1. Cookie

1.1 什么是 Cookie

1、Cookie 翻译过来是饼干的意思。

2、Cookie 是服务器通知客户端保存键值对的一种技术。

3、客户端有了 Cookie 后，每次请求都发送给服务器。

4、每个 Cookie 的大小不能超过 4kb

1.2 如何创建 Cookie

创建 CookieServlet

```
package com.admiral.web;

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @author 白世鑫
 * @title: CookieServlet
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/4  2:42 上午
 */
public class CookieServlet extends BaseServlet{

    protected void createCookie(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        //1.创建 Cookie 对象
        Cookie cookie = new Cookie("key1","value1");
        //2.通知浏览器保存 Cookie 对象
        response.addCookie(cookie);

        //3.响应
        response.getWriter().write("Cookie 被创建了");
    }
```

```
}
```

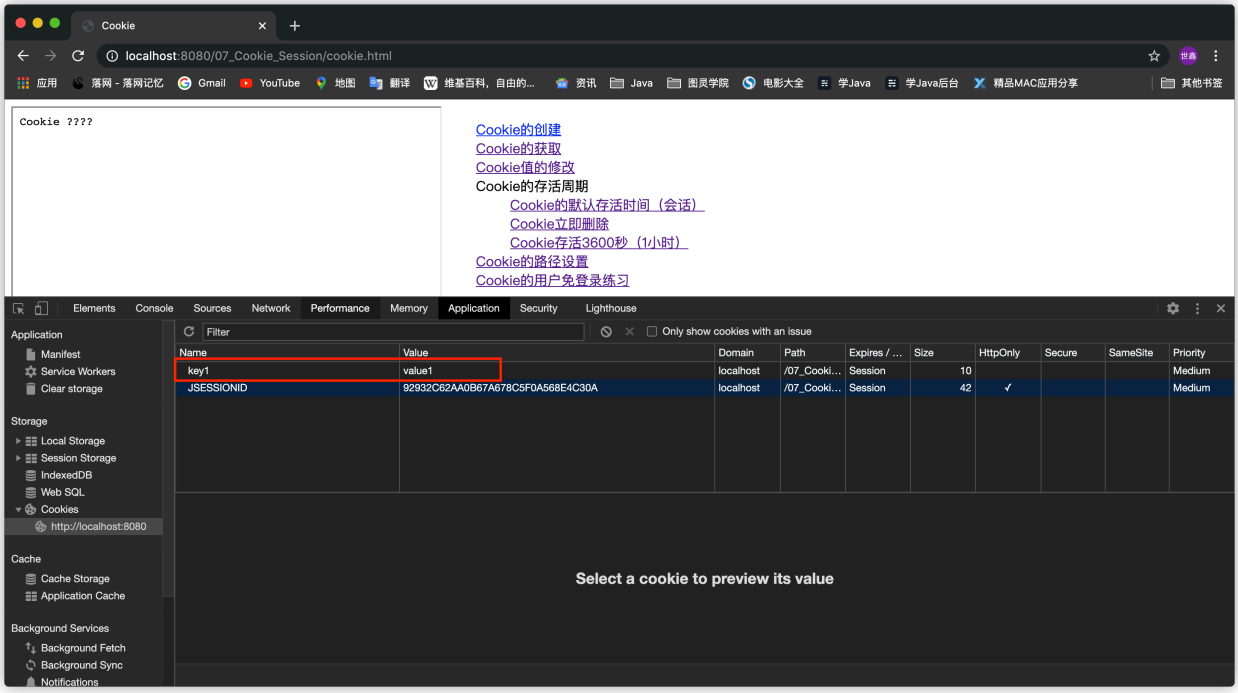
配置 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <servlet>
        <servlet-name>CookieServlet</servlet-name>
        <servlet-class>com.admiral.web.CookieServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>CookieServlet</servlet-name>
        <url-pattern>/cookieServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

在页面上添加请求地址

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="pragma" content="no-cache"/>
    <meta http-equiv="cache-control" content="no-cache"/>
    <meta http-equiv="Expires" content="0"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <base href="http://localhost:8080/07_Cookie_Session/">

    <title>Cookie</title>
    <style type="text/css">

        ul li {
            list-style: none;
        }

    </style>
</head>
<body>
<iframe name="target" width="500" height="500" style="float: left;"></iframe>
<div style="float: left;">
```

