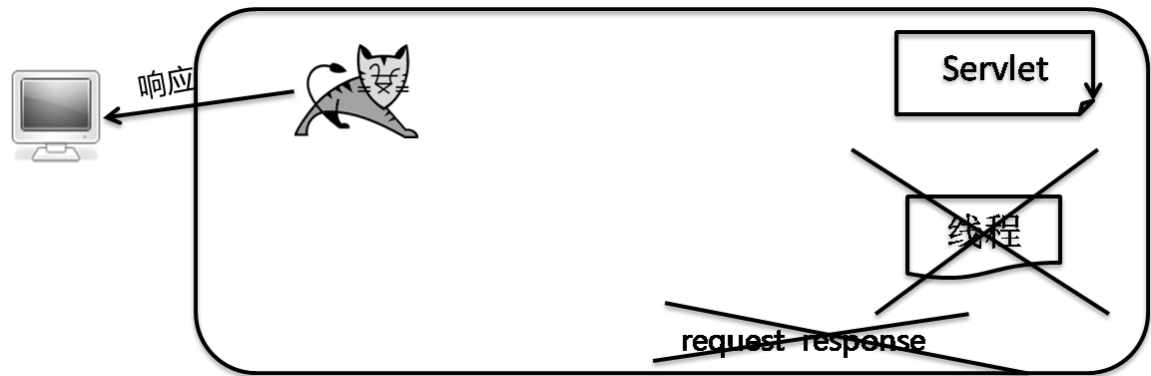
service方法根据请求头决定调用doPost方法



Servlet使用相应对象通过容器给客户端做出响应



service方法执行结束，访问线程和request、response对象被销毁



实例化

Web容器创建Servlet的实例

初始化

Web容器调用Servlet的init方法

服 务 

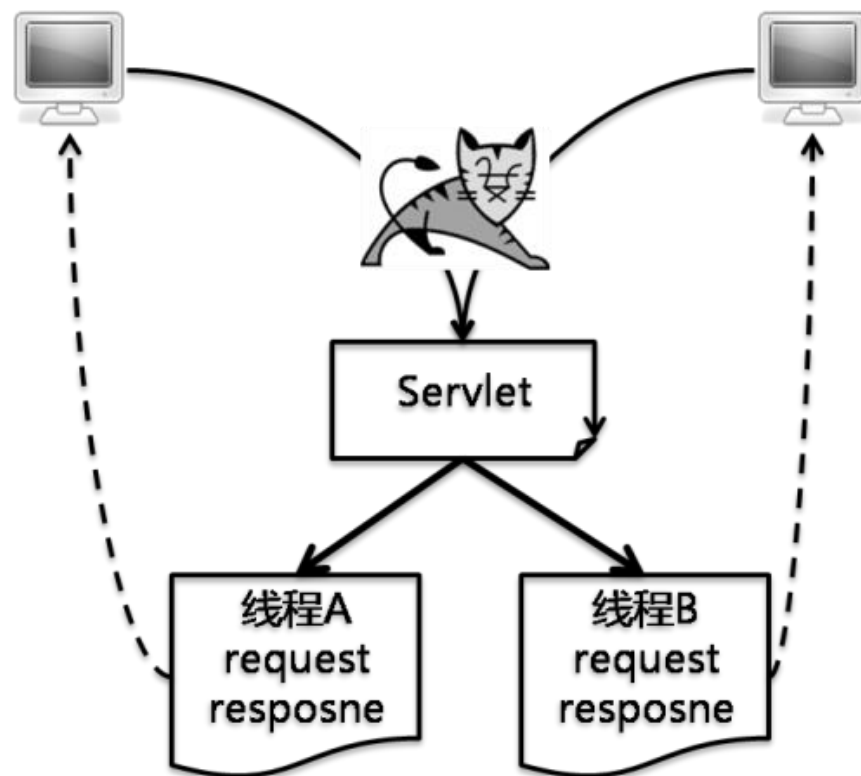
调用service方法，由service方法调用doXXX方法

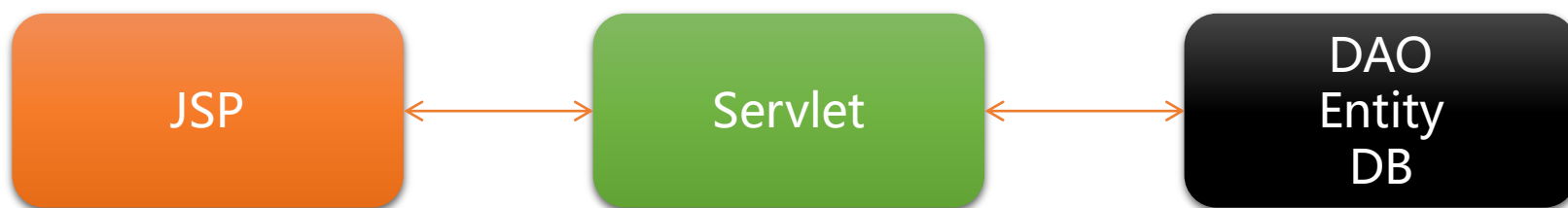
销 毁

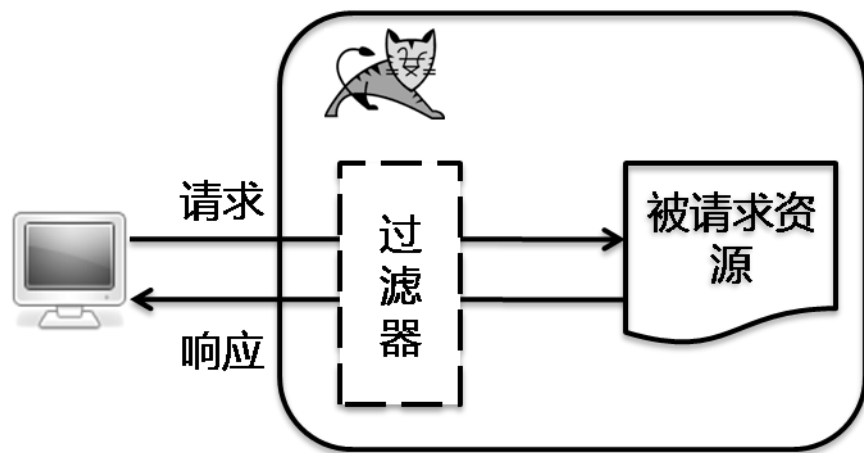
Web容器在销毁Servlet之前调用destroy方法

唯一的实例，不同的线程访问

凯盛软件

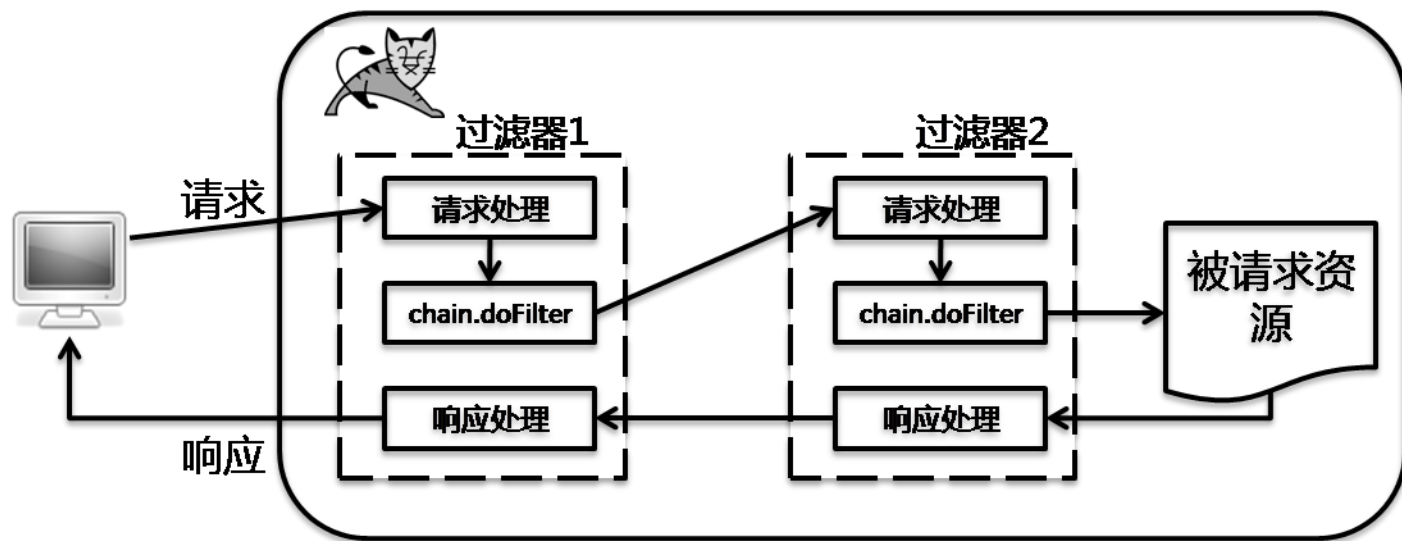






```
public class MyFilter implements Filter {  
  
    public void destroy() {  
        //销毁方法  
    }  
  
    public void doFilter(ServletRequest request, ServletResponse response,  
        FilterChain chain) throws IOException, ServletException {  
        //过滤方法  
    }  
  
    public void init(FilterConfig arg0) throws ServletException {  
        //初始化方法  
    }  
  
}
```

```
<filter>
  <filter-name>myfilter</filter-name>
  <filter-class>com.kaishengit.util.MyFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>myfilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```



- ServletContextListener
 - contextInitialized(ServletContextEvent sce)
 - contextDestroyed(ServletContextEvent sce)
- HttpSessionListener
 - sessionCreated(HttpSessionEvent se)
 - sessionDestroyed(HttpSessionEvent se)

