单点登录

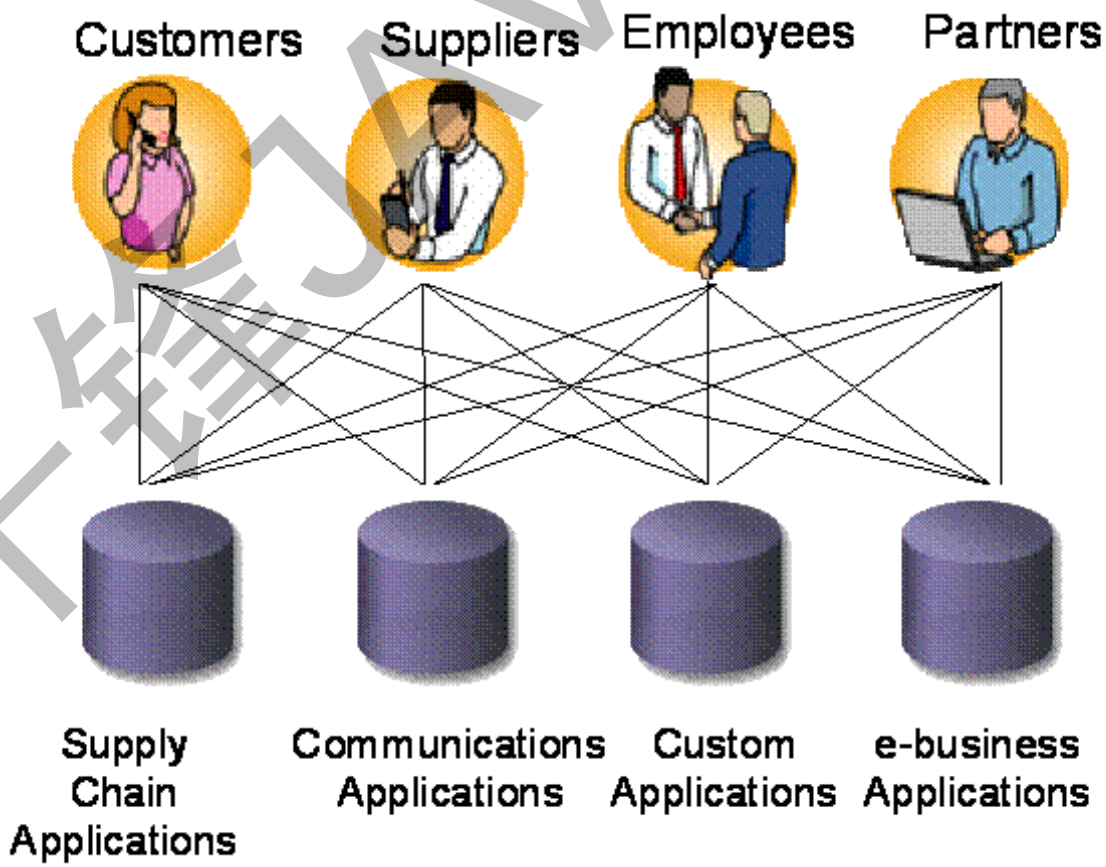
第一节 简介

1.1 什么是单点登陆

单点登录（Single Sign On），简称为 SSO，是目前比较流行的企业业务整合的解决方案之一。SSO的定义是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。

较大的企业内部，一般都有很多的业务支持系统为其提供相应的管理和IT服务。
例如财务系统为财务人员提供财务的管理、计算和报表服务；
人事系统为人事部门提供全公司人员的维护服务；
各种业务系统为公司内部不同的业务提供不同的服务等等。
这些系统的目的都是让计算机来进行复杂繁琐的计算工作，来替代人力的手工劳动，提高工作效率和质量。这些不同的系统往往是在不同的时期建设起来的，运行在不同的平台上；也许是由不同厂商开发，使用了各种不同的技术和标准。

如果举例说国内一著名的IT公司（名字隐去），内部共有60多个业务系统，这些系统包括两个不同版本的SAP的ERP系统，12个不同类型和版本的数据库系统，8个不同类型和版本的操作系统，以及使用了3种不同的防火墙技术，还有数十种互相不能兼容的协议和标准，你相信吗？不要怀疑，这种情况其实非常普遍。每一个应用系统在运行了数年以后，都会成为不可替换的企业IT架构的一部分，如下图所示