```
    <ul>
        <li><a href="cookieServlet?action=createCookie" target="target">Cookie的创建</a></li>
        <li><a href="" target="target">Cookie的获取</a></li>
        <li><a href="" target="target">Cookie值的修改</a></li>
        <li>Cookie的存活周期</li>
        <li>
            <ul>
                <li><a href="" target="target">Cookie的默认存活时间（会话）</a></li>
                <li><a href="" target="target">Cookie立即删除</a></li>
                <li><a href="" target="target">Cookie存活3600秒（1小时）</a></li>
            </ul>
        </li>
        <li><a href="" target="target">Cookie的路径设置</a></li>
        <li><a href="" target="target">Cookie的用户免登录练习</a></li>
    </ul>
</div>
</body>
</html>
```



在 CookieServlet 的 createCookie 中添加如下代码,解决响应中文乱码问题.

```
//解决响应中文乱码问题
response.setContentType("text/html; charset=utf-8");
```

1.3 服务器如何获取 Cookie

通过 request.getCookies() API可以获取到所有的 Cookie 数组

```
    protected void getCookie(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Cookie[] cookies = request.getCookies();
        for (Cookie cookie : cookies) {
            response.getWriter().write("Cookie["+ cookie.getName() + "=" +
cookie.getValue() + "] <br/>");
        }
    }
```

获取指定的Cookie

```
    protected void getCookie(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Cookie[] cookies = request.getCookies();
        for (Cookie cookie : cookies) {
            response.getWriter().write("Cookie["+ cookie.getName() + "=" +
cookie.getValue() + "]");
        }

        //获取指定的 Cookie
        Cookie myCookie = null;
        for (Cookie cookie : cookies) {
            if(cookie.getName().equals("key1")){
                myCookie = cookie;
            }
        }
        if(myCookie!=null){
            response.getWriter().write("找到了需要的 Cookie");
        }


    }
```

将获取指定Cookie的代码抽取到工具类中

```
package com.admiral.utils;

import javax.servlet.http.Cookie;
```

```
/**
 * @author 白世鑫
 * @title: CookieUtils
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/4  3:42 上午
 */
public class CookieUtils {

    public static Cookie getCookie(String name,Cookie[] cookies){
        if(name==null || cookies==null || cookies.length == 0){
            return  null;
        }
        for (Cookie cookie : cookies) {
            if(cookie.getName().equals(name)){
               return cookie;
            }
        }
        return null;
    }
}
```

1.4 Cookie 值的修改

方案一：

1. 先创建一个要修改的同名（指的就是 key）的 Cookie 对象
2. 在构造器，同时赋于新的 Cookie 值。
3. 调用 response.addCookie(Cookie);

```
    protected void updateCookie(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        //解决响应中文乱码问题
        response.setContentType("text/html; charset=utf-8");
        Cookie cookie = new Cookie("key2", "newValue2");
        response.addCookie(cookie);
        response.getWriter().write("key2 的值已经修改了");
    }
```

方案二：

1. 先查找到需要修改的 Cookie 对象
2. 调用 setValue()方法赋于新的 Cookie 值。
3. 调用 response.addCookie()通知客户端保存修改

```
    protected void updateCookie(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        //解决响应中文乱码问题
        response.setContentType("text/html; charset=utf-8");

        Cookie key3 = CookieUtils.getCookie("key3", request.getCookies());
        if (key3 != null) {
            key3.setValue("newValue3");
            response.addCookie(key3);
        }
```

1.5 浏览器查看 Cookie

1.5.1 谷歌浏览器

1.5.2 火狐浏览器

1.6 Cookie 生命控制

Cookie 的生命控制指的是如何管理 Cookie 什么时候被销毁（删除）

- setMaxAge()
    - 正数，表示在指定的秒数后过期
    - 负数，表示浏览器一关，Cookie 就会被删除（默认值是-1）
    - 零，表示马上删除 Cookie

```
package com.admiral.web;

import com.admiral.utils.CookieUtils;

