[今日课程大纲]

SSO 简介

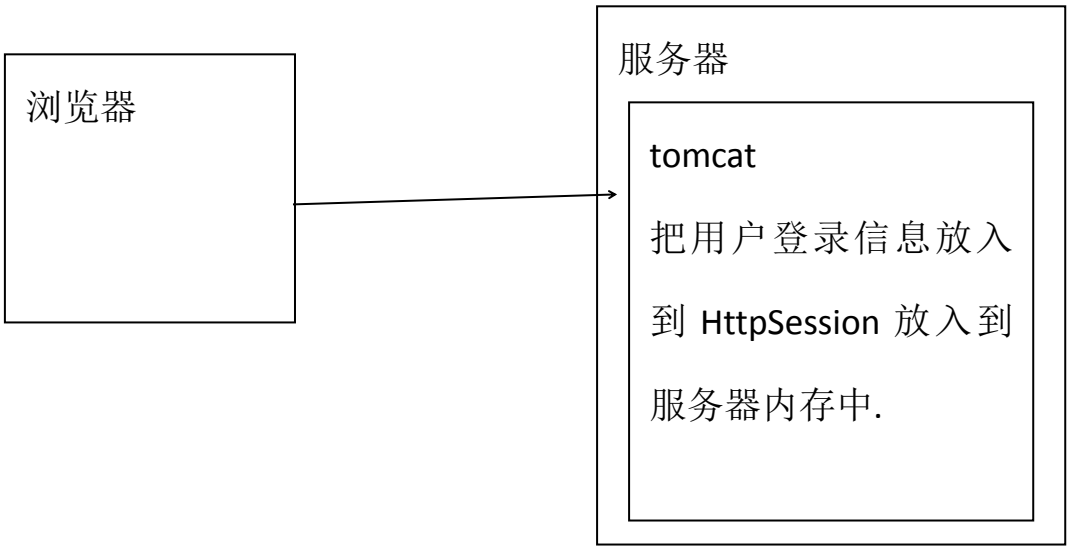
复习 HttpSession 和 Cookie

搭建 ego-passport 并编写代码

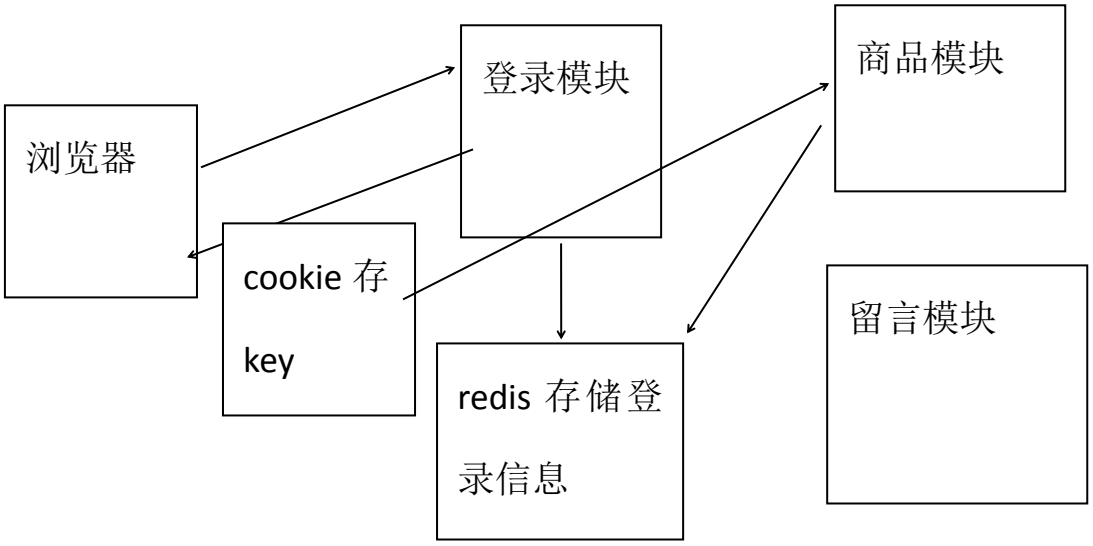
[知识点详解]

