第3章 Servlet容器模型

3.1 ServletContext接口

3.2 会话管理

3.3 Cookie及其应用

3.1 ServletContext接口

3.1.1 得到ServletContext引用

3.1.2 获取应用程序的初始化参数

3.1.3 通过ServletContext对象获得资源

3.1.4 登录日志

3.1.5 使用RequestDispatcher实现请求转发

3.1.6 使用ServletContext对象存储数据

3.1.7 检索Servlet容器的信息

什么是ServletContext?

- 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javax.servlet.ServletContext实例对象，一般称为Servlet上下文对象。

- Servlet可以使用上下文对象
  - 获得Web应用程序的初始化参数
  - servlet容器的版本等信息
  - 被Servlet用来与其他的Servlet共享数据。

3.1.1 得到ServletContext引用

- 在Servlet中可以有两种方法得到ServletContext引用：

① 直接调用Servlet的getServletContext()方法：

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

② 使用getServletConfig()方法得到ServletConfig引用，再调用它的getServletContext()方法：

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

3.1.2获取应用程序的初始化参数

- ServletContext也有初始化参数
  - 可以使用ServletContext的下面两个方法检索Servlet上下文初始化参数：

public String getInitParameter(String name)

public Enumeration getInitParameterNames()

•这些初始化参数是在DD中使用<context-param>元素定义的。

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

•注意： <context-param>元素是<web-app>元素的直接子元素，并不嵌套在某个<servlet>元素中。其中定义的初始化参数是针对整个应用的。

- 要检索DD中定义的email参数值，可以使用下面代码：

```
public void init(){
  ServletContext context = getServletContext();
  String email =
   context.getInitParameter(“adminEmail");
  // 使用email
}
```

- **Servlet上下文初始化参数与Servlet的初始化参数是不同的！**

  - **Servlet上下文初始化参数是属于Web应用程序的，可以被Web应用程序的所有的Servlet和JSP页面访问。**

    <context-param>

    </context-param>

  - **Servlet初始化参数是属于定义它们的Servlet的，不能被Web应用程序的其他组件访问。**

    <init-param>

    </init-param>

3.1.3 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**。路径应该以“/”开头，是相对于该**Web**应用程序的文档根目录。

public InputStream getResourceAsStream(String path)

- 如果只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**给定的虚拟路径的真实路径**。

示例程序：**FileDownloadServlet.java**

运行程序：**http://localhost/ch03/fileDownload.do**

3.1.4 登录日志

- 使用log()方法可以将指定的消息写到服务器的日志文件中，格式为：

  public void log(String msg)

– 参数msg为写到日志文件中的消息。默认情况下将把日志信息写到Tomcat安装目录的\logs\localhost.2018-04-11.log文件中，文件名中的日期为写入日志的日期。

public void log(String msg,
Throwable throwable)

– 该方法将msg指定的消息和异常的栈跟踪信息写入日志文件。

3.1.5 使用RequestDispatcher实现请求转发

- 使用ServletContext接口的getRequestDispatcher()方法也可以获得RequestDispatcher对象，声明格式为：

```
RequestDispatcher
    getRequestDispatcher(String path)
```

  - path参数表示资源路径，它必须以“/”开头，表示相对于应用程序的文档根目录。

```
RequestDispatcher
getNamedDispatcher(String name)
```

  - 参数name为一个命名的Servlet对象

- **ServletContext与ServletRequest的getRequestDispatcher()方法的区别：**
  - **ServletRequest的getRequestDispatcher()方法，可以传递一个相对路径，**
  - **ServletContext的getRequestDispatcher()方法只能传递以“/”开头的路径。**

  例如，

  request.getRequestDispatcher(“../html/copyright.html”)是合法的，该方法相对于请求的路径计算路径。

3.1.6 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，如下所示：

```
public void setAttribute(String name, Object object)
public Object getAttribute(String name)
public Enumeration getAttributeNames()
public void removeAttribute(String  name)
```

3.1.7 检索Servlet容器的信息

public String getServerInfo()

- 返回容器的名称和版本。

public int getMajorVersion()

- 返回容器所支持的Servlet API的主版本号。

public int getMinorVersion()

- 返回容器所支持的Servlet API的次版本号。

public String getServletContextName()