随着企业的发展，业务系统的数量在不断的增加，老的系统却不能轻易的替换，这会带来很多的开销。其一是管理上的开销，需要维护的系统越来越多。很多系统的数据是相互冗余和重复的，数据的不一致性会给管理工作带来很大的压力。业务和业务之间的相关性也越来越大，例如公司的计费系统和财务系统，财务系统和人事系统之间都不可避免的有着密切的关系。

为了降低管理的消耗，最大限度的重用已有投资的系统，很多企业都在进行着企业应用集成（EAI）。企业应用集成可以在不同层面上进行：例如在数据存储层面上的“数据大集中”，在传输层面上的“通用数据交换平台”，在应用层面上的“业务流程整合”，和用户界面上的“通用企业门户”等等。事实上，还用一个层面上的集成变得越来越重要，那就是“身份认证”的整合，也就是“单点登录”。

通常来说，每个单独的系统都会有自己的安全体系和身份认证系统。整合以前，进入每个系统都需要进行登录，这样的局面不仅给管理上带来了很大的困难，在安全方面也埋下了重大的隐患。下面是一些著名的调查公司显示的统计数据：

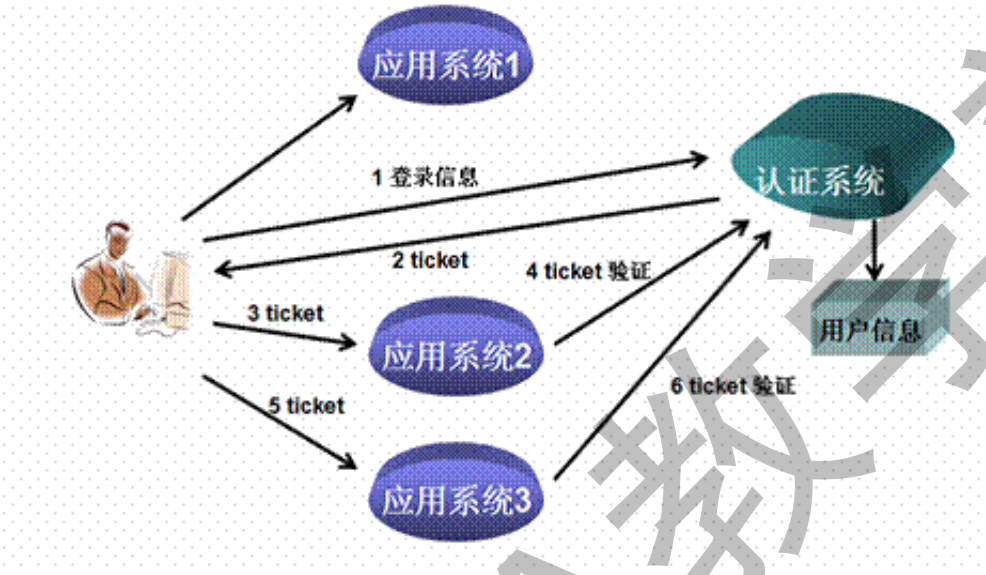
- 用户每天平均 **16** 分钟花在身份验证任务上 - *资料来源：IDS*
- 频繁的 **IT** 用户平均有 **21** 个密码 - *资料来源：NTA Monitor Password Survey*
- 49% 的人写下了其密码，而 **67%** 的人很少改变它们
- 每 **79** 秒出现一起身份被窃事件 - *资料来源：National Small Business Travel Assoc*
- 全球欺骗损失每年约 **12B** - *资料来源：Comm Fraud Control Assoc*
- 到 **2007** 年，身份管理市场将成倍增长至 **$4.5B** - *资料来源：IDS*

另外，使用“单点登录”还是SOA时代的需求之一。在面向服务的架构中，服务和服务之间，程序和程序之间的通讯大量存在，服务之间的安全认证是SOA应用的难点之一，应此建立“单点登录”的系统体系能够大大简化SOA的安全问题，提高服务之间的合作效率。

1.2 单点登陆的技术实现机制

单点登录的机制其实是比较简单的，用一个现实中的例子做比较。颐和园是北京著名的旅游景点，在颐和园内部有许多独立的景点，例如“苏州街”、“佛香阁”和“德和园”，都可以在各个景点门口单独买票。很多游客需要游玩所有的景点，这种买票方式很不方便，需要在每个景点门口排队买票，钱包拿进拿出的，容易丢失，很不安全。于是绝大多数游客选择在大门口买一张通票（也叫套票），就可以玩遍所有的景点而不需要重新再买票。他们只需要在每个景点门口出示一下刚才买的套票就能够被允许进入每个独立的景点

