1、关于web编程中的Session

1、--Session表示会话，不仅仅是在javaweb中有，只要是web开发都有会话机制

2、--java中会话对应的类型/接口是javax.servlet.http.HttpSession,简称session/会话

3、--Cookie是可以将会话保存在客户端，而HttpSession则是将会话保存到服务器端

4、--HttpSession对象是一个会话级别的对象，一次会话对应一个HttpSession对象
可以联想到ServletContext对象是应用（项目）级别，HttpServletRequest对象是请求级别的

5、--一次会话的含义：
目前这样理解：用户打开浏览器，对同一个web项目在浏览器上发送多次请求，直到关闭浏览器，这就是一次完整会话
通过创建web项目测试，用本电脑访问web项目，打印输出到控制台的的本机ip地址和会话session对象信息如下：
"0:0:0:0:0:0:0:1's session = org.apache.catalina.session.StandardSessionFacade@7f052f3"
没有关闭浏览器的情况下一直刷新该页面上面的session对象没有改变
当浏览器关闭后，再次打开浏览器访问该项目，输出信息如下：
"0:0:0:0:0:0:0:1's session = org.apache.catalina.session.StandardSessionFacade@338f1d61"
可以看到session对象更换了，是一次新的会话
然后用手机访问该项目，输出信息如下：
"192.168.0.100's session = org.apache.catalina.session.StandardSessionFacade@1c49feca"
从上我们也知道Tomcat服务器实现的HttpSession接口的类名是
org.apache.catalina.session.StandardSessionFacade

6、--由测试我们知道了，在会话进行过程中，web服务器一直在为当前这个用户维护这这个会话对象/HttpSession

7、--所以在Web容器中，它维护了大量的HttpSession对象，就是说在Web容器中有一个存储session对象的"session列表"
思考在一次会话中每一次请求都能获取到本次会话的session对象：
--通过对某一次会话的第一次次请求和刷新的请求协议和响应协议中内容的监测，可以知道session的实现原理如下：
* 打开浏览器，在浏览器上发送请求
* 服务器获取到提交的Cookie数组，判断该数组中是否有name为JSESSIONID的Cookie
如果没有：
1）服务器内部程序跳转到这样的代码：执行创建HttpSession对象和对应的Cookie对象，该Cookie的name是
JSESSIONID，其value是一个32位长度的字符串；
2）然后服务器将Cookie的value和HttpSession对象绑定到session列表中，该列表就是一个Map集合，
Cookie的value是作为Map集合的key值，HttpSession对象作为集合的value
3）服务器将Cookie发送到浏览器客户端，浏览器将Cookie保存到缓存中，只要浏览器不关闭，Cookie不会消
除
如果有：
1）服务器内部程序跳转到这样的代码：执行获取name为JSESSIONID的Cookie中的value，用这个value作为
key到sessio列表集合中检索以获取对应的HttpSession对象，返回到request对象中

8、--可以看到和HttpSession对象关联的Cookie的name属性是比较特殊的，在java中叫做：jsessionid

9、--浏览器禁用Cookie出现的问题？以及解决

当浏览器禁用Cookie后，浏览器客户端就不再保存发送过来的Cookie，每次请求也就没有Cookie可以发送，在服务器中

就得不到name为JSESSIONID的Cookie，即使浏览器不关闭，每一次请求都会创建新的HttpSession对象和对应的

Cookie，每一次请求就都是一次全新的会话

--禁用Cookie后还是想拿到对应的Session对象，需要重写URL，如下

http://localhost:8080/idea_servlet_http_session_13_war_exploded/user/accessMySelfSession;jsessionid=某个session对象对应Cookie的value

10、--浏览器关闭后session对象的状态

-浏览器关闭后，服务器不会立即销毁session对象

-B/S架构的系统是基于HTTP协议的，该协议是一种无连接/无状态协议

-无连接/无状态：

请求瞬间浏览器和服务器的通道打开，请求响应结束，通道关闭

这样做可以降低服务器的压力

11、--什么时候session对象被销毁

web系统中有一个session超时的概念

当很长一段时间（这段时间可配置）没有用户在访问该session对象时，该session对象超时，web服务器自动回收该对象

默认超时时间是30分钟，可以在web.xml文件中这样配置：

```
<session-config>
    <session-timeout>120</session-timeout>
</session-config>
```