- 返回与该ServletContext对应的Web应用程序名，它是在web.xml中使用<display-name>元素定义的名称。

3.2会话管理

3.2.1 理解状态与会话

3.2.2 会话管理机制

3.2.3 HttpSession API

3.2.4 使用HttpSession对象

3.2.5 会话超时与结束

3.2.1 理解状态与会话

1、HTTP协议的无状态特性

- 协议记住用户及其请求的能力称为状态(state)。
- 协议分成两种类型：

  1)有状态的　2)无状态的

- HTTP协议是一种无状态的协议

  对服务器的每个请求和相应的响应都是作为一个分离的事务处理。

3.2.1 理解状态与会话

2、HTTP协议连接过程

- 客户发出请求，服务器找到资源，向客户发送响应，然后断开连接。
- 客户再次请求，服务器不能记住客户。

- 无状态的应用：信息检索应用，图书目录查询等。
- 有状态的需求：购物网站：需要记住客户。

3、服务器如何区分不同的客户？

- 使用会话实现无状态的HTTP页面变成有状态的Web应用。

- 会话:是客户与服务器之间的不中断的请求响应序列。

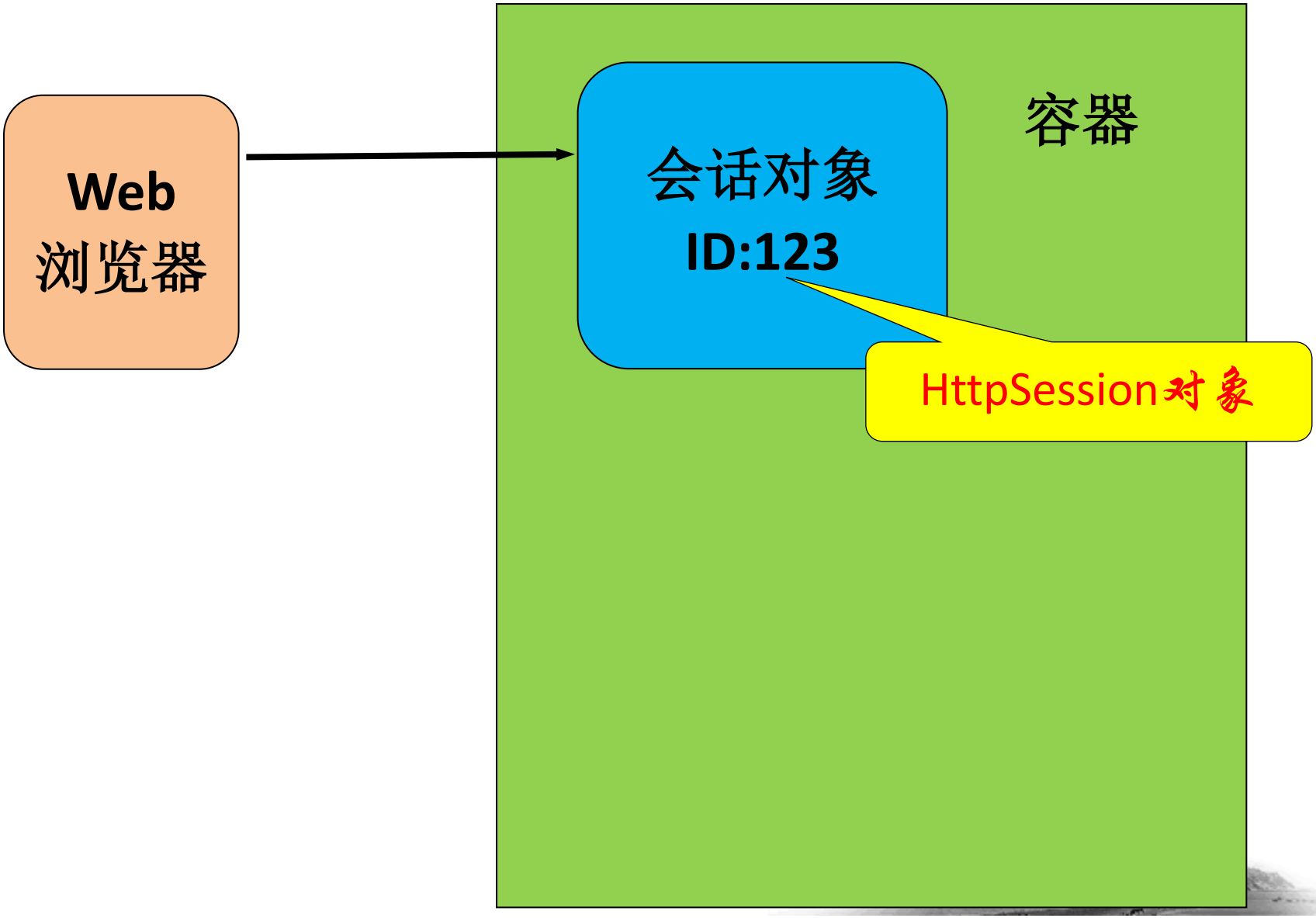
- 对会话的每个请求，服务器能够识别出请求来自于同一个客户。
- 当一个未知的客户向web应用程序发送第一个请求时就开始了一个会话。

3.2.2 会话管理机制

- 容器通过使用**HttpSession**对象管理会话。

1）当客户向服务器发送第一个请求时，服务器就可以为该客户创建一个**HttpSession**会话对象，并将请求对象与该会话对象关联。服务器在创建会话对象时为其指定一个唯一标识符，称为**会话ID**，它可作为该客户的唯一标识。

3.2.2 会话管理机制



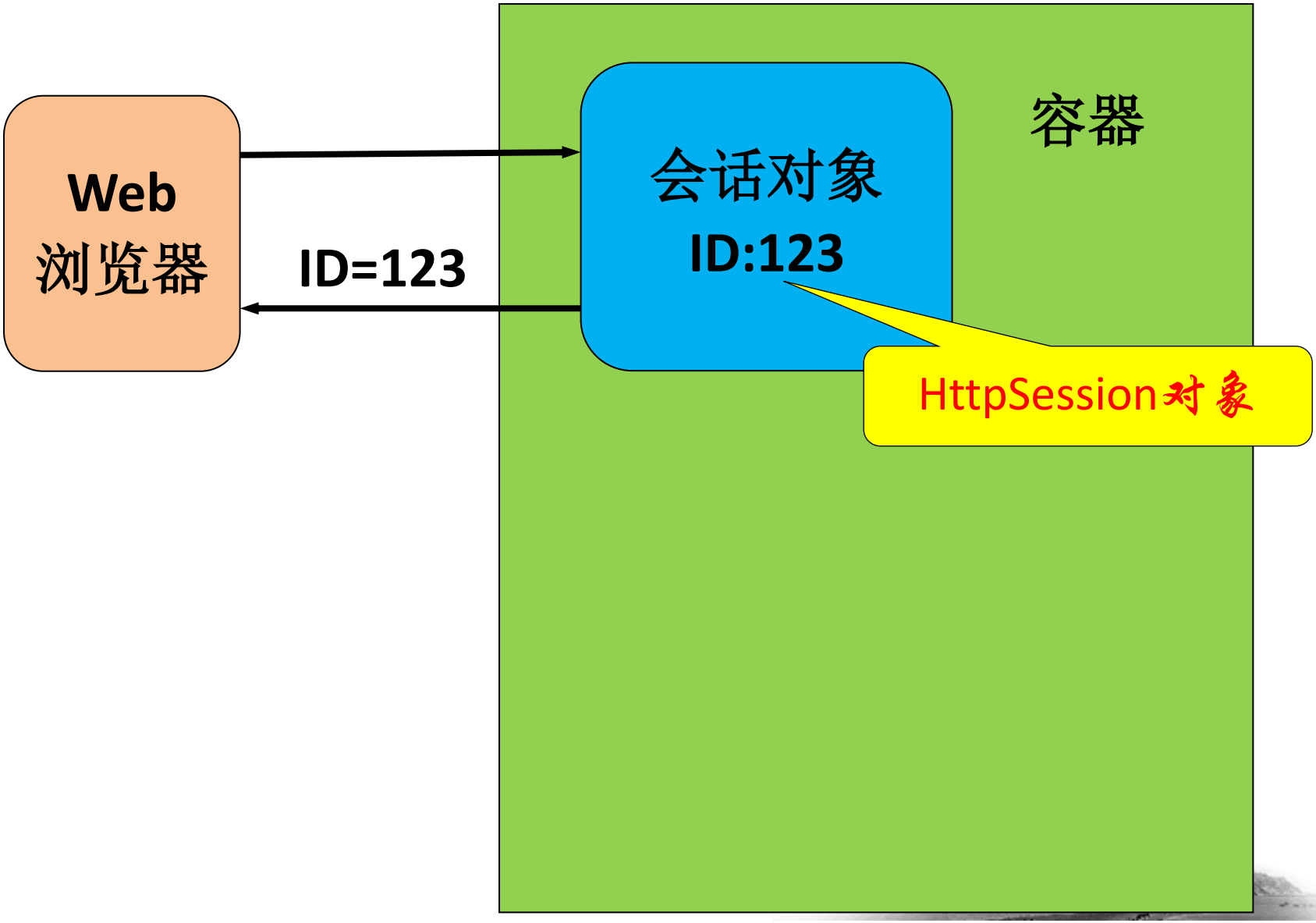
3.2.2 会话管理机制

2）当服务器向客户发送响应时，服务器将该会话ID与响应数据一起发送给客户，这是通过Set-Cookie响应头实现的，响应消息可能为：

```
HTTP/1.1 200 OK
Set-Cookie:JSESSIONID=61C4F2352452
                        1390E70993E5120263C6
Content-Type:text/html
…
```

