kisso 帮助文档

1、简介

kisso = cookie sso

基于 Cookie 的 SSO 中间件，它是一把快速开发 java Web 登录系统（SSO）的瑞士军刀。

1、支持单点登录

2、支持登录Cookie缓存

3、支持防止 xss攻击, SQL注入，脚本注入

4、支持 Base64 / MD5 / AES / PBE / RSA 算法

5、支持浏览器客户端校验

6、支持Cookie参数配置及扩展

7、支持跨域登录，模拟登录

8、支持在线人数统计

9、支持生成动态图片验证码

10、支持 app 移动端 api 服务验证，采用微信公众平台 api 验证机制认证

11、支持踢出指定登录用户

等等。。。。

支付宝扫码捐赠

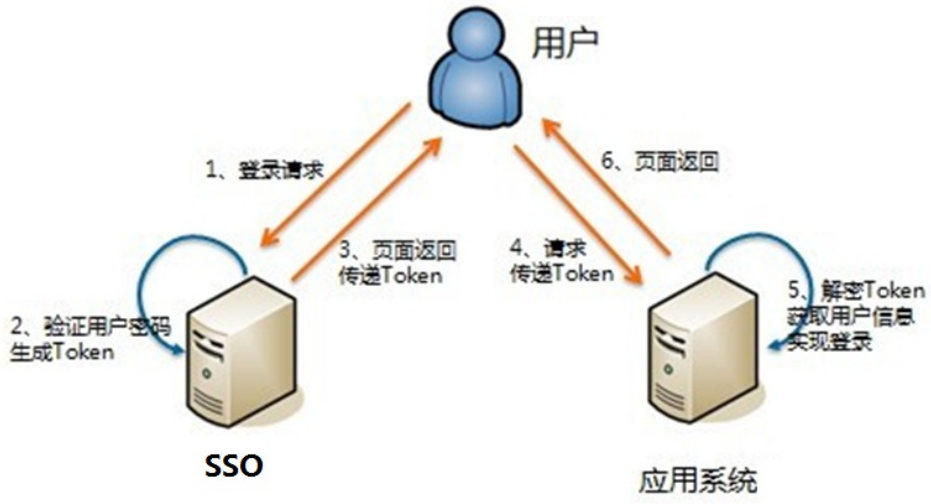
微信扫码捐赠

kisso 作者：青苗 jobob@qq.com

http://baomidou.com/

http://git.oschina.net/juapk/kisso

2、kisso 是什么原理与 cas 区别 ？



一、kisso 原理说明：

kisso 采用的是加密会话 cookie 机制实现单点登录 SSO 服务，具备“无状态”、“分散验证”等特性。

1、session 存放在服务器端，cookie 存放在客户端，存在 2 种状态：“第一种：持久 cookie 具有时效性，以文件的形式存放在客户机硬盘中，
时间一到生命周期结束自动被删除。第二种：临时 cookie 又叫会话 cookie 放在浏览器内存

中，浏览器关闭生命周期结束自动失效”。

2、单纯不做任何改变而言 session 更安全，如果 cookie 采取各种安全保护措施，此时的 cookie 一样安全。

3、cookie 轻松实现分布式服务部署，单点登录跨域访问等问题，换成 session 需要处理 session 复制及各种问题实现困难。

二、 kisso 与 cas 区别：

1、cas 是单点登录系统，它给你制定好了规则按照它的要求做就可以，配置（复杂）好一切即可实现单点登录。

2、kisso 是一个中间件，提供 cookie 搭建 java web sso 的组件式解决方案。
你不管使用任何架构都可以使用它，就像一个 U 盘需要使用就插入、不用就拔掉。

3、cas 集中验证，所有请求都由 cas 集中验证，缺点cas服务压力巨大。kisso 分散验证，由各个系统验证 cookie 合法性，缺点秘钥要保护好。

3、依赖包 jars 及 demo

核心 jar 文件 kisso-xxx.jar 下载地址：

http://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22com.baomidou%22%20AND%20a%3A%22kisso%22

解析 Token 依赖库，默认 fastjson-xxx.jar 下载地址：

http://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22com.alibaba%22%20AND%20a%3A%22fastjson%22

