今日内容

1. 会话技术
    1. Cookie
    2. Session
2. JSP：入门学习

会话技术

1. 会话：一次会话中包含多次请求和响应。
    * 一次会话：浏览器第一次给服务器资源发送请求，会话建立，直到有一方断开为止
2. 功能：在一次会话的范围内的多次请求间，共享数据
3. 方式：
    1. 客户端会话技术：Cookie
    2. 服务器端会话技术：Session

Cookie：

1. 概念：客户端会话技术，将数据保存到客户端

2. 快速入门：
    * 使用步骤：
        1. 创建Cookie对象，绑定数据
            * new Cookie(String name, String value)
        2. 发送Cookie对象
            * response.addCookie(Cookie cookie)
        3. 获取Cookie，拿到数据
            * Cookie[] request.getCookies()

3. 实现原理
    * 基于响应头set-cookie和请求头cookie实现

4. cookie的细节
    1. 一次可不可以发送多个cookie?
        * 可以
        * 可以创建多个Cookie对象，使用response调用多次addCookie方法发送cookie即可。
    2. cookie在浏览器中保存多长时间？
        1. 默认情况下，当浏览器关闭后，Cookie数据被销毁

2．持久化存储：

* setMaxAge(int seconds)

1．正数：将Cookie数据写到硬盘的文件中。持久化存储。并指定cookie存活时间，时间到后，cookie文件自动失效

2．负数：默认值

3．零：删除cookie信息

3．cookie能不能存中文？

* 在tomcat 8 之前 cookie中不能直接存储中文数据。

* 需要将中文数据转码---一般采用URL编码(%E3)

* 在tomcat 8 之后，cookie支持中文数据。特殊字符还是不支持，建议使用URL编码存储，URL解码解析

4．cookie共享问题？

1．假设在一个tomcat服务器中，部署了多个web项目，那么在这些web项目中cookie能不能共享？

* 默认情况下cookie不能共享

* setPath(String path):设置cookie的获取范围。默认情况下，设置当前的虚拟目录

* 如果要共享，则可以将path设置为"/"

2．不同的tomcat服务器间cookie共享问题？

* setDomain(String path):如果设置一级域名相同，那么多个服务器之间cookie可以共享

* setDomain(".baidu.com"),那么tieba.baidu.com和news.baidu.com中cookie可以共享

5．Cookie的特点和作用

1．cookie存储数据在客户端浏览器

2．浏览器对于单个cookie 的大小有限制(4kb) 以及 对同一个域名下的总cookie数量也有限制(20个)

* 作用：

1．cookie一般用于存出少量的不太敏感的数据

2．在不登录的情况下，完成服务器对客户端的身份识别

6．案例：记住上一次访问时间

1．需求：

1．访问一个Servlet，如果是第一次访问，则提示：您好，欢迎您首次访问。

2．如果不是第一次访问，则提示：欢迎回来，您上次访问时间为:显示时间字符串

2. 分析：
    1. 可以采用Cookie来完成
    2. 在服务器中的Servlet判断是否有一个名为lastTime的cookie
        1. 有：不是第一次访问
            1. 响应数据：欢迎回来，您上次访问时间为:2018年6月10日11:50:20
            2. 写回Cookie：lastTime=2018年6月10日11:50:01
        2. 没有：是第一次访问
            1. 响应数据：您好，欢迎您首次访问
            2. 写回Cookie：lastTime=2018年6月10日11:50:01