一. SSO 简介

1. SSO 英文名称(Single Sign On)
2. SSO 中文名称:单点登录
3. 单点登录解释:
   3.1 一次登录,让其他项目共享登录状态.
   3.2 本质:使用特定技术在分布式项目中模拟 HttpSession 功能.
      3.2.1 技术选型:Redis+Cookie 模拟 HttpSession 功能
4. 传统项目和分布式项目登录功能对比
   4.1 传统项目



4.2 分布式项目



二. Cookie 复习

1. 解释:客户端存值技术.

1.1 存储位置:客户端浏览器.

1.2 作用:存值

1.3 存值类型: 只能存储字符串.

2. Cookie 运行原理:

2.1 当浏览器输入 URL 访问服务器时会自动携带所有有效 Cookie(时间内,指定路径内,指定域名内),Tomcat 接收请求后会把 Cookie 放入到 HttpServletRequest 中,在代码中通过 request 对象获取到 Cookie 的内容.

2.2 服务器内容可以产生 Cookie 内容,需要放入到响应对象中响应给客户端浏览器.(要求跳转类型为重定向.),客户端浏览器接收响应内容后会把 Cookie 内容存储到指定文件夹内容.

3. 产生 cookie

```
@RequestMapping("demo")
   public String goIndex(HttpServletResponse response){
      Cookie c = new Cookie("key", "value");
      //1. 默认实现和 HttpSession 相同.
      //设置 cookie 存活时间,单位秒
//    c.setMaxAge(10);
      //2. 默认存储路径为 path=/

      //设置哪个目录下资源能访问
      c.setPath("/");
      //3. 域名和当前项目的域名相同
```

```
        c.setDomain(".bjsxt.com");
        response.addCookie(c);
        return "redirect:/index.jsp";
    }
```

4. 获取 cookie

```
<%
    Cookie [] cs = request.getCookies();
    if(cs!=null){
        for(Cookie c :cs){

    out.print("key:"+c.getName()+",value:"+c.getValue()
+"<br/>");
        }
    }else{
        out.println("没有 cookie");
    }
%>
```

三. HttpSession 运行原理.

1. 当客户端浏览器访问服务器时,服务器接收请求后会判断请求中是否在 Cookie 包含 JSESSIONID.

2. 如果包含 JSESSIONID 把对应值取出，做为 Key 从全局 Map<String,Session>对象往出取 Session 对象.

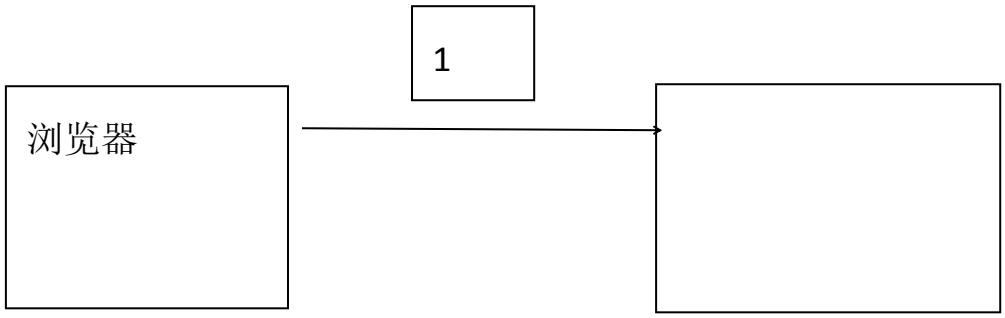
3. 如果有对应的 key,把 Session 对象取出.根据自己业务添加操作.