表示session对象有效时间120分钟

12、--一次会话定义

--多数情况是用户打开浏览器，在浏览器上进行多次操作，然后关闭浏览器，表示一次会话结束

--本质上是：从HttpSession对象被创建到销毁的整个过程，表示一次完整会话

13、--javax.servlet.http.HttpSession接口中的常用方法

一次会话中共享的数据（即跨请求之间共享的数据）的添加、获取以及移除的方法

--void setAttribute(String name, Object value)

--Object getAttribute(String name)

--void removeAttribute(String name)

--void invalidate()

该方法是负责销毁session对象的，例如我们登录成功后是一种会话状态，如果我们退出登录，会话结束，就需要销毁保

存之前登录成功状态的会话

14、--ServletContext接口、HttpSession接口和HttpServletRequest接口都有上面的三个方法，它们之间有什么区别：

14.1 上面三个类型的对象都是范围对象

ServletContext application; 这个对象是应用范围的

HttpSession session; 这个对象是会话范围

HttpServletRequest request; 该对象是请求范围的

14.2 范围大小排序：

application > session >request

14.3

application完成的是跨会话共享数据，前提是这些会话的请求是对同一个webapp项目发送的请求

session完成的是跨请求共享数据，前提是这些请求必须在同一个会话中

request完成的是跨Servlet共享数据，前提是这些Servlet是在同一个请求中（例如转发）

```
    14.4 实际开发中选择的原则是由小到大尝试
    --例如：
        我们登录成功后，已经登录的状态需要保存起来，可以将登录成功这个状态保存到session中
        登录成功后还要发送其他的请求来处理事务，所以：
        --登陆成功的状态不能保存到request范围中，因为一次请求对应一个新的request对象。
        --登陆成功的状态也不能保存到application范围中，登录状态是会话级别，一个用户的安全隐私，不能被所有用户        共享
    我们创建了一个Servlet实现类AccessMySelfSession.java，在该类中：执行如下
        session.setAttribute("username","zhangsan");//向会话范围中存储数据
    然后创建一个Servlet实习类GetDataFromSessionServlet.java,在该类中：执行如下
        HttpSession session = request.getSession();
        Object username = session.getAttribute("username");//从session范围获取数据
        System.out.println(username);
    部署项目启动后到欢迎页面点击请求AccessMySelfSession对象资源的链接，然后点击请求
    GetDataFromSessionServlet对象资源的链接，可以从控制台看到打印出了相关信息，--跨请求共享数据

15、--补充HttpServletRequest中的getSession方法
    HttpSession session = request.getSession();//获取当前的session，获取不到，创建新的session对象

    HttpSession session = request.getSession(true);//获取当前的session，获取不到，创建新的session对象

    HttpSession session = request.getSession(false);//获取当前的session，获取不到，返回null
```

2、创建web项目idea-servlet-http-session-13，测试session对象

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>accessMySelfSeesion</servlet-name>
        <servlet-class>com.servlet.http.AccessMySelfSession</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>accessMySelfSeesion</servlet-name>
        <url-pattern>/user/accessMySelfSession</url-pattern>
    </servlet-mapping>
</web-app>
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>welcome page</title>
</head>
<body>
    <a
href="/idea_servlet_http_session_13_war_exploded/user/accessMySelfSession">访问属
于我的Session对象</a>
</body>
</html>
```

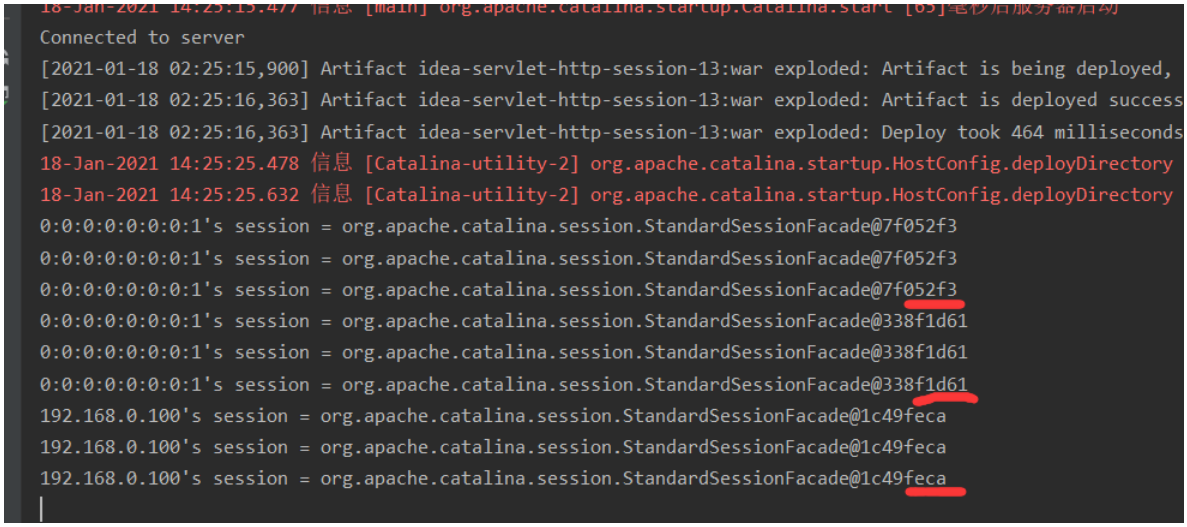
AccessMySelfSession.java