3. 代码实现：

```
package cn.itcast.cookie;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.util.Date;


@WebServlet("/cookieTest")
public class CookieTest extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        //设置响应的消息体的数据格式以及编码
        response.setContentType("text/html;charset=utf-8");

        //1.获取所有Cookie
        Cookie[] cookies = request.getCookies();
        boolean flag = false;//没有cookie为lastTime
        //2.遍历cookie数组
        if(cookies != null && cookies.length > 0){
            for (Cookie cookie : cookies) {
                //3.获取cookie的名称
                String name = cookie.getName();
```

```
                    //4.判断名称是否是：lastTime
                    if("lastTime".equals(name)){
                        //有该Cookie，不是第一次访问

                        flag = true;//有lastTime的cookie

                        //设置Cookie的value
                        //获取当前时间的字符串，重新设置Cookie的值，重新发送cookie
                        Date date  = new Date();
                        SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss");
                        String str_date = sdf.format(date);
                        System.out.println("编码前："+str_date);
                        //URL编码
                        str_date = URLEncoder.encode(str_date,"utf-8");
                        System.out.println("编码后："+str_date);
                        cookie.setValue(str_date);
                        //设置cookie的存活时间
                        cookie.setMaxAge(60 * 60 * 24 * 30);//一个月
                        response.addCookie(cookie);


                        //响应数据
                        //获取Cookie的value，时间
                        String value = cookie.getValue();
                        System.out.println("解码前："+value);
                        //URL解码：
                        value = URLDecoder.decode(value,"utf-8");
                        System.out.println("解码后："+value);
                        response.getWriter().write("<h1>欢迎回来，您上次访问时间为:"+value+"</h1>");

                        break;

                    }
                }
            }


            if(cookies == null || cookies.length == 0 || flag == false){
```

```
                //没有，第一次访问

                //设置Cookie的value
                //获取当前时间的字符串，重新设置Cookie的值，重新发送
cookie
                Date date  = new Date();
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM
月dd日 HH:mm:ss");
                String str_date = sdf.format(date);
                System.out.println("编码前："+str_date);
                //URL编码
                str_date = URLEncoder.encode(str_date,"utf-8");
                System.out.println("编码后："+str_date);

                Cookie cookie = new Cookie("lastTime",str_date);
                //设置cookie的存活时间
                cookie.setMaxAge(60 * 60 * 24 * 30);//一个月
                response.addCookie(cookie);

                response.getWriter().write("<h1>您好，欢迎您首次访问
</h1>");
            }


        }

        protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
            this.doPost(request, response);
        }
    }
```

JSP：入门学习

1. 概念：
    * Java Server Pages： java服务器端页面
        * 可以理解为：一个特殊的页面，其中既可以指定定义html标签，又可以定义java代码
        * 用于简化书写！！！

2. 原理
    * JSP本质上就是一个Servlet

3. JSP的脚本：JSP定义Java代码的方式
    1. <% 代码 %>：定义的java代码，在service方法中。service方法中可以定义什么，该脚本中就可以定义什么。
    2. <%! 代码 %>：定义的java代码，在jsp转换后的java类的成员位置。
    3. <%= 代码 %>：定义的java代码，会输出到页面上。输出语句中可以定义什么，该脚本中就可以定义什么。

4. JSP的内置对象：
    * 在jsp页面中不需要获取和创建，可以直接使用的对象
    * jsp一共有9个内置对象。
    * 今天学习3个：
        * request
        * response
        * out：字符输出流对象。可以将数据输出到页面上。和response.getWriter()类似
            * response.getWriter()和out.write()的区别：
                * 在tomcat服务器真正给客户端做出响应之前，会先找response缓冲区数据，再找out缓冲区数据。
                * response.getWriter()数据输出永远在out.write()之前

5. 案例:改造Cookie案例

Session：主菜

1. 概念：服务器端会话技术，在一次会话的多次请求间共享数据，将数据保存在服务器端的对象中。HttpSession
2. 快速入门：
    1. 获取HttpSession对象：
        ```
        HttpSession session = request.getSession();
        ```
    2. 使用HttpSession对象：
        ```
        Object getAttribute(String name)
        void setAttribute(String name, Object value)
        void removeAttribute(String name)
        ```

3. 原理
    * Session的实现是依赖于Cookie的。

4. 细节：
    1. 当客户端关闭后，服务器不关闭，两次获取session是否为同一个？
        * 默认情况下。不是。

* 如果需要相同，则可以创建Cookie,键为JSESSIONID，设置最大存活时间，让cookie持久化保存。

```
Cookie c = new Cookie("JSESSIONID",session.getId());
c.setMaxAge(60*60);
response.addCookie(c);
```

2. 客户端不关闭，服务器关闭后，两次获取的session是同一个吗？
    * 不是同一个，但是要确保数据不丢失。tomcat自动完成以下工作
        * session的钝化：
            * 在服务器正常关闭之前，将session对象系列化到硬盘上
        * session的活化：
            * 在服务器启动后，将session文件转化为内存中的session对象即可。