- 这里，JSESSIONID的值即为会话ID，它是32位的十六进制数。

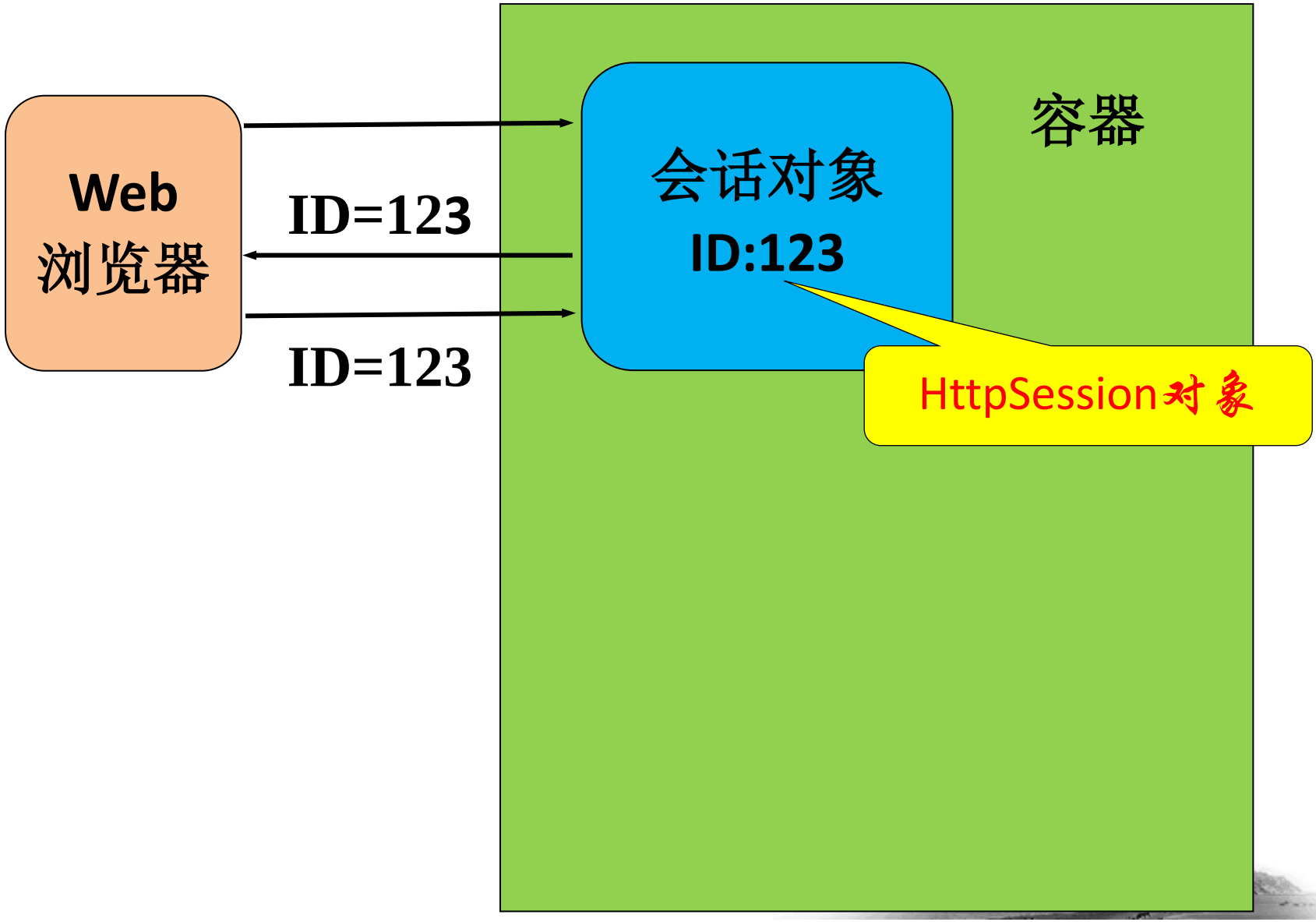
3.2.2 会话管理机制



3）客户在接收到响应后将会话ID存储在浏览器的内存中。当客户再次向服务器发送一个请求时，它将通过Cookie请求头把会话ID与请求一起发送给服务器。这时请求消息可能为：

```
POST /bookstore/selectBook.do HTTP/1.1
Host:www.mydomain.com
Cookie: JSESSIONID=
 61C4F23524521390E70993E5120263C6
  …
```