```
<listener>  
  <listener-class>com.kaishengit.web.listener.MyListener</listener-class>  
</listener>
```


- 添加Cookie

```
Cookie ck = new Cookie("username", "tom");  
ck.setDomain("test.com");  
ck.setPath("/");  
ck.setMaxAge(60 * 60 * 24);  
response.addCookie(ck);
```

- 获取Cookie

```
Cookie[] cookies = request.getCookies();  
for(Cookie c : cookies) {  
    if("username".equals(c.getName())) {  
        System.out.println(c.getValue());  
    }  
}
```



- 删除Cookie

```
Cookie[] cookies = request.getCookies();  
for(Cookie c : cookies) {  
    if("username".equals(c.getName())) {  
        c.setMaxAge(0);  
        c.setPath("/");  
        response.addCookie(c);  
    }  
}
```

- JQuery Cookie插件：

<https://github.com/carhartl/jquery-cookie>

- Cookie安全相关

<http://zh.wikipedia.org/zh-cn/Cookie>

<http://www.infoq.com/cn/articles/cookie-security>

<http://coolshell.cn/articles/5353.html>

EL(Expression Language)表达式又称为表达式语言，是JSP中很重要的组成部分。在JSP中使用EL表达式，可以简化对变量和对象的访问。

\$ { }

<body>

12 + 15 = \${12+15}

12 * 15 = \${12*15}

12 - 15 = \${12-15}

12 / 15 = \${12/15}

12 % 15 = \${12%15}

</body>

EL表达式关系运算

凯盛软件

<body>

12==15 \${12==15}

12<15 \${12<15}

12>15 \${12>15}

12<=15 \${12<=15}

12>=15 \${12>=15}

12!=15 \${12!=15}

</body>

12==15 \${12 **eq** 15}

12<15 \${12 **lt** 15}

12>15 \${12 **gt** 15}

12<=15 \${12 **le** 15}

12>=15 \${12 **ge** 15}

12!=15 \${12 **ne** 15}

EL表达式逻辑运算

凯盛软件

`${12 < 15 && 12 < 15 }
`

`${12 < 15 and 12 < 15 }
`

`${12 < 15 || 12 > 15 }
`

`${12 < 15 or 12 >15 }
`

`${!(12 < 15) }
`

`${not (12 < 15) }`

EL表达式empty运算符

```
<body>
```

```
<%
```

```
String str = null;  
List list = new ArrayList();  
list.add("tom");  
pageContext.setAttribute("str",str);  
pageContext.setAttribute("list",list);
```

```
%>
```

```
${empty str }
```

```
${empty list}
```

```
</body>
```

```
<body>
```

```
<%
```

```
    pageContext.setAttribute("name", "tom");  
    request.setAttribute("age", "22");  
    session.setAttribute("address", "中国");  
    application.setAttribute("score", "75.5");
```

```
%>
```

```
${name}<br>
```

```
${age}<br>
```

```
${address}<br>
```

```
${score}<br>
```

```
</body>
```


EL对特定空间的访问

凯盛软件

`${pageScope.name}
`

`${requestScope.age}
`

`${sessionScope.address}
`

`${applicationScope.score}
`

```
<%  
User user = new User();  
user.setUserName("tom");  
user.setUserPwd("123");  
pageContext.setAttribute("user",user);  
%>  
{user.userName}<br>  
{user.userPwd}<br>
```

注意将get关键字去掉后的单词首字母小写