3. session什么时候被销毁？
    1. 服务器关闭
    2. session对象调用invalidate() 。
    3. session默认失效时间 30分钟

        选择性配置修改

        ```
        <session-config>
            <session-timeout>30</session-timeout>
        </session-config>
        ```

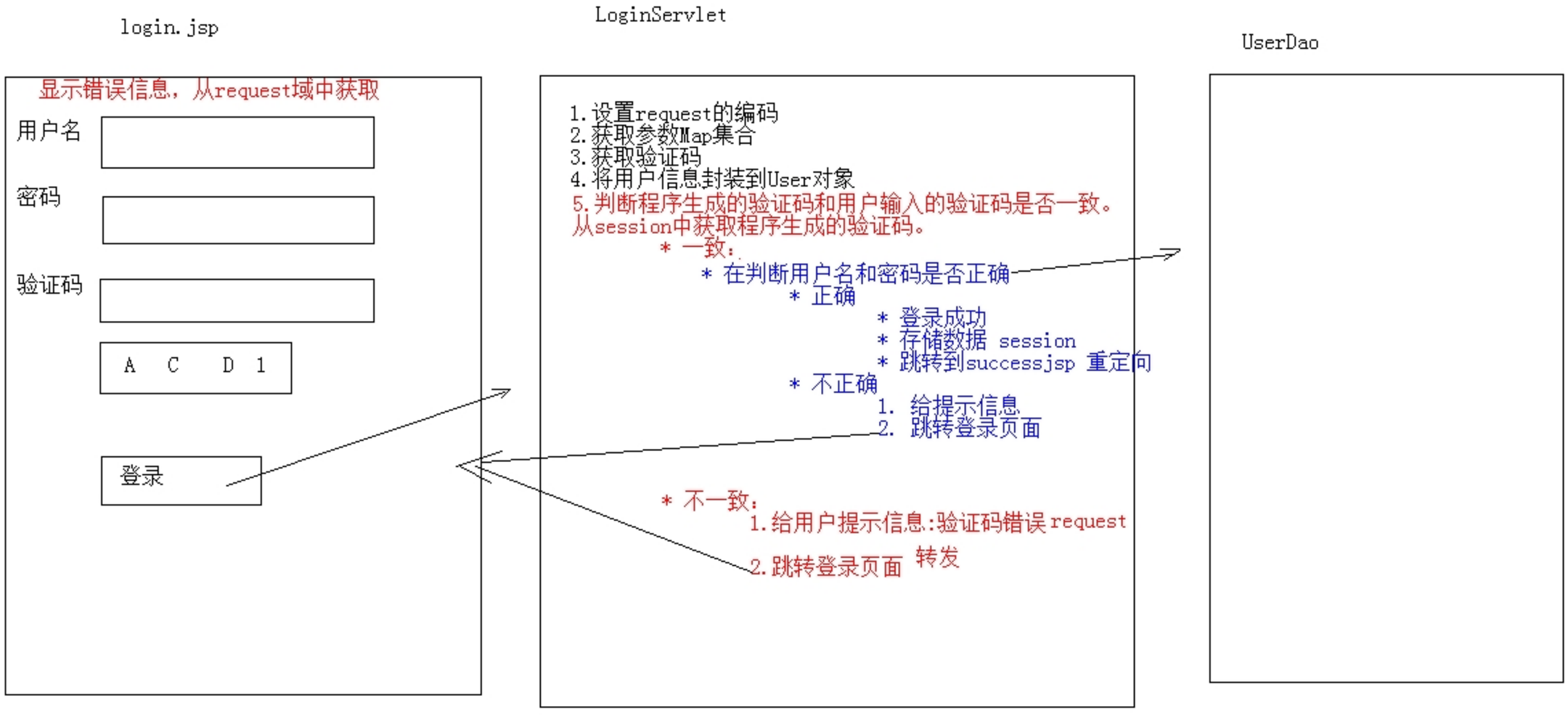
5. session的特点
    1. session用于存储一次会话的多次请求的数据，存在服务器端
    2. session可以存储任意类型，任意大小的数据