3.2.2 会话管理机制

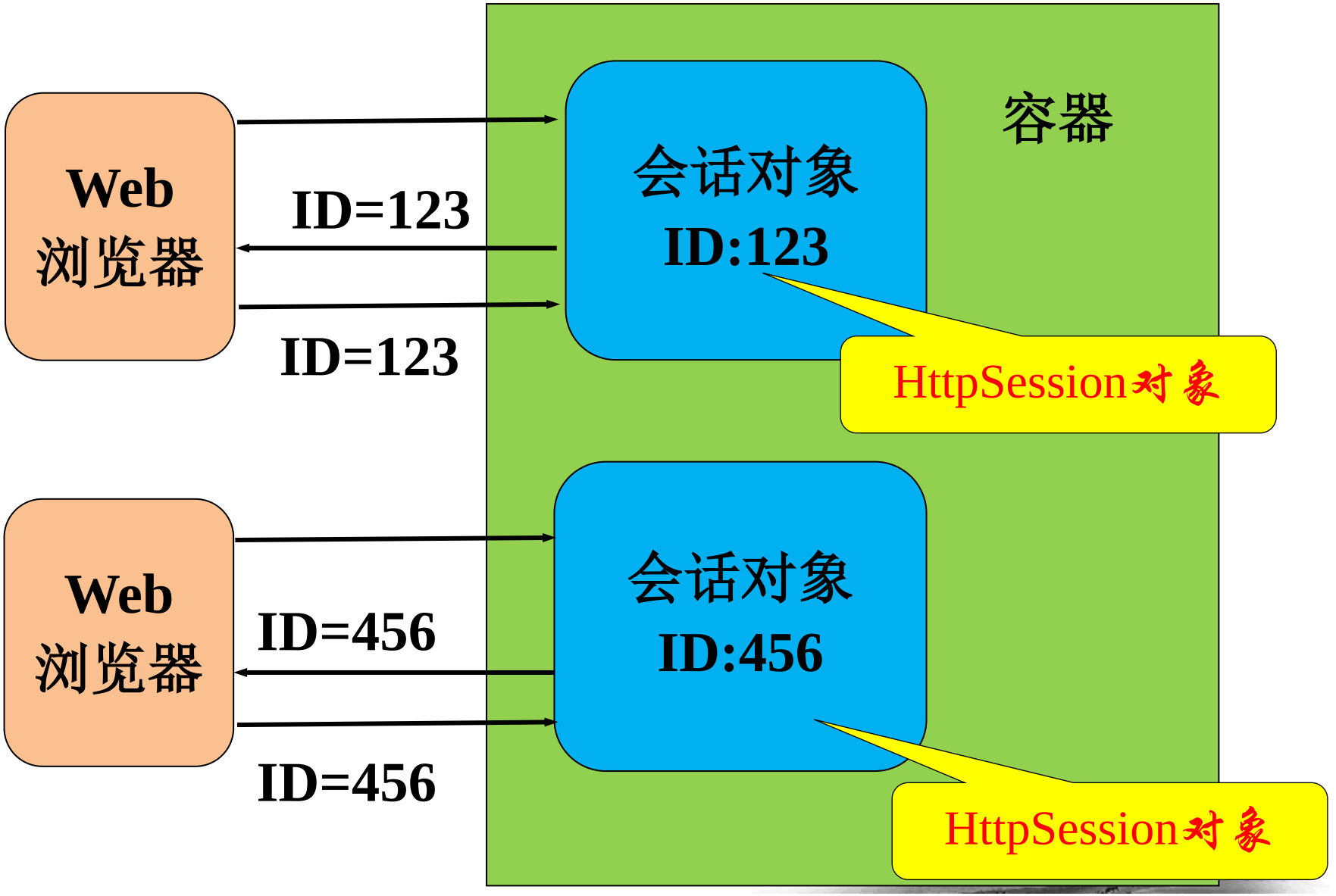


4）服务器接收到请求并看到会话ID,它将查找之前创建的会话对象,并将该请求与会话ID相同的会话对象关联起来,此时客户被称为加入了会话。

上述过程的第 2）到第 4）步一直保持重复。

如果客户在指定时间没有发送任何请求，服务器将使会话对象失效。一旦会话对象失效，即使客户再发送同一个会话ID，会话对象也不能恢复。

3.2.2 会话管理机制


容器
Web
浏览器
ID=123
ID=123
会话对象
ID:123
HttpSession对象
Web
浏览器
ID=456
ID=456
会话对象
ID:456
HttpSession对象

- **注意：**容器不能使用客户的**IP地址**唯一标识客户。
- 客户可能是通过局域网访问**Internet**。尽管在局域网中客户有一个**IP**地址，但对于服务器来说，客户的**IP**地址实际是路由器的**IP**地址，所以该局域网的所有客户的**IP**地址都相同！因此也就无法唯一标识客户。

3.2.3 HttpSession API

- **public String getId()**

  返回为该会话指定的唯一标识符。

- **public long getCreationTime()**

  返回会话创建时间

- **public long getLastAccessedTime()**

  返回会话最后被访问时间

- **public boolean isNew()**

  如果会话还没有同客户关联，则返回**true**

- **public ServletContext getServletContext()**

  返回该会话所属的 **ServletContext**对象

- **HttpSession**对象除了表示会话外，还用来存储与会话相关的数据信息。
- 如，电子 商务网站的购物车就应该存储在会话对象上。

- **public void setAttribute (String name, Object value)**
- **public Object getAttribute(String name)**
- **public Enumeration getAttributeNames()**
- **public void removeAttribute(String name)**

3.2.4使用HttpSession对象

- 使用HttpSession对象通常需要三步：

1）为客户创建或获得与请求关联的会话对象；

2）在会话对象中添加或删除名/值对属性；

3）如果需要可使会话失效。

创建或获得会话对象

◆使用**HttpServletRequest**接口提供的**getSession()**方法。有两种格式：

public HttpSession getSession(boolean create)

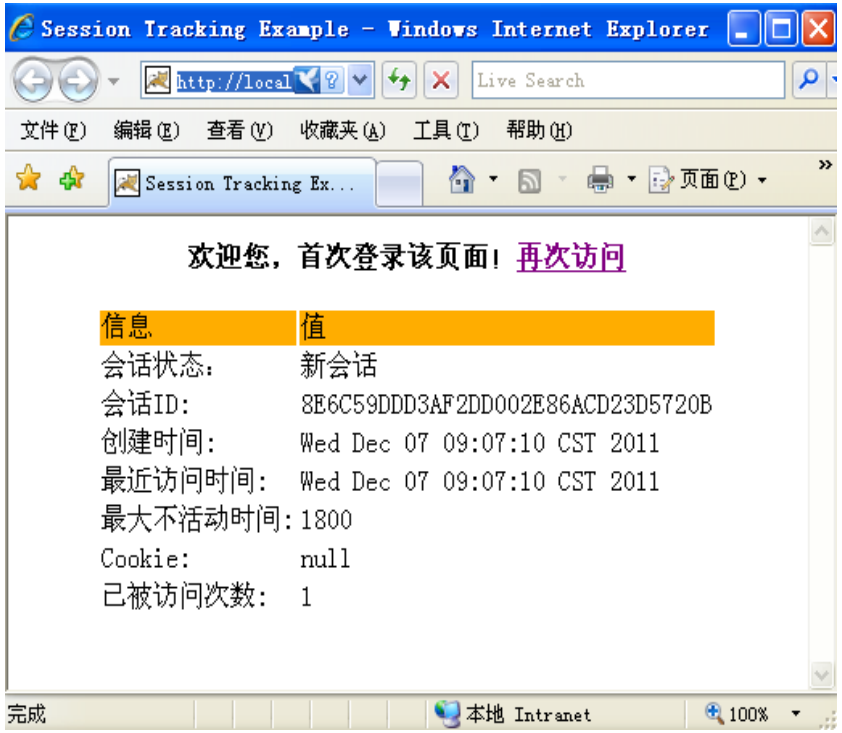
如果没有与当前请求关联的会话对象，当参数为**true**时**创建**一个新的会话对象，当参数为**false**时返回**null**。