4. 如果请求对象中 JSESSIONID 在全局 Map<String,Session>没有或请求对象中 Cookie 没有 JSESSSIONID ,会执行新建 Session 步骤

5. Tomcat 会新建(new)一个 Sesssion 对象,同时产生一个 UUID,把 UUID 做为 Map 的 key,新建的 Session 对象做为 Value,还会把 UUID 放入到 Cookie 做为 value,value 对应的 Key 是 JSESSIONID

四使用 Redis+Cookie 模拟 Session 步骤:

1. 流程图



2. 步骤:

2.1 步骤 1:第一次请求时 Cookie 中没有 token

2.2 产生一个 UUID,把”token”做为 Cookie 的 key,UUID 做为 Cookie

的 value

2.3 如果希望类似往 Session 存储内容,直接把 UUID 当作 redis 的 key(个别需要还需要考虑 key 重复问题.),把需要存储的内容做为 value.

2.4 把 cookie 对象响应给客户端浏览器.

四. 登录业务

1. 登录时,跳转到 ego-passport 的 login.jsp
2. 从哪个跳转到登录页面,登录成功后跳转回哪里

   2.1 借助请求头中 Referer 获取到从哪里跳转过来
3. 实现 SSO 模拟 Session 功能.
4. 当用户点击”加入购物车”按钮

   4.1 如果已经登录,进入到购物车页面

   4.2 如果没有登录,进入登录页面

五. 实现登录功能

1. 在 ego-service 新建 TbUserDubboServiceImpl 及实现类

```
public interface TbUserDubboService {
    /**
     * 根据用户名和密码查询登录
     * @param user
     * @return
```

```
    */
   TbUser selByUser(TbUser user);
}
```

```
public class TbUserDubboServiceImpl implements
TbUserDubboService {
   @Resource
   private TbUserMapper tbUserMapper;
   @Override
   public TbUser selByUser(TbUser user) {
      TbUserExample example = new TbUserExample();

   example.createCriteria().andUsernameEqualTo(user.ge
tUsername()).andPasswordEqualTo(user.getPassword());
      List<TbUser> list =
tbUserMapper.selectByExample(example);
      if(list!=null&&list.size()>0){
         return list.get(0);
      }
      return null;
   }
}
```

2. 在 ego-service-impl 的 applicationContext-dubbo.xml 注解接口

```
<!-- 用户 -->
   <dubbo:service
interface="com.ego.dubbo.service.TbUserDubboService"
ref="tbUserDubboServiceImpl"></dubbo:service>
   <bean                        id="tbUserDubboServiceImpl"
class="com.ego.dubbo.service.impl.TbUserDubboServiceI
mpl"></bean>
```

3. 在 ego-commons 的 EgoResult 中添加 String msg;属性

```
public class EgoResult {
   private int status;
   private Object data;
   private String msg;
```

4. 在 ego-passport 中新建 TbUserService 及实现类

```
public interface TbUserService {
   /**
    * 登录
    * @param user
    * @return
    */
   EgoResult login(TbUser user);
}
```

```
@Service
```

```
public class TbUserServiceImpl implements TbUserService
{
   @Reference
   private TbUserDubboService tbUserDubboServiceImpl;
   @Override
   public EgoResult login(TbUser user) {
      EgoResult er = new EgoResult();
      TbUser userSelect =
tbUserDubboServiceImpl.selByUser(user);
      if(userSelect!=null){
         er.setStatus(200);
      }else{
         er.setMsg("用户名和密码错误");
      }
      return er;
   }

}
```

5. 在 ego-passport 中新建控制器,提供 2 个方法

```
@Controller
public class TbUserController {
   @Resource
```

```
    private TbUserService tbUserServiceImpl;
    /**
     * 显示登录页面
     * @return
     */
    @RequestMapping("user/showLogin")
    public String showLogin(@RequestHeader("Referer")
String url,Model model){
        model.addAttribute("redirect", url);
        return "login";
    }
    /**
     * 登录
     * @param user
     * @return
     */
    @RequestMapping("user/login")
    @ResponseBody
    public EgoResult login(TbUser user){
        return tbUserServiceImpl.login(user);
    }
}
```

五. 实现 SSO 核心功能.