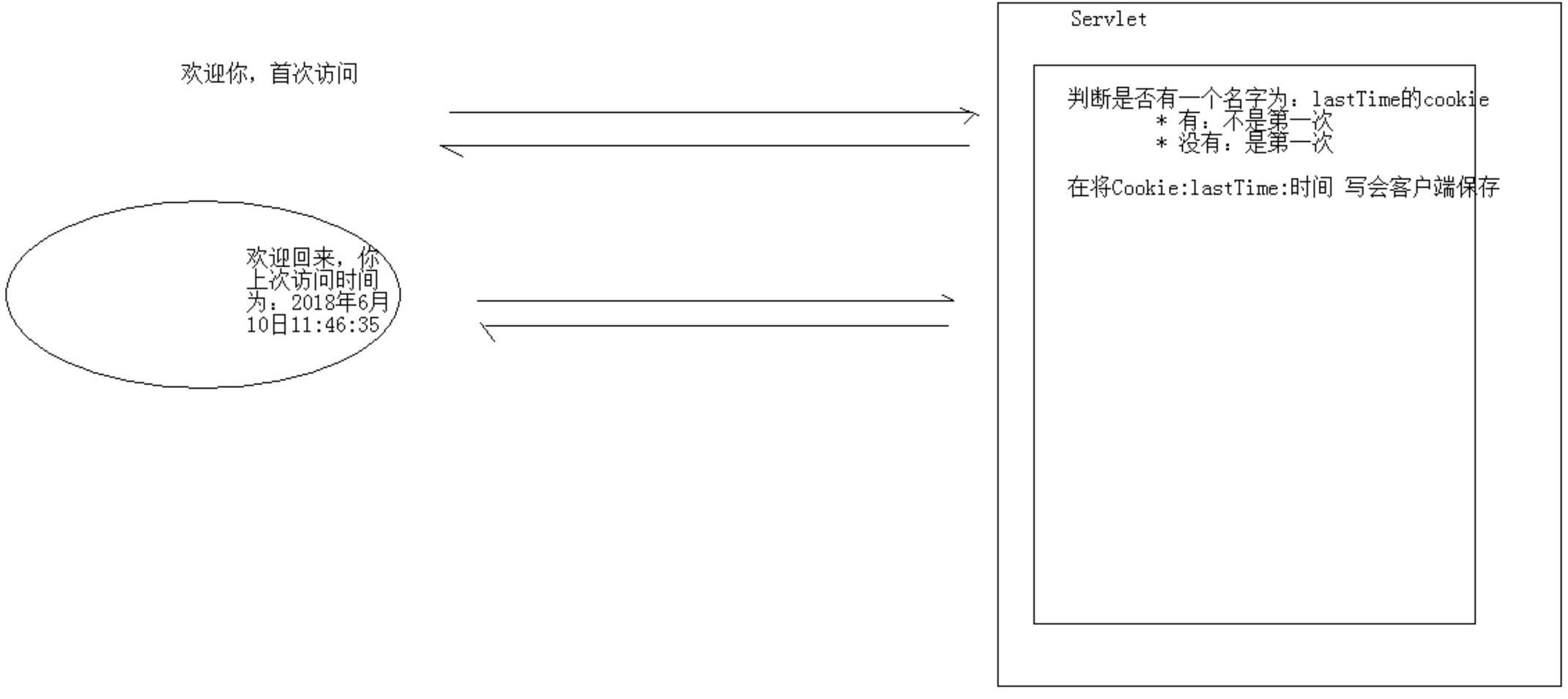
* session与Cookie的区别：
    1. session存储数据在服务器端，Cookie在客户端
    2. session没有数据大小限制，Cookie有
    3. session数据安全，Cookie相对于不安全