加密 Token 依赖库，默认 bcprov-xxx.jar 下载地址：

http://search.maven.org/#search%7Cga%7C1%7Cbcprov

maven 依赖

```
<!-- kisso begin -->
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>kisso</artifactId>
    <version>最新版本号</version>
</dependency>
<dependency>
    <groupId>org.bouncycastle</groupId>
    <artifactId>bcprov-jdk14</artifactId>
    <version>1.50</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.6</version>
</dependency>
<!-- kisso end -->
```

```
注意！

1、如果想自己实现 Token 解析，实现 com.baomidou.kisso.common.parser.SSOParser 类
（解析接口类），修改配置文件  sso.properties 属性 sso.parser.class
此时 kisso 会调用您定义的解析类，不需要再依赖 fastjson 库。


2、自己实现 Token 加密，实现 com.baomidou.kisso.common.encrypt.SSOEncrypt
 类（加密解密接口类），修改配置文件  sso.properties 属性
sso.encrypt.class  不在需要依赖 bcprov 库。
```

kisso-oauth2-demo oauth2 演示 demo 由于 apache-oltu-oauth2 很灵活 kisso 只做辅助支持，不重新造轮子。

kisso_ApiServer 移动端 api 演示 demo 采用微信验证机制， 当然你也可以不采用该方法、直接在移动端维护 cookie 方式访问

api

第一种、同一个根域名不同子域名，比如 my.baomidou.com 、 sso.baomidou.com 、other.baomidou.com 此时配置 domain 只需要配置 .baomidou.com 即可。 查看普通 demo ：

kisso_Jfinal jfinal 演示 demo

kisso_SpringMvc spring 演示 demo

第二种、完全不同的域名，比如 sso.baomidou.com git.oschina.net 此时比较复杂 kisso 采用的是 rsa 加密询问验证（较复杂）查看跨域 demo:

kisso_crossdomain 跨域演示 demo

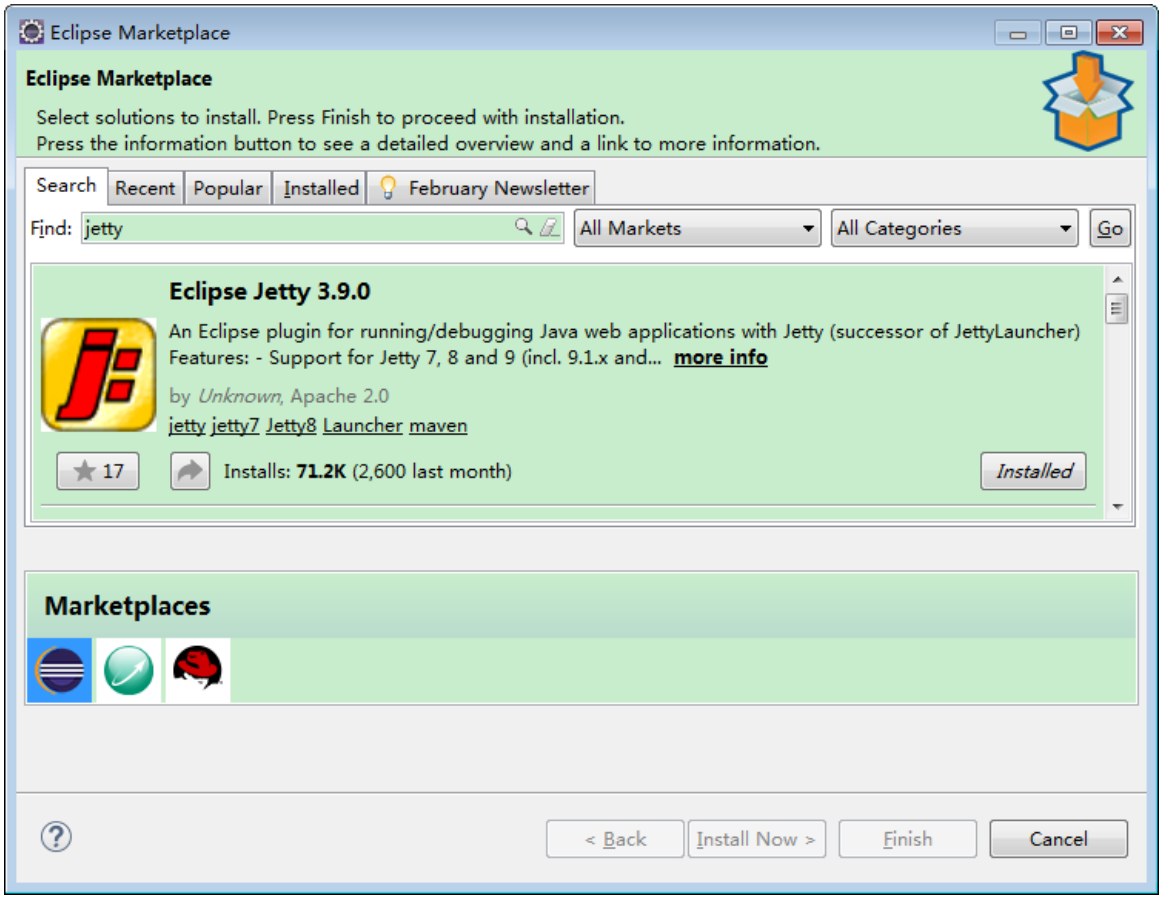
kisso + shiro 集成 demo【注意 dev-kisso 分支】
http://git.oschina.net/wangzhixuan/spring-shiro-training/tree/dev-kisso/