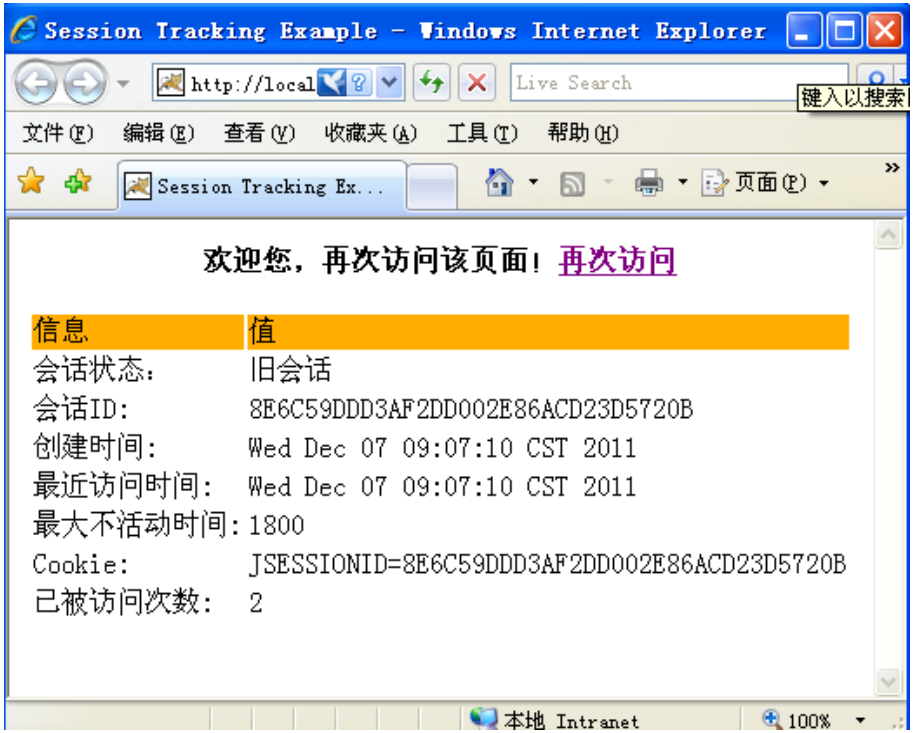
public HttpSession getSession()

程序3.2： ShowSessionServlet.java

运行程序：

http://localhost/ch03/ShowSessionServlet





3.2.5 会话超时与失效

- 规定当用户在一个指定的期限内处于不活动状态时，就将用户的会话终止，这称为会话超时（session timeout）。

1. 可以在DD文件中设置会话超时时间，例如：

```
<web-app>
…
  <session-config>
    <session-timeout>10</session-timeout>
  </session-config>
…
<web-app>
```

单位是分钟。0或小于0的值表示会话永不过期。默认值是30分钟

2. 使用HttpSession接口提供的方法，设置在容器使该会话失效前客户的两个请求之间最大间隔的秒数，参数值为负表示永不失效。格式为：

public void setMaxInactiveInterval (int interval)

public int getMaxInactiveInterval()

- 需要注意，方法仅对调用它的会话有影响，其他会话的超时期限仍然是DD文件中设置的值。

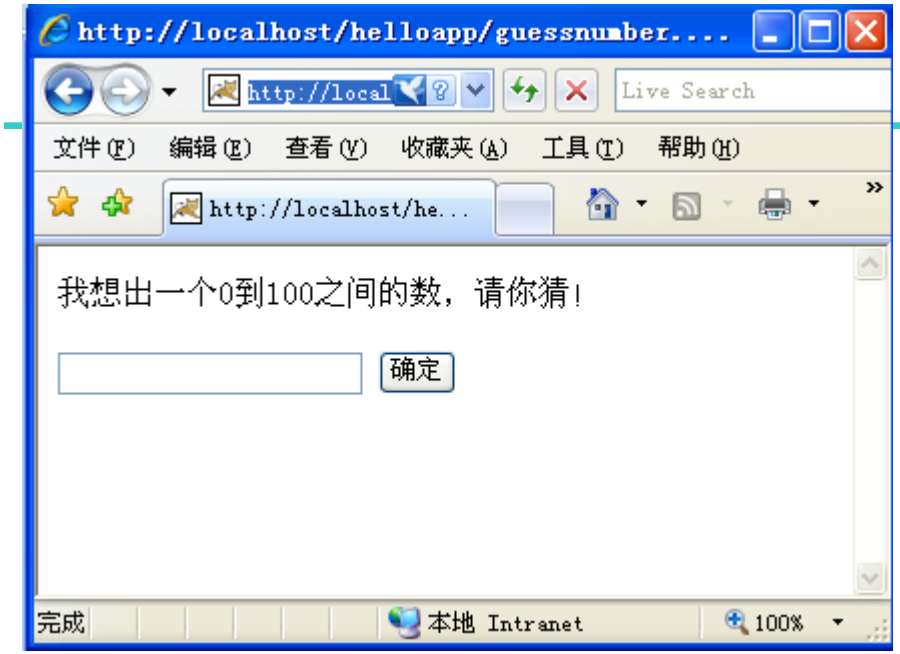
3. HttpSession接口提供了使一个会话失效的invalidate()方法，使会话失效并解除绑定到其上的任何对象。格式如下：