```
{user.name}<br>  
{user.address }<br>  
/body>
```

=

```
public String getName() {  
    return name;  
}
```

JSTL全称为（ JavaServer Page Standard Tag Libray Java标准标签库 ），是由SUN公司提供的简化JSP页面设计的标签。JSTL是由Core、I18N、SQL、XML、Functions五个核心标签库组成，其中最重要的是Core标签库。JSTL和EL表达式通常会一起使用。

Taglib指令

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

<c:out> 标签用于将空间内的变量进行输出。

属性名	描述
value	输出值的变量名
escapeXml	确定在结果字符串中的字符“<>‘“&”等符号应该被转换为对应的字符引用或预定义实体引用，默认值为true
default	如果value为null，则显示default中的值

条件标签包括<c:if>、<c:choose>、<c:when>和<c:otherwise>

<c:if>标签的作用和Java语言中的if语句功能是相同的

属性名	描述
test	条件语句，用于判断标签体是否可以被执行
var	将test条件语句执行的结果保存在var声明的变量中
scope	var的存储空间，默认为page

```
<c:if test="${12 > 10}" var="result" scope="page">
```

12 大于 10 test的结果为\${result }

```
</c:if>
```

```
<c:if test="${12 < 10}">
```

12 小于 10

```
</c:if>
```

<c:choose>、<c:when>和<c:otherwise>一起实现Java语言中的if/else if/else功能。

<c:choose>标签是<c:when>和<c:otherwise>标签的父标签，在<c:choose>标签中只能出现这两个子标签。

```
<c:set var="name" value="tom"></c:set>
<c:choose>
  <c:when test="${name eq 'jerry'}">
    jerry
  </c:when>
  <c:when test="${name eq 'tom'}">
    tom
  </c:when>
  <c:otherwise>
    other name
  </c:otherwise>
</c:choose>
```

<c:forEach> 标签为Core标签库中的迭代标签，主要属性有：

属性名	描述
var	存储当前迭代元素的变量名
items	被迭代的集合或数组
varStatus	迭代状态对象的变量名
begin	指定迭代开始的索引
end	指定迭代结束的索引
step	迭代的步长

对数组的迭代 :


```
<c:forEach var="str" items="${strs}" varStatus="status">
    ${status.count}
    ${str}<br/>
</c:forEach>
```

对List集合的迭代 :


```
<c:forEach var="listItem" items="${list}" begin="0" end="1">
    ${listItem}<br/>
</c:forEach>
```

对Map集合的迭代 :


```
<c:forEach var="mapItem" items="${map}">
    ${mapItem.key} : ${mapItem.value}<br/>
</c:forEach>
```

判断一个字符串是否包含了指定的字符串

- `fn:contains(String,subString)` 返回boolean
- `fn:containsIgnoreCase(String,subString)`返回boolean 忽略大小写

判断字符串是否以指定的字符串开头

- `fn:startsWith(String,prefix)` 返回boolean

判断字符串是否以指定的字符串结尾

- `fn:endsWith(String,suffix)` 返回boolean

查找指定的字符串，并返回第一个匹配的索引位置

- `fn:indexOf(String,subString)` 返回int索引值

替换字符串中指定的字符串

- `fn:replace(String,beforeSubString,afterSubString)`返回String替换后的结果

截取字符串中的某一部分

- `fn.substring(String,beginIndex,endIndex)` 返回String截取后的字符串
- `fn.substringBefore(String,subString)`返回指定字符串之前的字符串
- `fn.substringAfter(String,subString)`返回指定字符串之后的字符串

将一个字符串分拆为字符串数组

- `fn.split(String,delimiters)` 返回字符串数组

将数组中的所有元素连接为一个字符串

- `fn.join(array,separator)` 返回拼接后的字符串

将字符串转为小写字符

- `fn.toLowerCase(String)`

将字符串转为大写字符

- `fn.toUpperCase(String)`

去掉字符串前后的空格

- `fn:trim(String)`

将字符串中的字符<、>、'、"、&转换为对应的字符引用

- `fn:escapeXml(String)`

返回集合中元素的数量

- `fn:length(input)`
 - 其中input可以是数组、集合或字符串

1. 将异常转换为非强制捕获型异常
2. 将异常向上层抛出，并统一处理
3. 在错误页面给出客户友好提示，并通知管理员处理异常信息

- web.xml