4、如何运行 demo

一、安装 eclipse 及 jetty 插件（tomcat 也可以，这里推荐 jetty 因为觉得它好用）

1、eclipse 下载地址：http://www.eclipse.org/downloads/

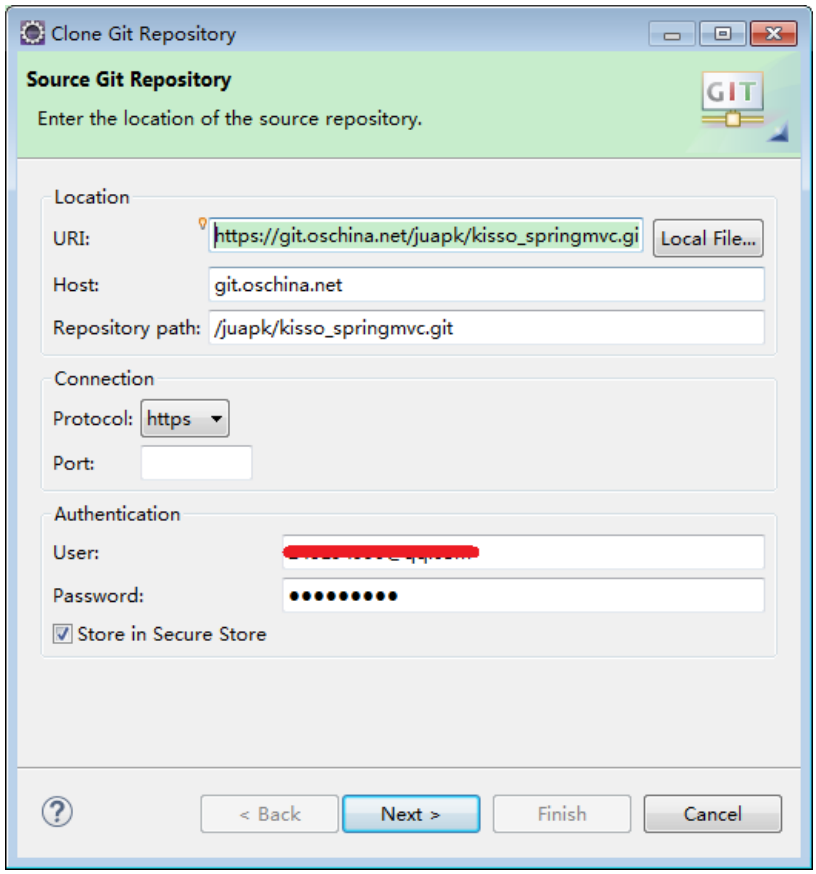
2、安装 jetty 插件： eclipse -》 help -》Eclipse Maketplace... -》 find 搜 jetty 如图安装 jetty 插件 -》重启 eclipse