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```

```
/**
 * @author 白世鑫
 * @title: CookieServlet
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/4  2:42 上午
 */
public class CookieServlet extends BaseServlet {

    /**
     * 设置存活一个小时的Cookie
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    protected void life3600(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Cookie cookie = new Cookie("life3600","life3600");
        cookie.setMaxAge(60*60);
        response.addCookie(cookie);
    }


    /**
     * 设置立即删除的Cookie
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    protected void deleteNow(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Cookie cookie = new Cookie("deleteNow","deleteNow");
        cookie.setMaxAge(0);
        response.addCookie(cookie);
    }

    /**
     * 设置默认的会话级别的Cookie
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    protected void defaultCookie(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        Cookie cookie = new Cookie("defaultLife","defaultLife");
```

```
        cookie.setMaxAge(-1);
        response.addCookie(cookie);
    }
}
```

1.7 Cookie 有效路径 Path 的设置

Cookie 的 path 属性可以有效的过滤哪些 Cookie 可以发送给服务器。哪些不发。

path 属性是通过请求的地址来进行有效的过滤。

CookieA path=/工程路径

CookieB path=/工程路径/abc

请求地址如下：

- http://ip:port/工程路径/a.html
  - CookieA 发送
  - CookieB 不发送
- http://ip:port/工程路径/abc/a.html
  - CookieA 发送
  - CookieB 发送

```
    protected void testPath(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Cookie cookie = new Cookie("path","path");
        cookie.setPath(request.getContextPath()+"/aaa");
        response.addCookie(cookie);
    }
```

1.8 Cookie 练习

login.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: baishixin
  Date: 2020/9/7
  Time: 2:33 上午
  To change this template use File | Settings | File Templates.
--%>
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="http://localhost:8080/07_Cookie_Session/loginServlet"
method="get">
        账号:<input type="text" name="username"
value="${cookie.username.value}"><br>
        密码:<input type="password" name="password" value=""><br>
        <input type="submit" value="登陆">
    </form>
</body>
</html>
```

LoginServlet.java

```
package com.admiral.web;

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        if("admin".equals(username) && "123456".equals(password)){
            System.out.println("登陆成功");

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

        }else {
            System.out.println("登陆失败");
        }
```

```
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <servlet>
        <servlet-name>CookieServlet</servlet-name>
        <servlet-class>com.admiral.web.CookieServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>CookieServlet</servlet-name>
        <url-pattern>/cookieServlet</url-pattern>
    </servlet-mapping>


    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>com.admiral.web.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/loginServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

2. Session

2.1 什么是 Session 会话

1、Session 就一个接口（HttpSession）。

2、Session 就是会话。它是用来维护一个客户端和服务器之间关联的一种技术。

3、每个客户端都有自己的一个 Session 会话。

4、Session 会话中，我们经常用来保存用户登录之后的信息。

2.2 如何创建 Session 和获取(id 号,是否为新)

如何创建和获取 Session。它们的 API 是一样的。

request.getSession()

第一次调用是：创建 Session 会话

之后调用都是：获取前面创建好的 Session 会话对象。

isNew(); 判断到底是不是刚创建出来的（新的）

true 表示刚创建

false 表示获取之前创建

每个会话都有一个身份证号。也就是 ID 值。而且这个 ID 是唯一的。

getId() 得到 Session 的会话 id 值。

SessionServlet.java

```
package com.admiral.web;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * @author 白世鑫
 * @title: SessionServlet
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/7  2:49 上午
 */
public class SessionServlet extends BaseServlet{

    protected void createOrGetSession(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/html; charset=utf-8");
```

```
        HttpSession session = req.getSession();

        //判断 Session 是否为新创建的
        boolean isNew = session.isNew();

        //获取 Session 的ID
        String id = session.getId();

        resp.getWriter().write("得到的Session,它的ID为:" + id + "<br/>");
        resp.getWriter().write("Session是否为新创建的:" + isNew + "<br/>");


    }
}
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <servlet>
        <servlet-name>SessionServlet</servlet-name>
        <servlet-class>com.admiral.web.SessionServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SessionServlet</servlet-name>
        <url-pattern>/sessionServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Session.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="pragma" content="no-cache"/>
    <meta http-equiv="cache-control" content="no-cache"/>
    <meta http-equiv="Expires" content="0"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <base href="http://localhost:8080/07_Cookie_Session/">
```

```
    <title>Session</title>
    <style type="text/css">

        ul li {
            list-style: none;
        }

    </style>
</head>
<body>
<iframe name="target" width="500" height="500" style="float: left;"></iframe>
<div style="float: left;">
    <ul>
        <li><a href="sessionServlet?action=createOrGetSession"
target="target">Session的创建和获取（id号、是否为新创建）</a></li>
        <li><a href="" target="target">Session域数据的存储</a></li>
        <li><a href="" target="target">Session域数据的获取</a></li>
        <li>Session的存活</li>
        <li>
            <ul>
                <li><a href="" target="target">Session的默认超时及配置</a></li>
                <li><a href="" target="target">Session3秒超时销毁</a></li>
                <li><a href="" target="target">Session马上销毁</a></li>
            </ul>
        </li>
        <li><a href="" target="target">浏览器和Session绑定的原理</a></li>
    </ul>
</div>
</body>
</html>
```