```
<error-page>
```

```
  <error-code>500</error-code>
```

```
  <location>/WEB-INF/views/error/500.jsp</location>
```

```
</error-page>
```

```
<error-page>
```

```
  <exception-type>java.sql.SQLException</exception-type>
```

```
  <location>/WEB-INF/views/error/dberror.jsp</location>
```

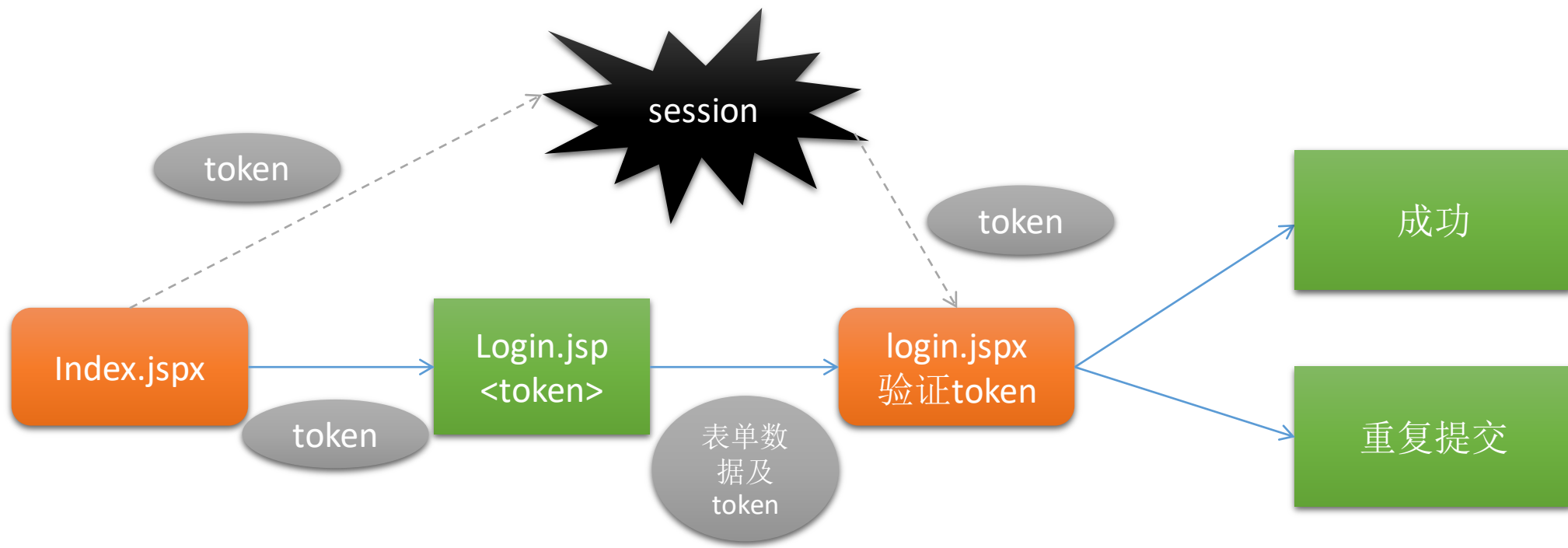
```
</error-page>
```

- `javax.servlet.error.status_code`: Integer HTTP协议的状态代码
- `javax.servlet.error.exception_type`: Class 未捕获异常的Class类的对象
- `javax.servlet.error.message`: String 传递给`sendError()`方法的消息
- `javax.servlet.error.exception`: Throwable 调用错误页面的未捕获异常
- `javax.servlet.error.request_uri`: String 当前请求的URI
- `javax.servlet.error.servlet_name`: String 导致错误页面被调用的Servlet的名字

使用Token机制处理表单重复提交

凯盛软件

1. 接收客户请求，产生Token，并放入session和request中
2. 在表单中接收Token
3. 提交表单
4. 处理表单请求前验证Token是否和Session中的一致，如果一致则不是重复提交，删除Session中Token，如果不一致则提示提交失败



//MD5加密

```
String md5 = DigestUtils.md5Hex("000000");  
System.out.println(md5);
```

//SHA-1加密

```
String sha1 = DigestUtils.shaHex("000000");  
System.out.println(sha1);
```



<http://code.google.com/p/crypto-js/>

```
<script src="http://crypto-js.googlecode.com/svn/tags/3.0.2/build/rollups/sha1.js"></script>
```