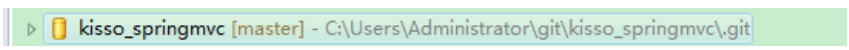
二、clone 演示 demo

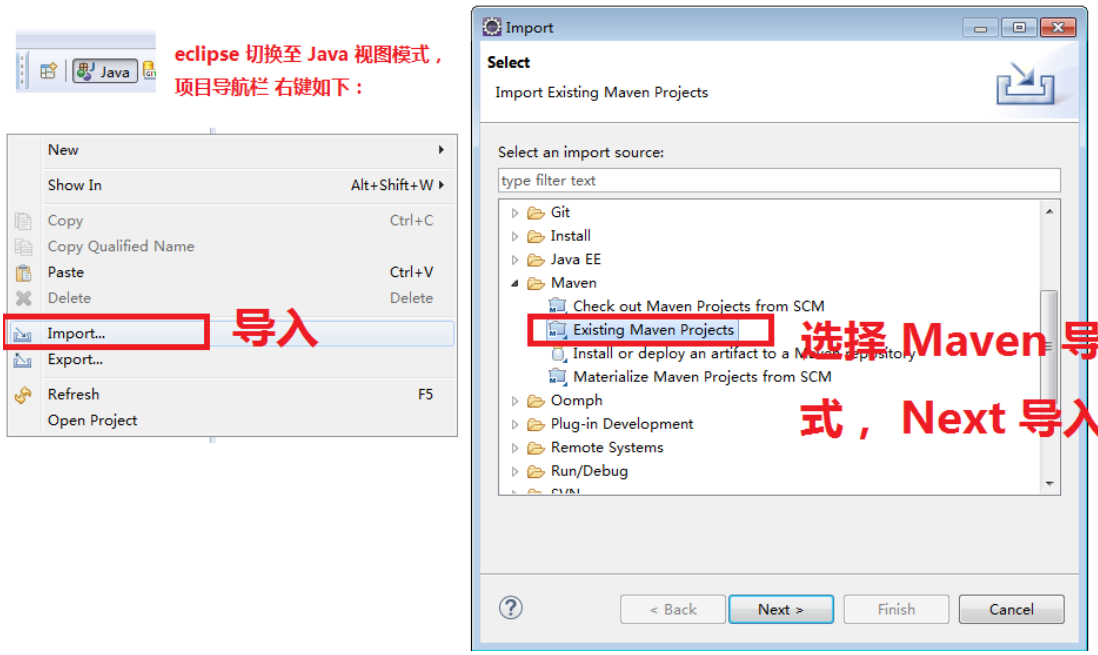
1、进入 demo git 页面，如图复制 git 地址

2、切换 eclipse git 界面，如图点击 Clone 弹出框 URI 粘贴上一步 Git 地址 Next -》Next -》Finsh