单点登录的机制也一样，如下图所示，当用户第一次访问应用系统1的时候，因为还没有登录，会被引导到认证系统中进行登录（1）；根据用户提供的登录信息，认证系统进行身份效验，如果通过效验，应该返回给用户一个认证的凭据－－ticket（2）；用户再访问别的应用的时候（3，5）就会将这个ticket带上，作为自己认证的凭据，应用系统接受到请求之后会把ticket送到认证系统进行效验，检查ticket的合法性（4，6）。如果通过效验，用户就可以在不用再次登录的情况下访问应用系统2和应用系统3了。



从上面的视图可以看出，要实现SSO，需要以下主要的功能：

所有应用系统共享一个身份认证系统。
统一的认证系统是SSO的前提之一。认证系统的主要功能是将用户的登录信息和用户信息库相比较，对用户进行登录认证；认证成功后，认证系统应该生成统一的认证标志（ticket），返还给用户。另外，认证系统还应该对ticket进行效验，判断其有效性。

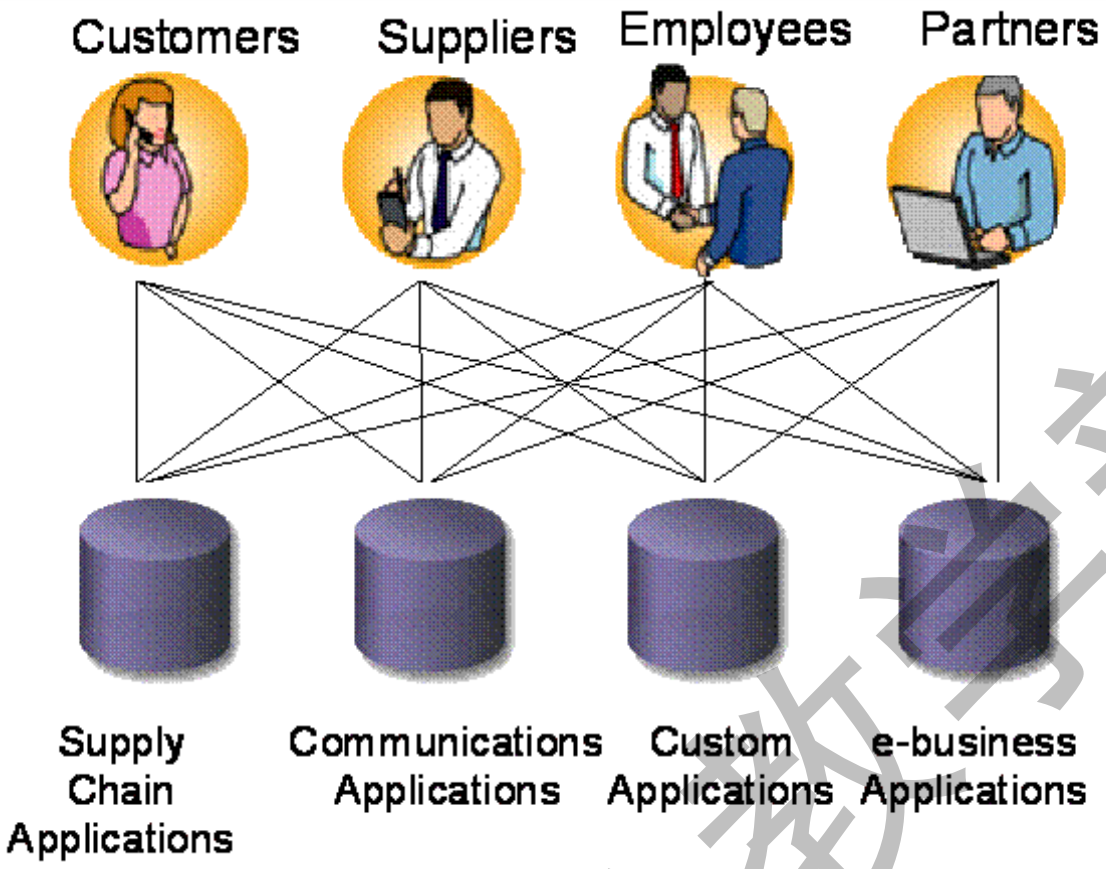
所有应用系统能够识别和提取ticket信息
要实现SSO的功能，让用户只登录一次，就必须让应用系统能够识别已经登录过的用户。应用系统应该能对ticket进行识别和提取，通过与认证系统的通讯，能自动判断当前用户是否登录过，从而完成单点登录的功能。