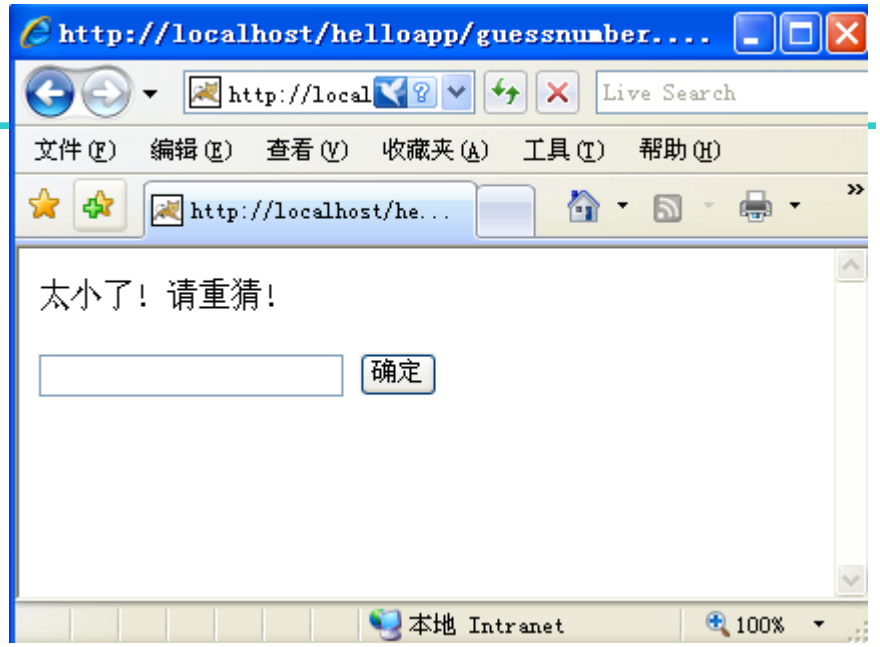
```
public void invalidate()
```

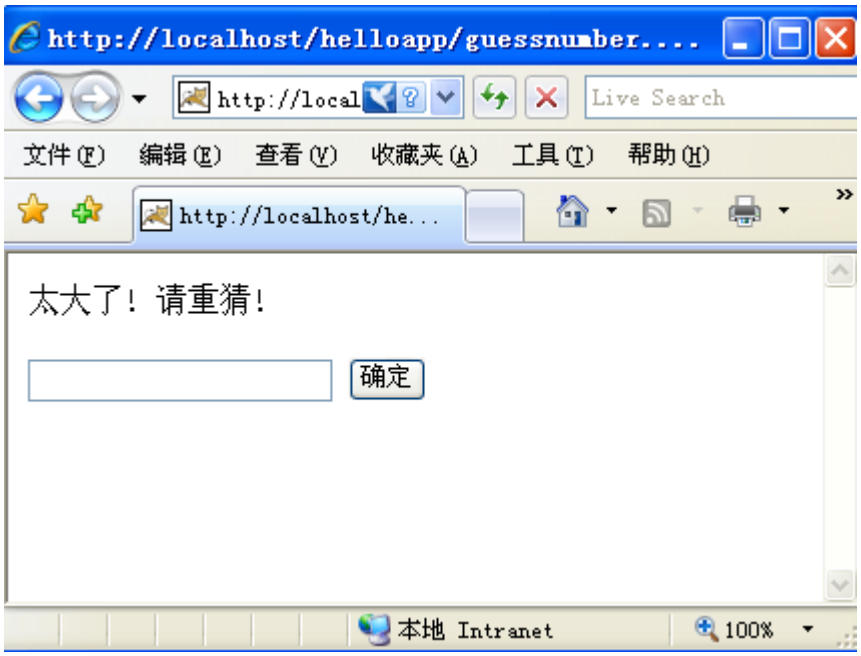
例如，在购物车的例子中，我们希望在付款处理完成后结束会话，这样，当用户再发送请求时，就会创建一个购物车中不包含商品的新的会话。

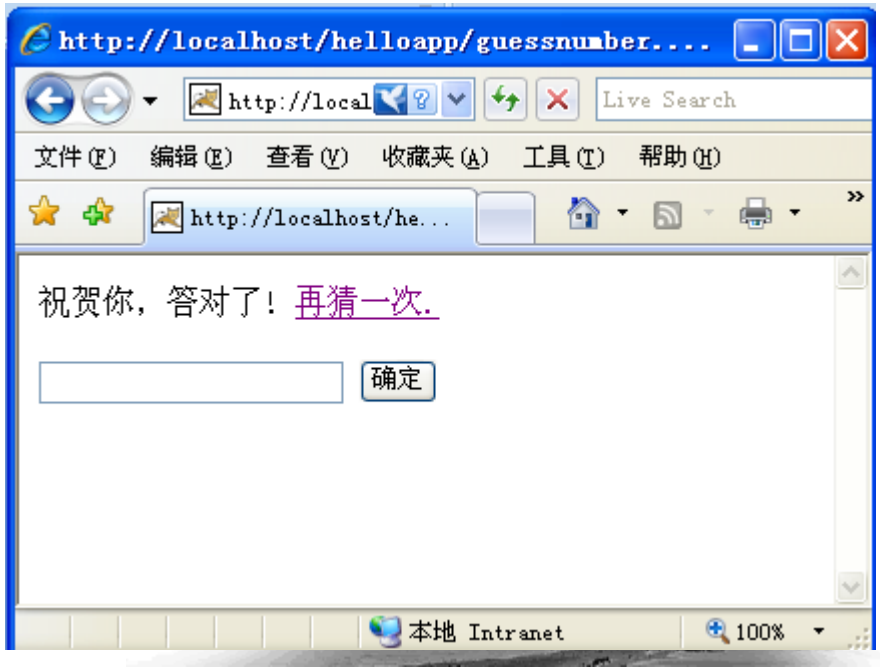
- 示例:一个猜数游戏的Servlet:
- 程序3.3 GuessNumberServlet.java
- 运行:

  http://localhost/chap03/GuessNumberServlet


http://localhost/helloapp/guessnumber....
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
http://localhost/he...
我想出一个0到100之间的数，请你猜！
确定
完成
本地 Intranet
100%


http://localhost/helloapp/guessnumber....
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
http://localhost/he...
太小了！请重猜！
确定
本地 Intranet
100%


http://localhost/helloapp/guessnumber....
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
http://localhost/he...
太大了！请重猜！
确定
本地 Intranet
100%


http://localhost/helloapp/guessnumber....
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
http://localhost/he...
祝贺你，答对了！再猜一次.
确定
本地 Intranet
100%

3.3 Cookie及其应用

3.3.1 Cookie API

- Cookie是客户访问Web服务器时，服务器在客户硬盘上存放的信息，好像是服务器送给客户的“点心”。

- Cookie实际上是一小段文本信息，客户以后访问同一个Web服务器时浏览器会把它们原样发送给服务器。

- 通过让服务器读取它原先保存到客户端的信息，网站能够为浏览者提供一系列的方便。例如，在线交易过程中标识用户身份、安全要求不高的场合避免客户登录时重复输入用户名和密码等等。