2.3 Session 域数据的存取

SessionServlet.java

```
package com.admiral.web;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
```

```
/**
 * @author 白世鑫
 * @title: SessionServlet
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/7  2:49 上午
 */
public class SessionServlet extends BaseServlet{

    protected void getAttribute(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        resp.setContentType("text/html; charset=utf-8");
        Object value = req.getSession().getAttribute("key1");
        resp.getWriter().write("Session中key1的值为:" + value);
    }
    protected void setAttribute(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        resp.setContentType("text/html; charset=utf-8");
        req.getSession().setAttribute("key1","vaule1");
        resp.getWriter().write("已经在Session中保存数据");

    }
    protected void createOrGetSession(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/html; charset=utf-8");
        HttpSession session = req.getSession();

        //判断 Session 是否为新创建的
        boolean isNew = session.isNew();

        //获取 Session 的ID
        String id = session.getId();

        resp.getWriter().write("得到的Session,它的ID为:" + id + "<br/>");
        resp.getWriter().write("Session是否为新创建的:" + isNew + "<br/>");
    }
}
```

session.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="pragma" content="no-cache"/>
    <meta http-equiv="cache-control" content="no-cache"/>
    <meta http-equiv="Expires" content="0"/>
```

```
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <base href="http://localhost:8080/07_Cookie_Session/">

    <title>Session</title>
    <style type="text/css">

        ul li {
            list-style: none;
        }

    </style>
</head>
<body>
<iframe name="target" width="500" height="500" style="float: left;"></iframe>
<div style="float: left;">
    <ul>
        <li><a href="sessionServlet?action=createOrGetSession"
target="target">Session的创建和获取（id号、是否为新创建）</a></li>
        <li><a href="sessionServlet?action=setAttribute"
target="target">Session域数据的存储</a></li>
        <li><a href="sessionServlet?action=getAttribute"
target="target">Session域数据的获取</a></li>
        <li>Session的存活</li>
        <li>
            <ul>
                <li><a href="" target="target">Session的默认超时及配置</a></li>
                <li><a href="" target="target">Session3秒超时销毁</a></li>
                <li><a href="" target="target">Session马上销毁</a></li>
            </ul>
        </li>
        <li><a href="" target="target">浏览器和Session绑定的原理</a></li>
    </ul>
</div>
</body>
</html>
```

2.4 Session 生命周期控制

- public void setMaxInactiveInterval(int interval) 设置 Session 的超时时间（以秒为单位），超过指定的时长，Session就会被销毁。
    - 值为正数的时候，设定 Session 的超时时长。
    - 负数表示永不超时（极少使用）

- public int getMaxInactiveInterval()获取 Session 的超时时间

- public void invalidate() 让当前 Session 会话马上超时无效。

Session 默认的超时时长是多少!

Session 默认的超时时间长为 30 分钟。

因为在 Tomcat 服务器的配置文件 web.xml中默认有以下的配置，它就表示配置了当前 Tomcat 服务器下所有的 Session超时配置默认时长为：30 分钟。

```
<session-config>

  <session-timeout>30</session-timeout>

</session-config>
```

如果说。你希望你的 web 工程，默认的 Session 的超时时长为其他时长。你可以在你自己的 web.xml 配置文件中做

以上相同的配置。就可以修改你的 web 工程所有 Seession 的默认超时时长。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">


    <!--表示当前 web 工程。创建出来 的所有 Session 默认是 20 分钟 超时时长-->
    <session-config>
        <session-timeout>20</session-timeout>
    </session-config>
</web-app>
```