第二节 跨域请求

2.1 同源策略

所谓同源是指，域名，协议，端口相同。所谓“同源策略“，简单的说就是基于安全考虑，当前域不能访问其他域的东西。

在同源策略下，在某个服务器下的页面是无法获取到该服务器以外的数据的。例如我们在自己的网站通过ajax去获取豆瓣上https://developers.douban.com/wiki/?title=api_v2提供的接口数据。这里我们以搜索图书为例，参数链接为：https://api.douban.com/v2/book/search?q=javascript&count=1



2.2 JSONP

JSONP 是 JSON with padding（填充式 JSON 或参数式 JSON）的简写
JSONP实现跨域请求的原理简单的说，就是动态创建<script>标签，然后利用<script>的src 不受同源策略约束来跨域获取数据。

JSONP 由两部分组成：回调函数和数据。回调函数是当响应到来时应该在页面中调用的函数。回调函数的名字一般是在请求中指定的。而数据就是传入回调函数中的 JSON 数据。

实际项目中JSONP通常用来获取json格式数据，这时前后端通常约定一个参数callback，该参数的值，就是处理返回数据的函数名称。

2.3 JSONP的优缺

JSONP目前还是比较流行的跨域方式，虽然JSONP使用起来方便，但是也存在一些问题：
首先， JSONP 是从其他域中加载代码执行。如果其他域不安全，很可能会在响应中夹带一些恶意代码，而此时除了完全放弃 JSONP 调用之外，没有办法追究。因此在使用不是你自己运维的 Web 服务时，一定得保证它安全可靠。

其次，要确定 JSONP 请求是否失败并不容易。虽然 HTML5 给<script>元素新增了一个 onerror事件处理程序，但目前还没有得到任何浏览器支持。为此，开发人员不得不使用计时器检测指定时间内是否接收到了响应。

第三节 实现SSO

Maven+Idea

这里是采用Redis+Cookie实现单点登录

没有使用框架，只是通过Servlet+JSONP实现

3.1 Servlet代码

3.1.1 Login

登录，这里没有使用数据库，仅仅是模拟登录效果，只有传递用户名就权当登录成功

```
@WebServlet("/login.d")
public class Login extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //super.doGet(req, resp);
        String ck=CookieUtils.getCookieValue(req,"p2ptoken");
        if(ck!=null && ck.length()>0){
            RedisUtils.setTime("p2ptoken:"+ck,300);
        }else {
            String name=req.getParameter("un");
            String token=UUID.randomUUID().toString();
            RedisUtils.set("p2ptoken:"+ token,name,300);
            CookieUtils.setCookie(req,resp,"p2ptoken",token,300);
        }
        resp.sendRedirect("index.jsp");
    }
}
```

3.1.2 CheckToken

根据当前的Cookie获取存储的令牌对应的用户信息

```
@WebServlet("/checktoken.d")
public class CheckToken extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
       // super.doGet(req, resp);
        String ck=CookieUtils.getCookieValue(req,"p2ptoken");
        System.out.println(ck+"----->"+RedisUtils.get("p2ptoken:"+ck));
        String callback = req.getParameter("callback");
        String json= "
{'id':'"+ck+"','name':'"+RedisUtils.get("p2ptoken:"+ck)+"'}";
```

```
        System.err.println("返回: "+json);
        resp.getWriter().print(callback+"("+json+")");
    }
}
```

3.1.3 LoginOut

退出登录

```
@WebServlet("/loginout.d")
public class LoginOut  extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //super.doGet(req, resp);
        String ck=CookieUtils.getCookieValue(req,"p2ptoken");
        if(ck!=null && ck.length()>0){
            //失效
            RedisUtils.setTime("p2ptoken:"+ck,0);
            CookieUtils.setCookie(req,resp,"p2ptoken","",0);
        }
        resp.sendRedirect("index.jsp");
    }
}
```