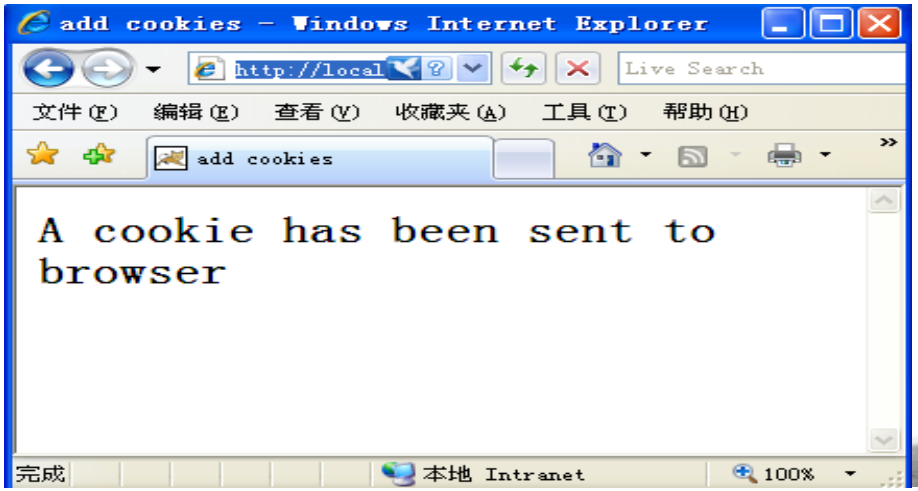
- 对Cookie的管理需要使用javax.servlet.http.Cookie类，
  构造方法：
  public Cookie(String name, String value)
- Cookie类的常用方法如下：
  public String getName()
  public String getValue()
  public void setValue(String newValue)
  public void setMaxAge(int expiry)
  设置Cookie在浏览器中的最长存活时间，单位为秒。如果参数值为负，表示并不永久存储Cookie，如果是0表示删除该Cookie。
  public int getMaxAge()
  public void setDomain(String pattern)
  public String getDomain()

3.3.2向客户端发送Cookie

```
Cookie userCookie = new Cookie("username",
"hacker");
userCookie. setMaxAge(60*60*24*7);
response.addCookie(userCookie);
```

程序3.4 SendCookieServlet.java

运行:http://localhost/ch03/SendCookie

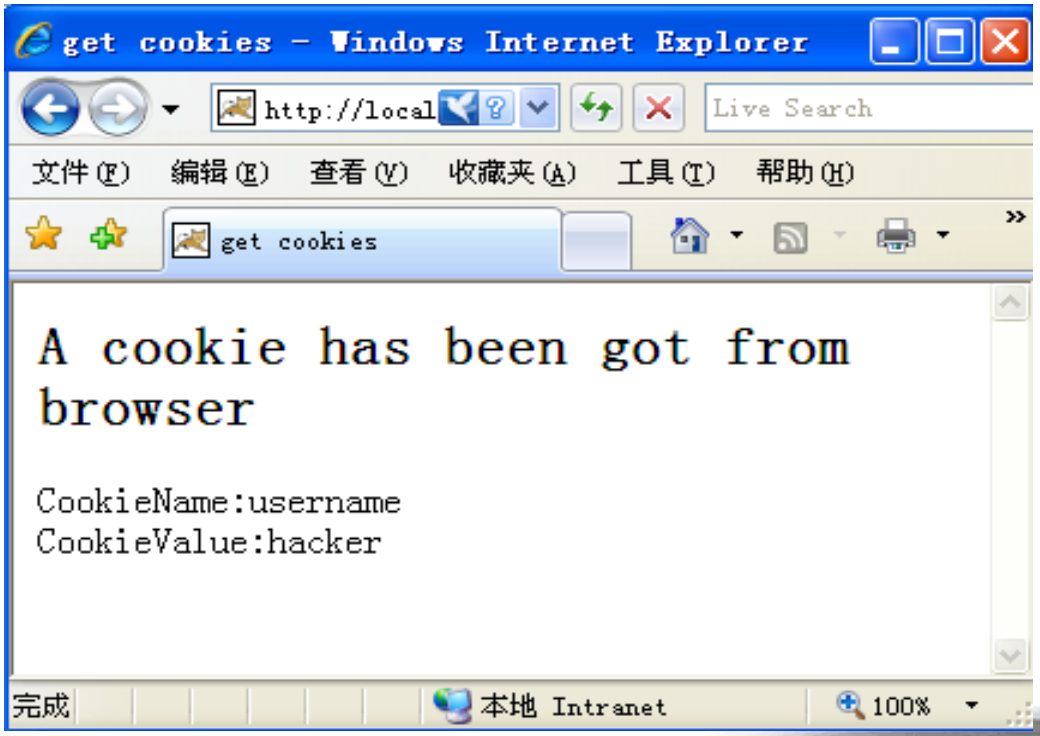


3.3.3从客户端读取Cookie

Cookie[] cookies=request.getCookies();

程序3.5 **ReadCookieServlet.java**

运行: **http://localhost/ch03/ReadCookie**



3.3.4 Cookie的安全问题

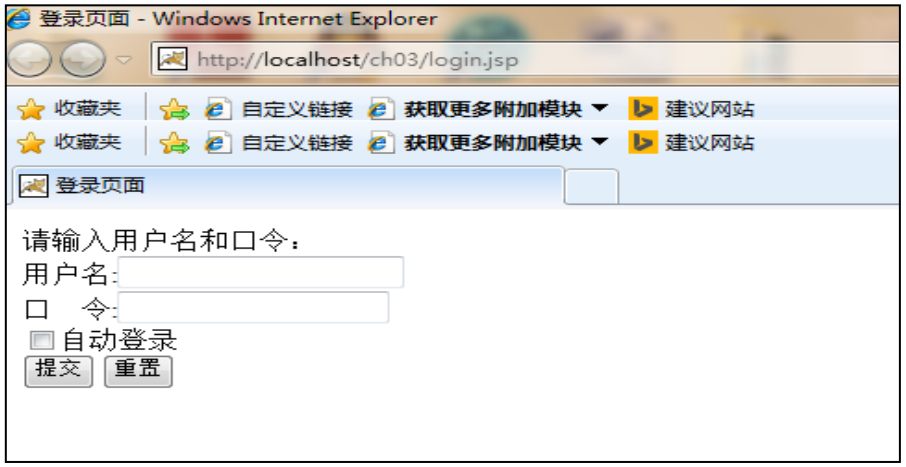
- 客户可能认为Cookie会带来安全问题，因此禁用Cookie。
- 事实上，Cookie并不会造成严重的安全威胁。Cookie永远不会以任何方式执行，因此也不会带来病毒或攻击你的系统。
- 另外，由于浏览器一般只允许存放300个Cookie，每个站点的Cookie最多存放20个，每个Cookie的大小限制为4 KB，因此Cookie不会塞满你的硬盘，更不会被用作“拒绝服务”攻击手段。
- 在IE浏览器中可以设置是否使用Cookie