如果你想只修改个别 Session 的超时时长。就可以使用上面的 API。setMaxInactiveInterval(int interval)来进行单独的设置。

- session.setMaxInactiveInterval(int interval)单独设置超时时长。

```
    protected void life3(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        req.getSession().setMaxInactiveInterval(3);
        resp.getWriter().write("Session的超时时间已经设置为3秒");
    }
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="pragma" content="no-cache"/>
    <meta http-equiv="cache-control" content="no-cache"/>
    <meta http-equiv="Expires" content="0"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <base href="http://localhost:8080/07_Cookie_Session/">

    <title>Session</title>
    <style type="text/css">

        ul li {
            list-style: none;
        }

    </style>
</head>
<body>
<iframe name="target" width="500" height="500" style="float: left;"></iframe>
<div style="float: left;">
    <ul>
        <li><a href="sessionServlet?action=createOrGetSession"
target="target">Session的创建和获取（id号、是否为新创建）</a></li>
        <li><a href="sessionServlet?action=setAttribute"
target="target">Session域数据的存储</a></li>
        <li><a href="sessionServlet?action=getAttribute"
target="target">Session域数据的获取</a></li>
        <li>Session的存活</li>
        <li>
            <ul>
                <li><a href="sessionServlet?action=getMaxInactiveInterval"
target="target">Session的默认超时及配置</a></li>
                <li><a href="sessionServlet?action=life3"
target="target">Session3秒超时销毁</a></li>
                <li><a href="" target="target">Session马上销毁</a></li>
            </ul>
        </li>
        <li><a href="" target="target">浏览器和Session绑定的原理</a></li>
    </ul>
```

```
</div>
</body>
</html>
```

Session 马上被超时示例：

```
    protected void deleteNow(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        //设置Session马上超时
        req.getSession().invalidate();
        resp.getWriter().write("已经设置Session为超时(无效)");
    }
```

session.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="pragma" content="no-cache"/>
    <meta http-equiv="cache-control" content="no-cache"/>
    <meta http-equiv="Expires" content="0"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <base href="http://localhost:8080/07_Cookie_Session/">

    <title>Session</title>
    <style type="text/css">

        ul li {
            list-style: none;
        }

    </style>
</head>
<body>
<iframe name="target" width="500" height="500" style="float: left;"></iframe>
<div style="float: left;">
    <ul>
        <li><a href="sessionServlet?action=createOrGetSession"
target="target">Session的创建和获取（id号、是否为新创建）</a></li>
        <li><a href="sessionServlet?action=setAttribute"
target="target">Session域数据的存储</a></li>
        <li><a href="sessionServlet?action=getAttribute"
target="target">Session域数据的获取</a></li>
        <li>Session的存活</li>
        <li>
```

```
            <ul>
                <li><a href="sessionServlet?action=getMaxInactiveInterval"
target="target">Session的默认超时及配置</a></li>
                <li><a href="sessionServlet?action=life3"
target="target">Session3秒超时销毁</a></li>
                <li><a href="sessionServlet?action=deleteNow"
target="target">Session马上销毁</a></li>
            </ul>
        </li>
        <li><a href="" target="target">浏览器和Session绑定的原理</a></li>
    </ul>
</div>
</body>
</html>
```

2.5 浏览器和 Session 之间关联的技术内幕

3. 项目第六阶段

3.1 登陆---显示用户名

登陆成功后先将用户信息保存到 Session 域中

```
protected void login(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        //1.获取请求参数
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        //2.调用 UserService 的 login 方法
        User login = userService.login(new User(null, username, password,
null));
        if (login != null) {
            //成功,跳转到登陆成功页面
```

```
            //保存用户信息到 Session 域中
            request.getSession().setAttribute("user",login);


 request.getRequestDispatcher("/pages/user/login_success.jsp").forward(request
, response);
        } else {
            //保存错误信息到reques域中
            request.setAttribute("errorMsg", "用户名或密码错误!");
            request.setAttribute("username", username);
            //失败.跳转到登陆页面

 request.getRequestDispatcher("/pages/user/login.jsp").forward(request,
response);
        }
    }
```

修改success_menu.jsp,从 Session域中取值