```
<script>
```

```
  var hash = CryptoJS.SHA1("Message");
```

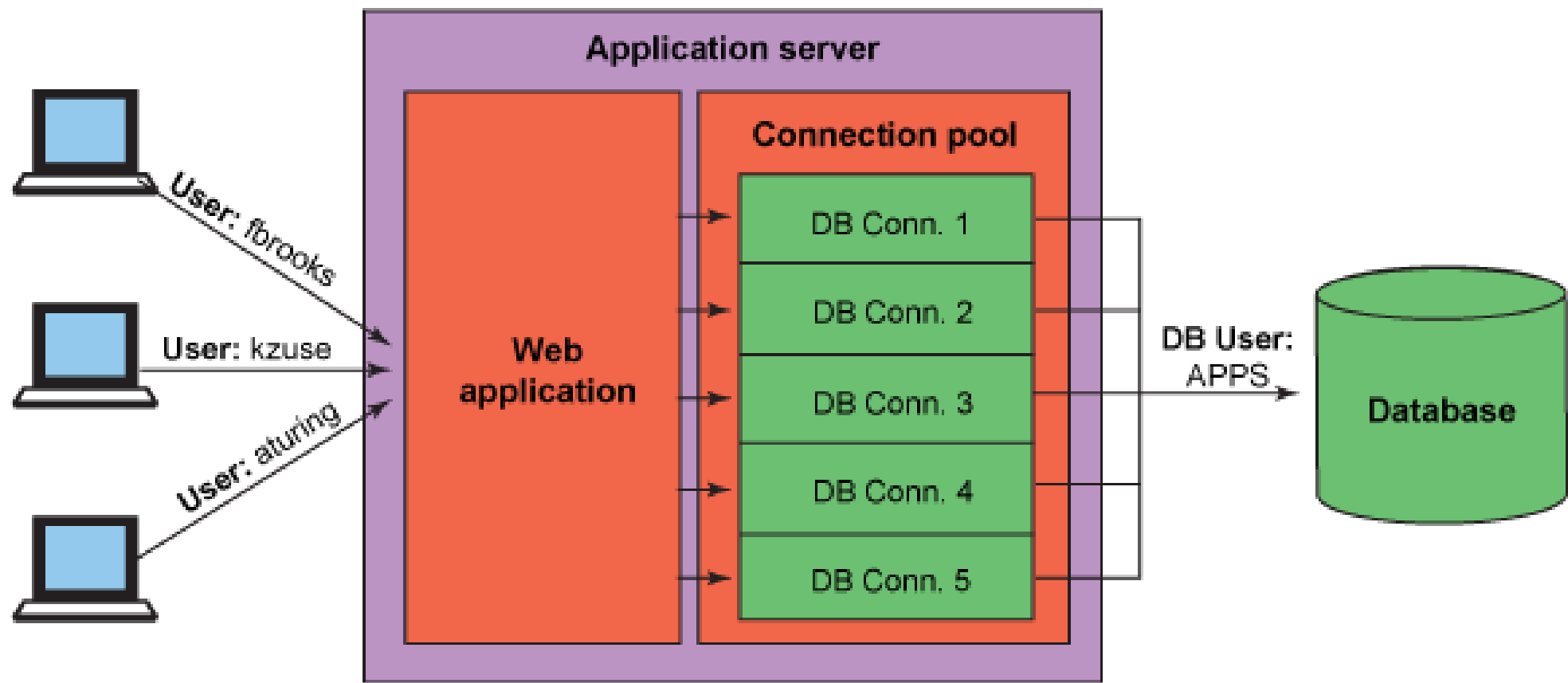
```
</script>
```

```
<script src="http://crypto-js.googlecode.com/svn/tags/3.0.2/build/rollups/md5.js"></script>
```

```
<script>
```

```
  var hash = CryptoJS.MD5("Message");
```

```
</script>
```



- DBCP <http://commons.apache.org/dbcp/>
- C3P0 <http://sourceforge.net/projects/c3p0/>

```
BasicDataSource dataSource = new BasicDataSource();
```

```
dataSource.setDriverClassName(DRIVER);
```

```
dataSource.setUrl(URL);
```

```
dataSource.setUsername(NAME);
```

```
dataSource.setPassword(PWD);
```

```
dataSource.setInitialSize(5);
```

```
dataSource.setMaxWait(5000);
```

```
dataSource.setMaxActive(20);
```

```
dataSource.setMinIdle(10);
```

```
conn = dataSource.getConnection();
```

文件上传 commons-fileupload

凯盛软件

```
<form action="file.jspx" method="post" enctype="multipart/form-data">  
  Desc:<input type="text" name="desc"/><br/>  
  <input type="file" name="myfile"/><br/>  
  <input type="submit" value="upload"/>  
</form>
```

<https://gist.github.com/4368621>

<https://gist.github.com/4368629>