Clone 导入成功





三、配置访问 host

推荐一强大的工具： http://oldj.github.io/SwitchHosts/

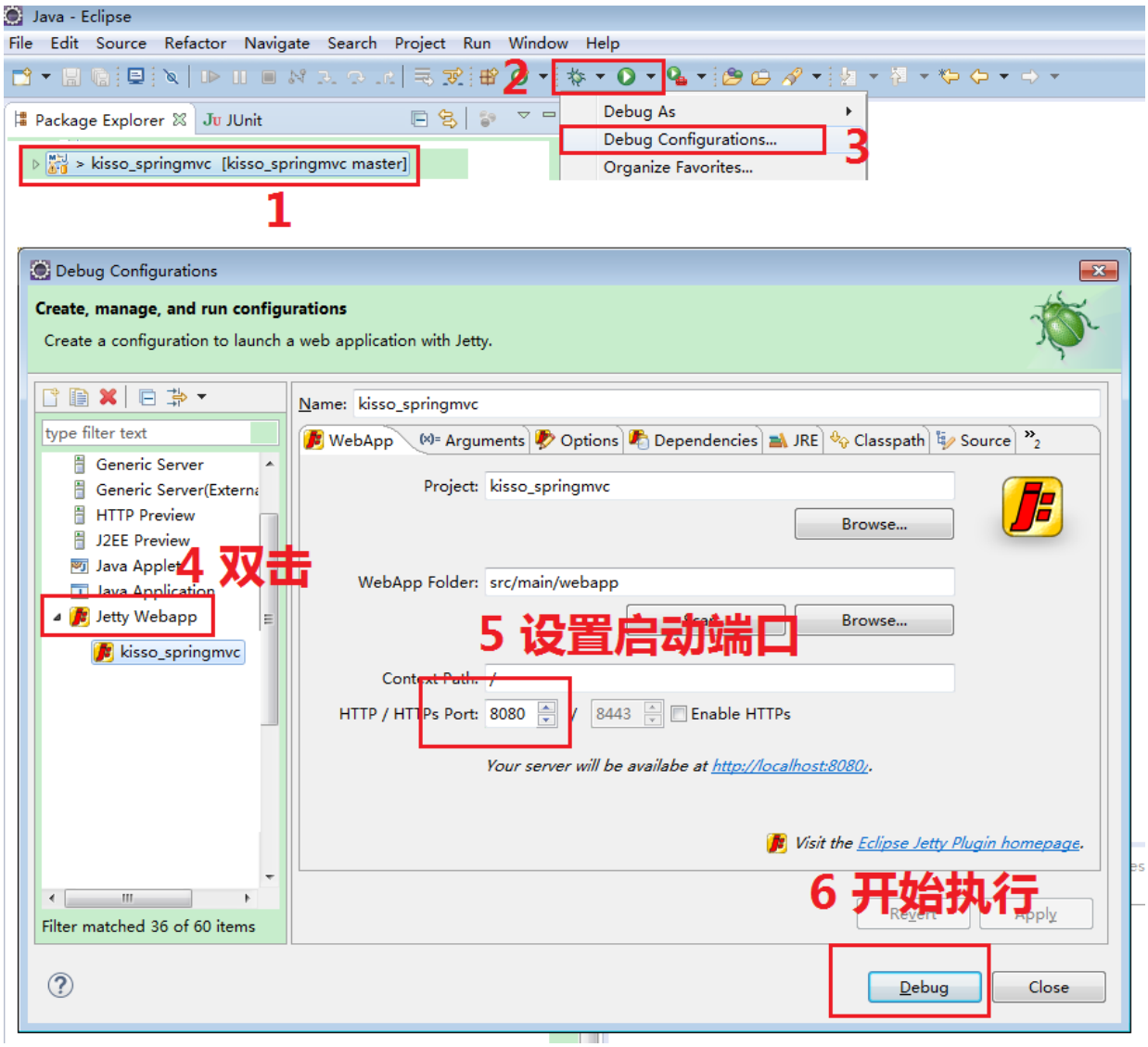
~~host 内容：设置完记得重启浏览器！！！~~