```
package com.servlet.http;

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

public class AccessMySelfSession extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        //获取每次会话用户的IP地址
        String IP = request.getRemoteAddr();

        //获取每次会话用户的Session对象
        HttpSession session = request.getSession();

        //输出
        System.out.println(IP + "'s session = "+session);

    }
}
```

部署项目运行服务器，进入欢迎页面

访问属于我的Session对象

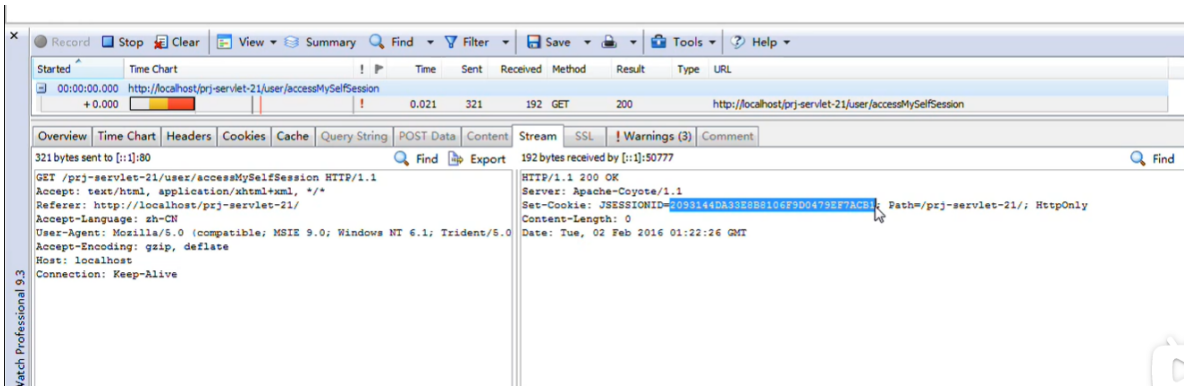
不同用户访问项目结果



其中前三次是本机访问后刷新两次的输出结果，中间三次是关闭浏览器后访问刷新两次的结果，后三次输出是用手机访问刷新两次的结果。

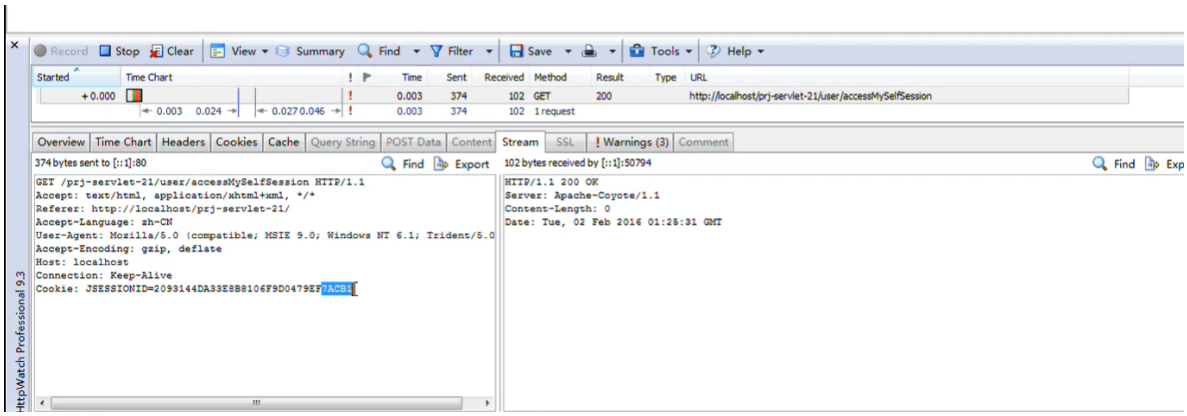
对首次请求和刷新后的请求协议和响应协议内容监测结果

首次请求响应协议



可以看到发送一个Cookie，绑定的路径是web项目的根目录，没有有效时长

刷新请求请求协议



可以看到发送了一个Cookie

Session范围共享数据操作方法测试

创建的GetDataFromSessionServlet.java