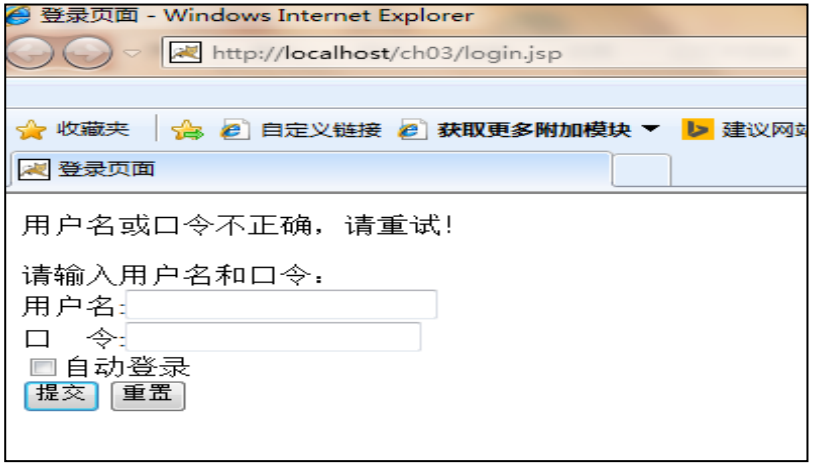
3.3.5实例：用Cookie实现自动登录

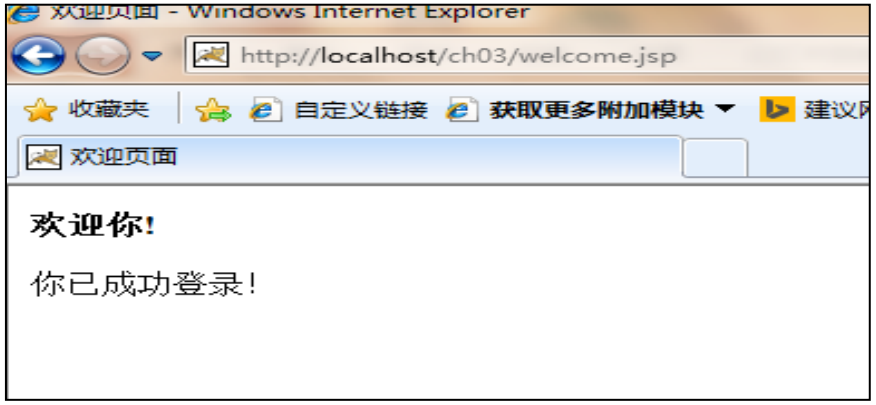
- 下面的Servlet可以通过Cookie实现自动登录的功能。
- 程序3.6 login.jsp
- 程序3.7 CheckUserServlet.java

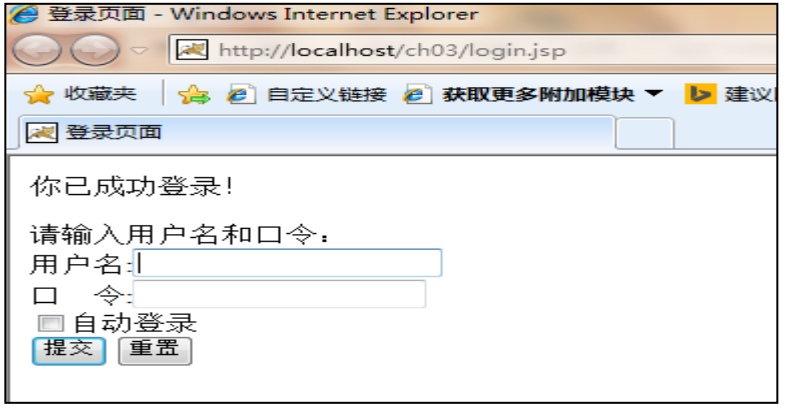
运行 http://localhost/ch03/login.jsp

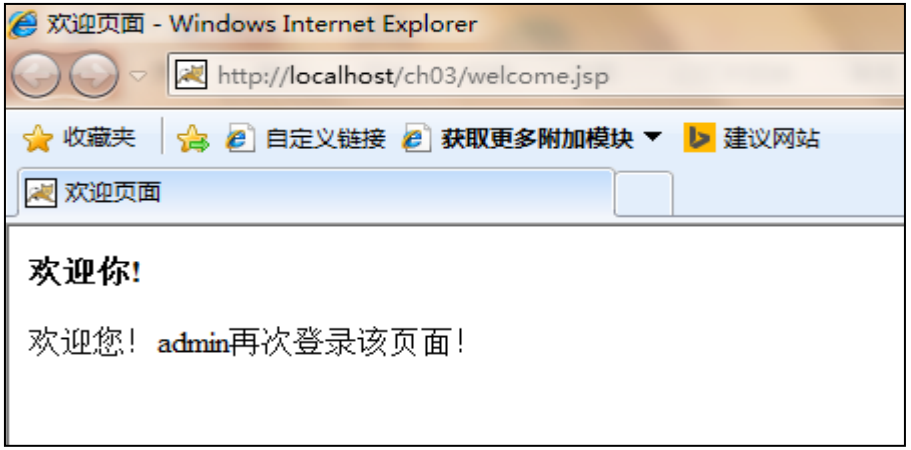
http://localhost/ch03/CheckUserServlet


登录页面 - Windows Internet Explorer
http://localhost/ch03/login.jsp
收藏夹
自定义链接
获取更多附加模块
建议网站
登录页面
请输入用户名和口令：
用户名:
口　令:
自动登录
提交
重置


登录页面 - Windows Internet Explorer
http://localhost/ch03/login.jsp
收藏夹
自定义链接
获取更多附加模块
登录页面
用户名或口令不正确，请重试!
请输入用户名和口令：
用户名:
口　令:
自动登录
提交
重置


欢迎页面 - Windows Internet Explorer
http://localhost/ch03/welcome.jsp
收藏夹
自定义链接
获取更多附加模块
欢迎页面
欢迎你!
你已成功登录!


登录页面 - Windows Internet Explorer
http://localhost/ch03/login.jsp
收藏夹
自定义链接
获取更多附加模块
登录页面
你已成功登录!
请输入用户名和口令：
用户名:
口　令:
自动登录
提交
重置


欢迎页面 - Windows Internet Explorer
http://localhost/ch03/welcome.jsp
收藏夹
自定义链接
获取更多附加模块
建议网站
欢迎页面
欢迎你!
欢迎您！ admin再次登录该页面!

3.4 小 结

- 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javax.servlet.ServletContext实例对象，一般称为Servlet上下文对象。
- 容器通过使用HttpSession对象管理会话
- Cookie是客户访问Web服务器时，服务器在客户硬盘上存放的信息。对Cookie的管理需要使用javax.servlet.http.Cookie类。