```
<%--
  Created by IntelliJ IDEA.
  User: baishixin
  Date: 2020/9/1
  Time: 3:34 上午
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<div>
    <span>欢迎<span class="um_span">${sessionScope.user.username}</span>光临书城
</span>
    <a href="pages/order/order.jsp">我的订单</a>
    <a href="index.jsp">注销</a>  
    <a href="index.jsp">返回</a>
</div>
```

还需要修改 index.jsp 页面中的菜单

```
    <div>
        <c:if test="${empty sessionScope.user}">
            <a href="pages/user/login.jsp">登录</a> |
            <a href="pages/user/regist.jsp">注册</a>   
        </c:if>
        <c:if test="${not empty sessionScope.user}">
            <span>欢迎<span class="um_span">${sessionScope.user.username}
</span>光临书城</span>
            <a href="pages/order/order.jsp">我的订单</a>
            <a href="index.jsp">注销</a>
        </c:if>
        <a href="pages/cart/cart.jsp">购物车</a>
        <a href="pages/manager/manager.jsp">后台管理</a>
    </div>
```

3.2 登出---注销用户

在 UserServlet中添加注销的方法

```
protected void logout(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //销毁 Session
        request.getSession().invalidate();
        //重定向到首页
        response.sendRedirect(request.getContextPath());
    }
```

修改 success_menu.jsp 和 index.jsp 中注销按钮的请求地址

```
<%--
  Created by IntelliJ IDEA.
  User: baishixin
  Date: 2020/9/1
  Time: 3:34 上午
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<div>
    <span>欢迎<span class="um_span">${sessionScope.user.username}</span>光临书城
</span>
    <a href="pages/order/order.jsp">我的订单</a>
```

```
    <a href="userServlet?action=logout">注销</a>  
    <a href="index.jsp">返回</a>
</div>
```

index.jsp

```
<div id="header">
    <img class="logo_img" alt="" src="static/img/logo.gif">
    <span class="wel_word">网上书城</span>
    <div>
        <c:if test="${empty sessionScope.user}">
            <a href="pages/user/login.jsp">登录</a> |
            <a href="pages/user/regist.jsp">注册</a>   
        </c:if>
        <c:if test="${not empty sessionScope.user}">
            <span>欢迎<span class="um_span">${sessionScope.user.username}
</span>光临书城</span>
            <a href="pages/order/order.jsp">我的订单</a>
            <a href="userServlet?action=logout">注销</a> 
        </c:if>
        <a href="pages/cart/cart.jsp">购物车</a>
        <a href="pages/manager/manager.jsp">后台管理</a>
    </div>
</div>
```

3.3 表单重复提交之-----验证码

表单重复提交有三种常见的情况：

- 一：提交完表单。服务器使用请求转来进行页面跳转。这个时候，用户按下功能键 F5，就会发起最后一次的请求。造成表单重复提交问题。解决方法：使用重定向来进行跳转
- 二：用户正常提交服务器，但是由于网络延迟等原因，迟迟未收到服务器的响应，这个时候，用户以为提交失败，就会着急，然后多点了几次提交操作，也会造成表单重复提交。
- 三：用户正常提交服务器。服务器也没有延迟，但是提交完成后，用户回退浏览器。重新提交。也会造成表单重复提交。

3.4 谷歌 kaptcha 图片验证码的使用

谷歌验证码 kaptcha 使用步骤如下：

1. 导入谷歌验证码的 jar 包
    - kaptcha-2.3.2.jar

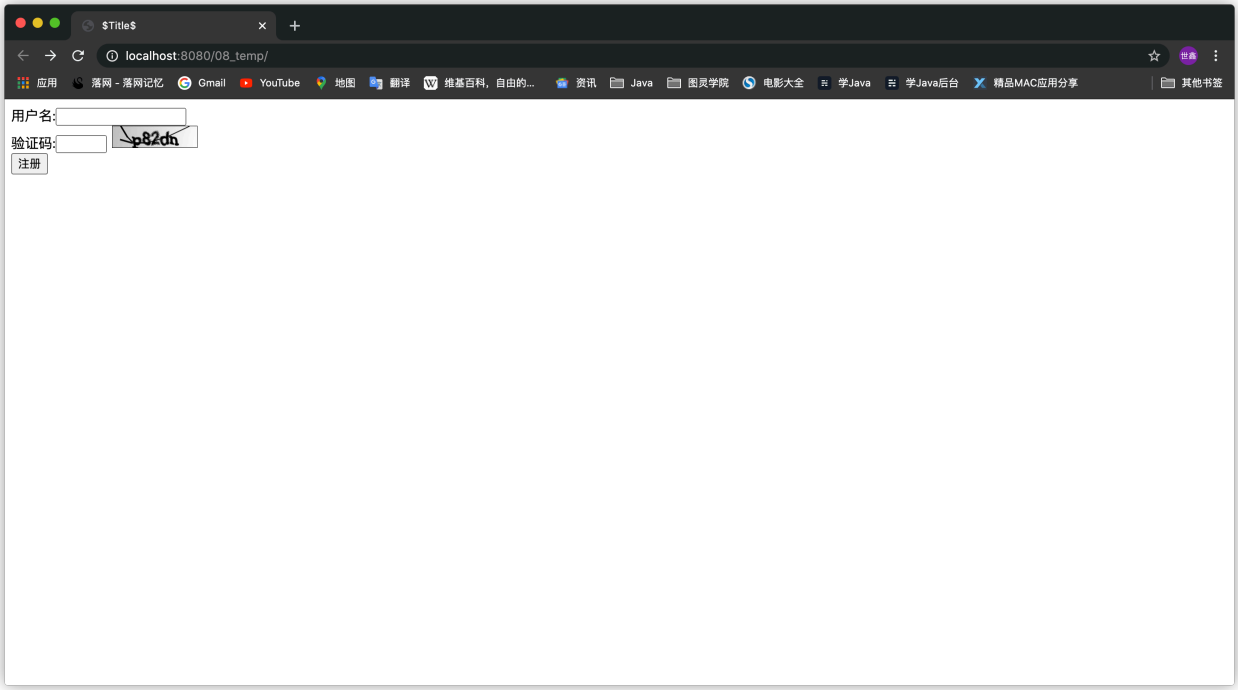
2. 在 web.xml 中去配置用于生成验证码的 Servlet 程序

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>RegisterServlet</servlet-name>
        <servlet-class>com.admiral.web.RegisterServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>RegisterServlet</servlet-name>
        <url-pattern>/registerServlet</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>KaptchaServlet</servlet-name>
        <servlet-
class>com.google.code.kaptcha.servlet.KaptchaServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>KaptchaServlet</servlet-name>
        <url-pattern>/kaptcha.jpg</url-pattern>
    </servlet-mapping>
</web-app>
```