```
package com.servlet.http;

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

public class GetDataFromSessionServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        HttpSession session = request.getSession();

        //获取session范围的数据
        Object username = session.getAttribute("username");
        System.out.println(session);
    }
}
```

下面是测试销毁session对象的方法void invalidate()

web.xml修改如下

```
<?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>accessMySelfSeesion</servlet-name>
        <servlet-class>com.servlet.http.AccessMySelfSession</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>accessMySelfSeesion</servlet-name>
        <url-pattern>/user/accessMySelfSession</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>getData</servlet-name>
        <servlet-class>com.servlet.http.GetDataFromSessionServlet</servlet-
class>
    </servlet>
    <servlet-mapping>
        <servlet-name>getData</servlet-name>
        <url-pattern>/user/getDataFromSessionServlet</url-pattern>
    </servlet-mapping>
```

```
    <servlet>
        <servlet-name>logout</servlet-name>
        <servlet-class>com.servlet.http.LogoutServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>logout</servlet-name>
        <url-pattern>/user/logout</url-pattern>
    </servlet-mapping>

    <session-config>
        <session-timeout>120</session-timeout>
    </session-config>
</web-app>
```

index.html修改如下

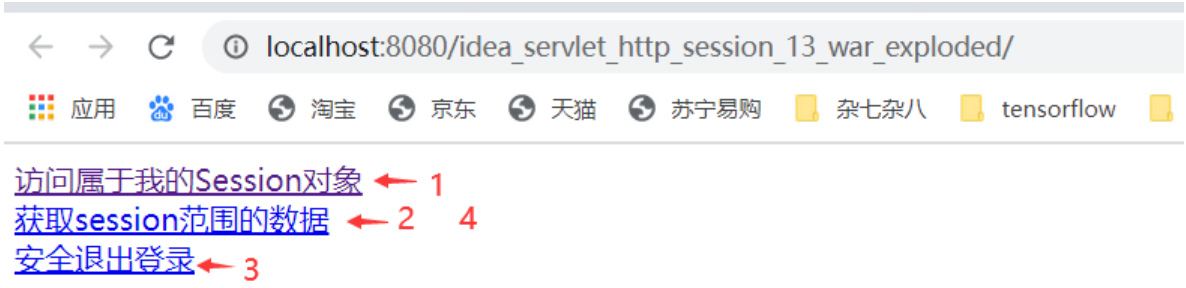
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>welcome page</title>
</head>
<body>
    <a
href="/idea_servlet_http_session_13_war_exploded/user/accessMySelfSession">访问属
于我的Session对象</a>
    <br>
    <a
href="/idea_servlet_http_session_13_war_exploded/user/getDataFromSessionServlet"
>获取session范围的数据</a>
    <br>
    <a href="/idea_servlet_http_session_13_war_exploded/user/logout">安全退出登录
</a>
</body>
</html>
```

LogoutServlet.java如下

```
package com.servlet.http;

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

public class LogoutServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        //实现的是退出登录删除session功能，获取session如果获取不到没有必要创建session对象
        HttpSession session = request.getSession(false);
```

```
        if (session != null) {

            //销毁session对象
            session.invalidate();

        }

    }
}
```

然后部署项目启动服务器，到欢迎页面如下，从上往下依次点击链接，然后点击获取session范围数据的链接



控制台输出结果如下

可知安全退出登录后session被销毁，获取到的session不再是之前的那个session对象，没有共享数据，所以是null