3.1.4 CookieUtils

封装的操作Cookie的类

```
public final class CookieUtils {

    /**
     * 得到Cookie的值, 不编码
     *
     * @param request
     * @param cookieName
     * @return
     */
    public static String getCookieValue(HttpServletRequest request, String
cookieName) {
        return getCookieValue(request, cookieName, false);
    }

    /**
     * 得到Cookie的值,
     *
     * @param request
     * @param cookieName
```

```
     * @return
     */
    public static String getCookieValue(HttpServletRequest request, String
cookieName, boolean isDecoder) {
        Cookie[] cookieList = request.getCookies();
        if (cookieList == null || cookieName == null) {
            return null;
        }
        String retValue = null;
        try {
            for (int i = 0; i < cookieList.length; i++) {
                if (cookieList[i].getName().equals(cookieName)) {
                    if (isDecoder) {
                        retValue = URLDecoder.decode(cookieList[i].getValue(),
"UTF-8");
                    } else {
                        retValue = cookieList[i].getValue();
                    }
                    break;
                }
            }
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return retValue;
    }

    /**
     * 得到Cookie的值,
     *
     * @param request
     * @param cookieName
     * @return
     */
    public static String getCookieValue(HttpServletRequest request, String
cookieName, String encodeString) {
        Cookie[] cookieList = request.getCookies();
        if (cookieList == null || cookieName == null) {
            return null;
        }
        String retValue = null;
        try {
            for (int i = 0; i < cookieList.length; i++) {
                if (cookieList[i].getName().equals(cookieName)) {
                    retValue = URLDecoder.decode(cookieList[i].getValue(),
encodeString);
                    break;
                }
            }
```

```
        } catch (UnsupportedEncodingException e) {
             e.printStackTrace();
        }
        return retValue;
    }

    /**
     * 设置Cookie的值 不设置生效时间默认浏览器关闭即失效,也不编码
     */
    public static void setCookie(HttpServletRequest request, HttpServletResponse
response, String cookieName,
            String cookieValue) {
        setCookie(request, response, cookieName, cookieValue, -1);
    }

    /**
     * 设置Cookie的值 在指定时间内生效,但不编码
     */
    public static void setCookie(HttpServletRequest request, HttpServletResponse
response, String cookieName,
            String cookieValue, int cookieMaxage) {
        setCookie(request, response, cookieName, cookieValue, cookieMaxage,
false);
    }

    /**
     * 设置Cookie的值 不设置生效时间,但编码
     */
    public static void setCookie(HttpServletRequest request, HttpServletResponse
response, String cookieName,
            String cookieValue, boolean isEncode) {
        setCookie(request, response, cookieName, cookieValue, -1, isEncode);
    }

    /**
     * 设置Cookie的值 在指定时间内生效, 编码参数
     */
    public static void setCookie(HttpServletRequest request, HttpServletResponse
response, String cookieName,
            String cookieValue, int cookieMaxage, boolean isEncode) {
        doSetCookie(request, response, cookieName, cookieValue, cookieMaxage,
isEncode);
    }

    /**
     * 设置Cookie的值 在指定时间内生效, 编码参数(指定编码)
     */
    public static void setCookie(HttpServletRequest request, HttpServletResponse
response, String cookieName,
```

```
            String cookieValue, int cookieMaxage, String encodeString) {
        doSetCookie(request, response, cookieName, cookieValue, cookieMaxage,
encodeString);
    }

    /**
     * 删除Cookie带cookie域名
     */
    public static void deleteCookie(HttpServletRequest request,
HttpServletResponse response,
            String cookieName) {
        doSetCookie(request, response, cookieName, "", -1, false);
    }

    /**
     * 设置Cookie的值，并使其在指定时间内生效
     *
     * @param cookieMaxage cookie生效的最大秒数
     */
    private static final void doSetCookie(HttpServletRequest request,
HttpServletResponse response,
            String cookieName, String cookieValue, int cookieMaxage, boolean
isEncode) {
        try {
            if (cookieValue == null) {
                cookieValue = "";
            } else if (isEncode) {
                cookieValue = URLEncoder.encode(cookieValue, "utf-8");
            }
            Cookie cookie = new Cookie(cookieName, cookieValue);
            if (cookieMaxage > 0)
                cookie.setMaxAge(cookieMaxage);
            if (null != request) {// 设置域名的cookie
                String domainName = getDomainName(request);
                System.out.println(domainName);
                if (!"localhost".equals(domainName)) {
                    cookie.setDomain(domainName);
                }
            }
            cookie.setPath("/");
            response.addCookie(cookie);
        } catch (Exception e) {
             e.printStackTrace();
        }
    }

    /**
     * 设置Cookie的值，并使其在指定时间内生效
     *
```

```
     * @param cookieMaxage cookie生效的最大秒数
     */
    private static final void doSetCookie(HttpServletRequest request,
HttpServletResponse response,
            String cookieName, String cookieValue, int cookieMaxage, String
encodeString) {
        try {
            if (cookieValue == null) {
                cookieValue = "";
            } else {
                cookieValue = URLEncoder.encode(cookieValue, encodeString);
            }
            Cookie cookie = new Cookie(cookieName, cookieValue);
            if (cookieMaxage > 0)
                cookie.setMaxAge(cookieMaxage);
            if (null != request) {// 设置域名的cookie
                String domainName = getDomainName(request);
                System.out.println(domainName);
                if (!"localhost".equals(domainName)) {
                    cookie.setDomain(domainName);
                }
            }
            cookie.setPath("/");
            response.addCookie(cookie);
        } catch (Exception e) {
             e.printStackTrace();
        }
    }

    /**
     * 得到cookie的域名
     */
    private static final String getDomainName(HttpServletRequest request) {
        String domainName = null;

        String serverName = request.getRequestURL().toString();
        if (serverName == null || serverName.equals("")) {
            domainName = "";
        } else {
            serverName = serverName.toLowerCase();
            serverName = serverName.substring(7);
            final int end = serverName.indexOf("/");
            serverName = serverName.substring(0, end);
            final String[] domains = serverName.split("\\.");
            int len = domains.length;
            if (len > 3) {
                // www.xxx.com.cn
                domainName = "." + domains[len - 3] + "." + domains[len - 2] + "."
+ domains[len - 1];
```

```
            } else if (len <= 3 && len > 1) {
                // xxx.com or xxx.cn
                domainName = "." + domains[len - 2] + "." + domains[len - 1];
            } else {
                domainName = serverName;
            }
        }

        if (domainName != null && domainName.indexOf(":") > 0) {
            String[] ary = domainName.split("\\:");
            domainName = ary[0];
        }
        return domainName;
    }

}
```