127.0.0.1 sso.test.com

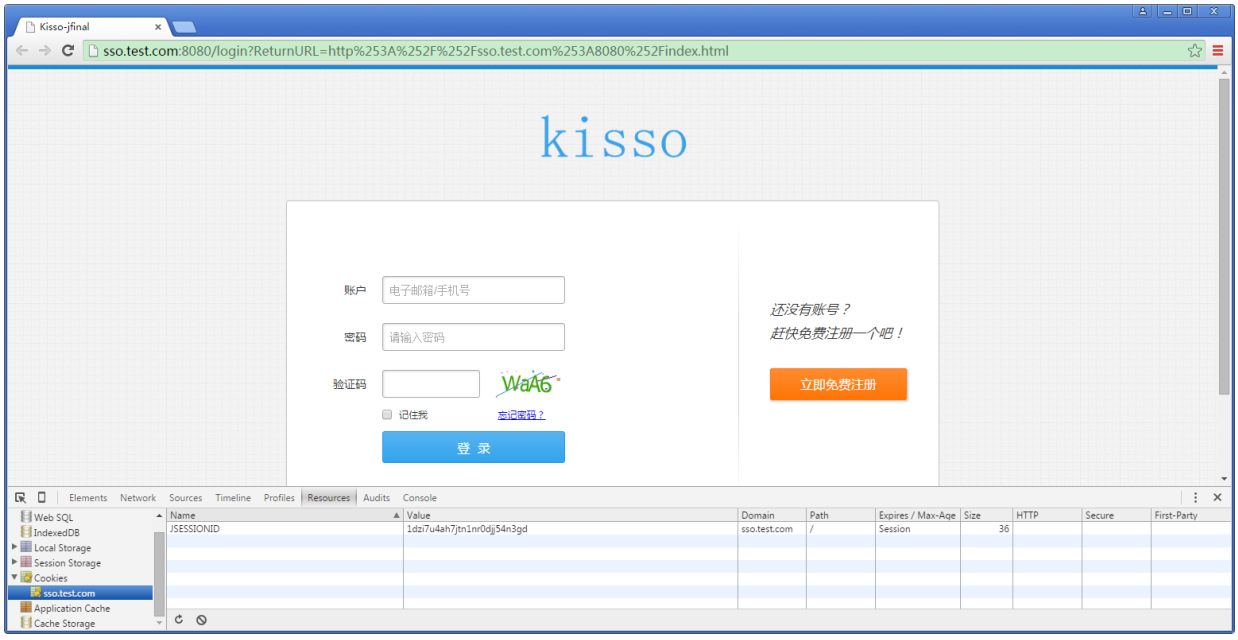
127.0.0.1 my.web.com

四、jetty 配置项目及启动

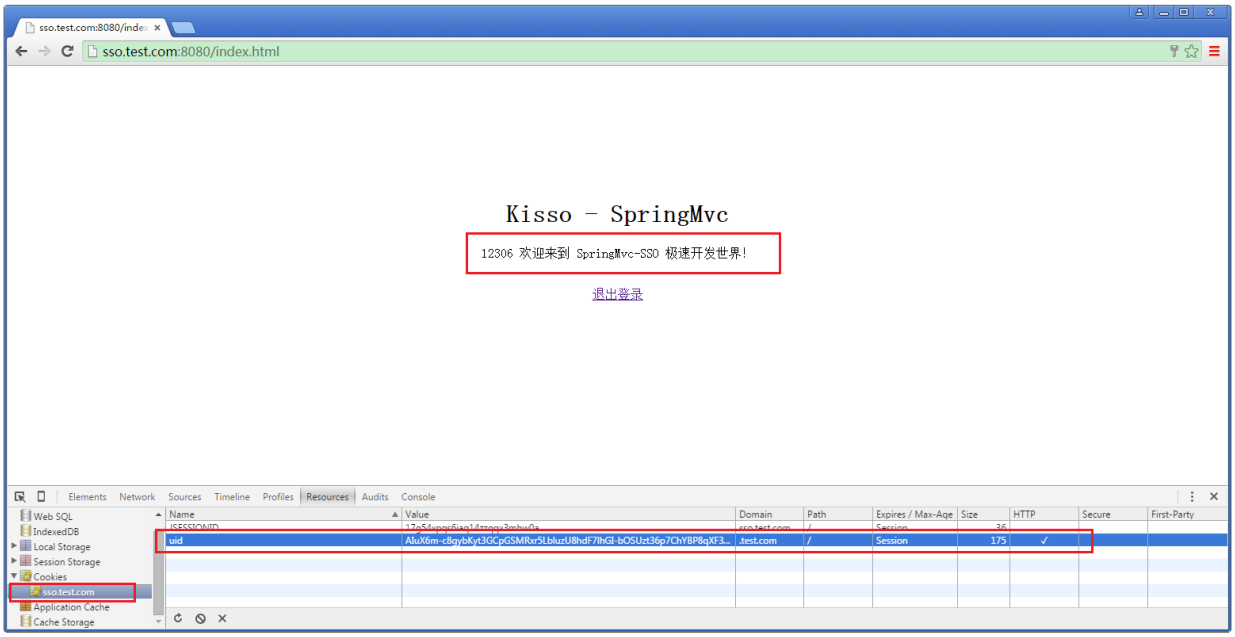
1、切换 eclipse git 界面，完成 如图 6 步， 启动 8080 端口