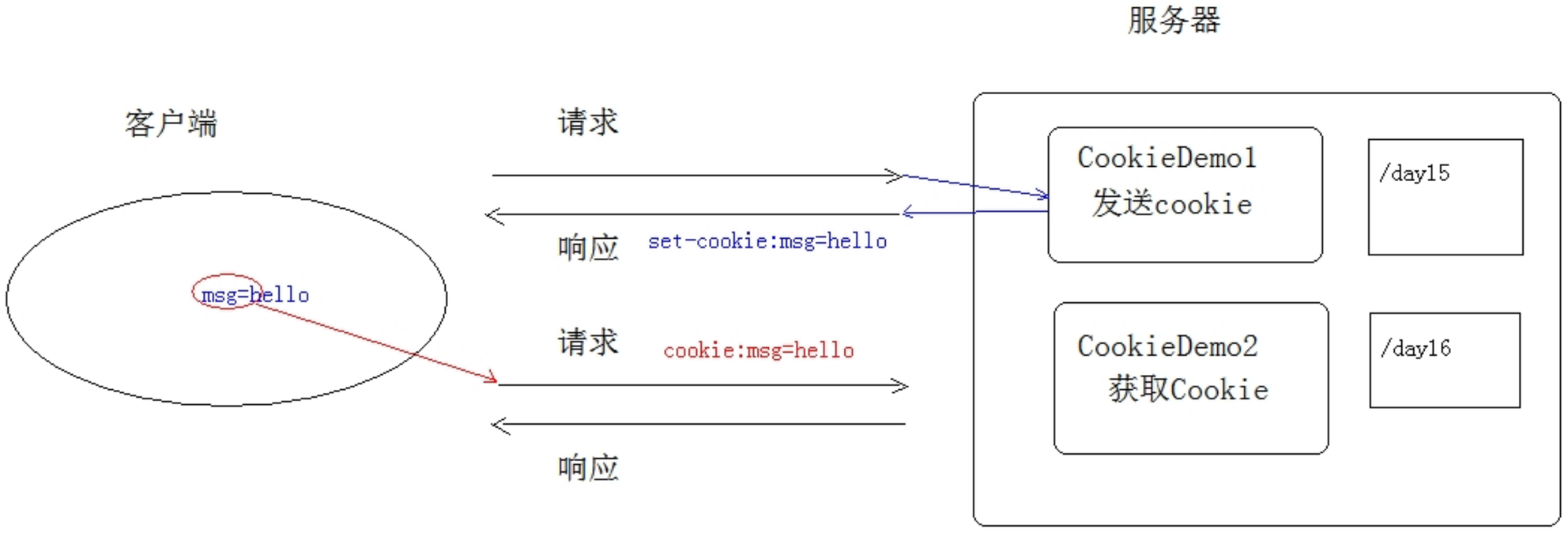
案例：验证码

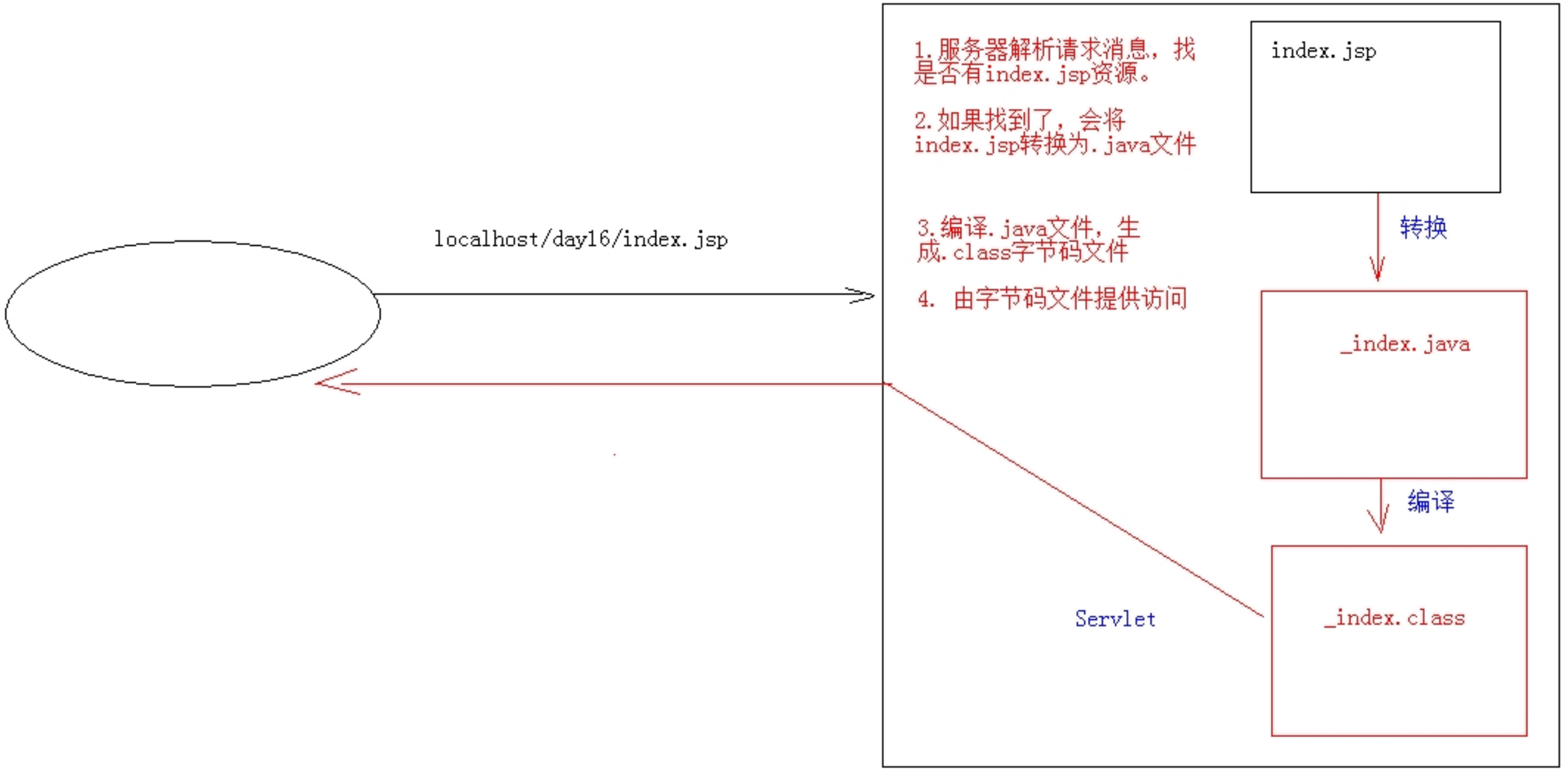
1. 案例需求：
    1. 访问带有验证码的登录页面login.jsp
    2. 用户输入用户名，密码以及验证码。
        * 如果用户名和密码输入有误，跳转登录页面，提示:用户名或密码错误
        * 如果验证码输入有误，跳转登录页面，提示：验证码错误
        * 如果全部输入正确，则跳转到主页success.jsp，显示：用户名,欢迎您

2. 分析：


login.jsp
LoginServlet
UserDao
显示错误信息，从request域中获取
用户名
密码
验证码
A C D 1
登录
1.设置request的编码
2.获取参数Map集合
3.获取验证码
4.将用户信息封装到User对象
5.判断程序生成的验证码和用户输入的验证码是否一致。
从session中获取程序生成的验证码。
* 一致：
* 在判断用户名和密码是否正确
* 正确
* 登录成功
* 存储数据 session
* 跳转到successjsp 重定向
* 不正确
1. 给提示信息
2. 跳转登录页面
* 不一致：
1.给用户提示信息:验证码错误 request
2.跳转登录页面 转发


欢迎你，首次访问
欢迎回来，你上次访问时间为：2018年6月10日11:46:35
Servlet
判断是否有一个名字为：lastTime的cookie
* 有：不是第一次
* 没有：是第一次
在将Cookie:lastTime:时间 写会客户端保存


服务器
客户端
请求
响应
set-cookie:msg=hello
msg=hello
请求
cookie:msg=hello
响应
CookieDemo1
发送cookie
/day15
CookieDemo2
获取Cookie
/day16


1.服务器解析请求消息，找是否有index.jsp资源。
2.如果找到了，会将index.jsp转换为.java文件
3.编译.java文件，生成.class字节码文件
4. 由字节码文件提供访问
index.jsp
转换
_index.java
编译
_index.class
Servlet
localhost/day16/index.jsp

Session是依赖于Cookie的!

服务器如何确保在一次会话范围内，多次获取的Session对象是同一个？？？