3.1.5 RedisUtils

Redis的工具类

```
public class RedisUtils {
    private static JedisPool pool;
   static {
       pool=new JedisPool("10.211.55.15",6379);

   }
   public static void set(String key,String value,int time){
       Jedis jedis=pool.getResource();
       jedis.auth("1714");
       jedis.set(key,value);
       jedis.expire(key,time);
       jedis.close();
   }
   public  static void setTime(String key,int time){
       Jedis jedis=pool.getResource();
       jedis.auth("1714");
       jedis.expire(key,time);
       jedis.close();
   }
   public static String get(String key){
        Jedis jedis=pool.getResource();
        jedis.auth("1714");
       String r=jedis.get(key);
       jedis.close();
       return  r;
```

```
    }
}
```

3.2 前端页面

3.2.1 index.html

主页面

```
<html>
<head>
    <title>主页</title>
    <script type="application/javascript" src="js/jquery-3.2.1.min.js"></script>
    <script type="application/javascript" src="js/jquery.cookie.js"></script>
    <script type="application/javascript">

        var ssologin = {
            checkLogin : function(){
                var ton = $.cookie("p2ptoken");
                console.log("令牌: "+ton);
                if(!ton){
                    location.href="login.jsp";
                    return ;
                }
                //第一种方式
                $.ajax({
                    url : "http://localhost:8087/checktoken.d?token=" + ton,
                    dataType : "jsonp",
                    type : "GET",
                    success :  function getData(result) {
//                        if(result.length>0 ){
                            var html = result.name + ", 欢迎来到SSO系统! <a
href=\"http://localhost:8087/loginout.d\">[退出]</a>";
                            $("#loginbar").html(html);
                    }
                });
            }
        }

        $(function () {
                ssologin.checkLogin();
        });
    </script>
</head>

<body>

<h2>欢迎访问主页</h2>
```

```
<input type="button" value="获取Token" onclick="ssologin.checkLogin()">
<div id="loginbar" ></div>
</body>
</html>
```

3.2.2 login.html

登录页面

```
<html>
<head>
    <title>单点登录</title>
    <script type="application/javascript" src="js/jquery-3.2.1.min.js"></script>
    <script type="application/javascript" src="js/jquery.cookie.js"></script>
</head>
<body>
<div id="loginbar"></div>
<div id="loginf" >
<form action="http://localhost:8087/login.d">
    <input name="un" placeholder="请输入账号">
    <br/>
    <input name="pw" placeholder="请输入密码" type="password"><br/>
    <input type="submit" value="登录">
</form>
</div>
</body>
</html>
```