1. 在 ego-commons 中放入工具类 CookitUtils

2. 在 ego-passport 的 TbUserServiceImpl 的 login 方法修改如下

```
/**
    * 登录
    * @param user
    * @return
    */
   EgoResult    login(TbUser    user,HttpServletRequest
request,HttpServletResponse response);
```

```
@Resource
   private JedisDao jedisDaoImpl;
   @Override
   public EgoResult login(TbUser
user,HttpServletRequest request,HttpServletResponse
response) {
      EgoResult er = new EgoResult();
      TbUser userSelect =
tbUserDubboServiceImpl.selByUser(user);
```

```
        if(userSelect!=null){
            er.setStatus(200);
            //当用户登录成功后把用户信息放入到 redis 中
            String key = UUID.randomUUID().toString();
            jedisDaoImpl.set(key,
JsonUtils.objectToJson(userSelect));
            jedisDaoImpl.expire(key, 60*60*24*7);
            //产生 Cookie
            CookieUtils.setCookie(request, response,
"TT_TOKEN", key, 60*60*24*7);
        }else{
            er.setMsg("用户名和密码错误");
        }
        return er;
    }
```

六. 实现通过 token 获取用户信息

1. 在 ego-passport 的 TbUserService 及实现类添加

```
/**
     * 根据 token 查询用户信息
     * @param token
     * @return
```

```
    */
    EgoResult getUserInfoByToken(String token);
```

```
@Override
    public EgoResult getUserInfoByToken(String token) {
        EgoResult er = new EgoResult();
        String json = jedisDaoImpl.get(token);
        if(json!=null&&!json.equals("")){
            TbUser tbUser = JsonUtils.jsonToPojo(json,
TbUser.class);
            //可以把密码清空
            tbUser.setPassword(null);
            er.setStatus(200);
            er.setMsg("OK");
            er.setData(tbUser);
        }else{
            er.setMsg("获取失败");
        }

        return er;
    }
```

2. 在 ego-passport 中 TbUserController 添加

```
/**
    * 通过 token 获取用户信息
    * @param token
    * @param callback
    * @return
    */
   @RequestMapping("user/token/{token}")
   @ResponseBody
   public Object getUserInfo(@PathVariable String
token,String callback){
      EgoResult er =
tbUserServiceImpl.getUserInfoByToken(token);
      if(callback!=null&&!callback.equals("")){
         MappingJacksonValue mjv = new
MappingJacksonValue(er);
         mjv.setJsonpFunction(callback);
         return mjv;
      }
      return er;

   }
```

七. 退出功能

1. 在 ego-passport 中 TbUserService 及实现类添加

```
/**
    * 退出
    * @param token
    * @param request
    * @param response
    * @return
    */
   EgoResult   logout(String   token,HttpServletRequest
request,HttpServletResponse response);
```

```
@Override
   public EgoResult logout(String token,
HttpServletRequest request, HttpServletResponse
response) {
      Long index = jedisDaoImpl.del(token);
      CookieUtils.deleteCookie(request, response,
"TT_TOKEN");
      EgoResult er = new EgoResult();
      er.setStatus(200);
      er.setMsg("OK");
      return er;
```

```
    }
```

2. 在 ego-passport 的 TbUserController 中添加

```
/**
     * 退出
     * @param token
     * @param callback
     * @return
     */
    @RequestMapping("user/logout/{token}")
    @ResponseBody
    public Object logout(@PathVariable String
token,String callback,HttpServletRequest
request,HttpServletResponse response){
        EgoResult er =
tbUserServiceImpl.logout(token,request,response);
        if(callback!=null&&!callback.equals("")){
            MappingJacksonValue mjv = new
MappingJacksonValue(er);
            mjv.setJsonpFunction(callback);
            return mjv;
        }
        return er;
```

```
}
```