3. 在表单中使用 img 标签去显示验证码图片并使用它

```
<%--
  Created by IntelliJ IDEA.
  User: baishixin
  Date: 2020/9/7
  Time: 4:15 上午
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>$Title$</title>
</head>
<body>
  <form action="http://localhost:8080/08_temp/registerServlet" method="get">
      用户名:<input type="text" name="username"><br>
      验证码:<input type="text" name="code" style="width: 60px">
```

```
      <img src="http://localhost:8080/08_temp/kaptcha.jpg" style="width:
100px;height: 26px"><br>
      <input type="submit" value="注册">
  </form>
</body>
</html>
```



4. 在服务器获取谷歌生成的验证码和客户端发送过来的验证码比较使用。

```
package com.admiral.web;

import java.io.IOException;

import static com.google.code.kaptcha.Constants.KAPTCHA_SESSION_KEY;

public class RegisterServlet extends javax.servlet.http.HttpServlet {
    protected void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, IOException {

    }

    protected void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");

        //获取 Session 中的验证码
        Object token = request.getSession().getAttribute(KAPTCHA_SESSION_KEY);
        //删除 Session 中的验证码
```

```
        request.getSession().removeAttribute(KAPTCHA_SESSION_KEY);

        String code = request.getParameter("code");
        String username = request.getParameter("username");

        if(token!=null && token.equals(code)){
            System.out.println("保存数据" + username);
            response.sendRedirect(request.getContextPath()+"/ok.jsp");
        }else {
            System.out.println("请勿重复提交表单");
        }
    }
}
```

切换验证码:

```
// 给验证码的图片，绑定单击事件
            $("#code_img").click(function () {
                // 在事件响应的 function 函数中有一个 this 对象。这个 this 对象，是当前正在响应事件的 dom 对象
                // src 属性表示验证码 img 标签的 图片路径。它可读，可写
                // alert(this.src);
                this.src = "${basePath}kaptcha.jpg?d=" + new Date();
            });
```