3.3 配置文件

3.3.1 pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <parent>
        <artifactId>SSO_Case</artifactId>
        <groupId>xph</groupId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>SSOWeb</artifactId>
    <packaging>war</packaging>
    <name>SSOWeb Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <dependencies>
```

```
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
       <!-- https://mvnrepository.com/artifact/redis.clients/jedis -->
        <dependency>
            <groupId>redis.clients</groupId>
            <artifactId>jedis</artifactId>
            <version>2.9.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api --
>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <finalName>SSOWeb</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>
                    <port>8087</port>
                    <path>/</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

3.3.2 web.xml

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>SSOWeb</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

3.4 实现单点

因为要实现单点登录所以根据再创建2个项目，分别将前端页面拷贝到对应项目中，无需Servlet代码，让另外的项目跨域访问SSOWeb项目，pom文件的发布端口改的各不相同

3.4.1 SSOWeb项目的pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <parent>
        <artifactId>SSO_Case</artifactId>
        <groupId>xph</groupId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>SSOWeb</artifactId>
    <packaging>war</packaging>
    <name>SSOWeb Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
         <!-- https://mvnrepository.com/artifact/redis.clients/jedis -->
        <dependency>
            <groupId>redis.clients</groupId>
            <artifactId>jedis</artifactId>
            <version>2.9.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api --
>
        <dependency>
```

```
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <finalName>SSOWeb</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>
                    <port>8087</port>
                    <path>/</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

3.4.2 OAWeb的pom

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>SSO_Case</artifactId>
        <groupId>xph</groupId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>OaWeb</artifactId>
    <packaging>war</packaging>

    <build>

    <finalName>SSOWeb</finalName>
    <plugins>
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <configuration>
                <port>8083</port>
                <path>/</path>
            </configuration>
```

```
        </plugin>
    </plugins>
    </build>
</project>
```

3.4.3 ERPWeb的pom

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>SSO_Case</artifactId>
        <groupId>xph</groupId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>ErpWeb</artifactId>
    <packaging>war</packaging>

    <build>

        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>
                    <port>8084</port>
                    <path>/</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

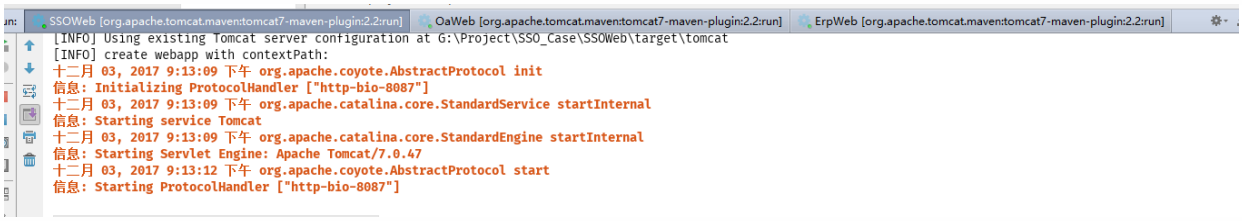
3.5 使用tomcat7:run启动

分别启动三个程序，进行单点登录的测试

最终的测试目标：

三个项目任意一个登录都可以共享

三个项目任意一个退出都可以退出


SSOWeb [org.apache.tomcat.maven:tomcat7-maven-plugin:2.2:run]
OaWeb [org.apache.tomcat.maven:tomcat7-maven-plugin:2.2:run]
ErpWeb [org.apache.tomcat.maven:tomcat7-maven-plugin:2.2:run]
[INFO] Using existing Tomcat server configuration at G:\Project\SSO_Case\SSOWeb\target\tomcat
[INFO] create webapp with contextPath:
十二月 03, 2017 9:13:09 下午 org.apache.coyote.AbstractProtocol init
信息: Initializing ProtocolHandler ["http-bio-8087"]
十二月 03, 2017 9:13:09 下午 org.apache.catalina.core.StandardService startInternal
信息: Starting service Tomcat
十二月 03, 2017 9:13:09 下午 org.apache.catalina.core.StandardEngine startInternal
信息: Starting Servlet Engine: Apache Tomcat/7.0.47
十二月 03, 2017 9:13:12 下午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8087"]