2、访问 sso.test.com:8080



3、 输入：用户 kisso 密码 123 进入系统如图发现 uid 登录 cookie



5、sso.properties 怎么去配置 ？

除了密钥、域名、必须修改，其他默认配置或根据需要选择配置即可。

配置说明：

---

sso.run.mode 模式配置，默认不带后缀为线上模式，模式设置：_dev_mode 开发，_test_mode 测试 ，_online_mode 生产
sso.encoding 编码格式： 默认 UTF-8
sso.secretkey 加密密钥

------ cookie 设置部分 ------
sso.cookie.name 名称，默认 uid
sso.cookie.secure 是否只能HTTPS访问，默认 false 【客户端配置可无】
sso.cookie.httponly 是否设置 httponly脚本无法访问，默认 true 【客户端配置可无】
sso.cookie.maxage 过期时间，默认 -1 关闭浏览器失效 【客户端配置可无】
sso.cookie.domain 所在域，请设置根域，如 .baomidou.com 【客户端配置可无】
sso.cookie.path 路径，默认 / 【客户端配置可无】
sso.cookie.browser 是否检查浏览器，默认 true
sso.cookie.checkip 是否检查登录IP，默认 false
sso.encrypt.class 自定义对称加密类，默认AES，自定义例如：com.baomidou.my.DES
sso.parser.class 自定义解析类，默认 FastJsonParser 【采用默认解析必须加入 fastjson 依赖】
sso.statistic.class 自定义在线人数统计类，必须实现 SSOStatistic 接口
sso.token.class 自定义票据，默认SSOToken，自定义例如：com.baomidou.bo.LoginToken

------ 缓存部分 ------
sso.cache.class 自定义缓存实现：com.baomidou.cache.RedisCache
sso.cache.expires 单位s秒，设置 -1永不失效，大于 0 失效时间

------ SSO 请求地址设置 ------
sso.login.url_online_mode 线上模式，登录地址：http://sso.testdemo.com/login.html
sso.login.url_dev_mode 开发模式，登录地址：http://localhost:8080/login.html

sso.logout.url_online_mode　　线上模式，退出地址：http://sso.testdemo.com/logout.html
sso.logout.url_dev_mode　　开发模式，退出地址：http://localhost:8080/logout.html
sso.param.returl　　重定向地址参数配置，默认：ReturnURL

------ Shiro 权限部分参数 ------
sso.permission.uri　　是否开启 uri 认证、默认 true

------ 跨域 cookie 设置部分，不开启跨域功能可不设置 ------
sso.role　　应用角色名（根据该值判断对应系统 RSA 公钥私钥 ）
sso.authcookie.secretkey　　加密密钥
sso.authcookie.name　　名称pid，请不要与登录 cookie 名称一致
sso.authcookie.maxage　　过期时间，默认 -1 关闭浏览器失效

6、kisso 启动配置

Servlet 方式初始化

----

kisso 启动 web.xml 配置, spring 支持 bean注入启动， **jfinal 支持插件启动。 具体查看具体 Demo**

、、

```
<!-- SSO 配置 -->
<context-param>
   <param-name>kissoConfigLocation</param-name>
   <param-value>classpath:properties/sso.properties</param-value>
</context-param>
<listener>
   <listener-class>com.baomidou.kisso.web.KissoConfigListener</listener-class>
</listener>
```

Servlet 方式 SSO 拦截器 web.xml 配置 【可选】

、、

```
<!-- SSOFilter use . -->
<filter>
   <filter-name>SSOFilter</filter-name>
   <filter-class>com.baomidou.kisso.web.filter.SSOFilter</filter-class>
   <init-param>
      <param-name>over.url</param-name>
      <param-value>/login.html</param-value>
   </init-param>
</filter>
<filter-mapping>
   <filter-name>SSOFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

Spring 方式初始化

----

kisso 使用 spring 方式 【选择这种方式，web.xml 就不需要配置了】

~~~
` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` 
` ` `
  <!-- KISSO 初始化 -->
  <bean id="kissoInit" class="com.baomidou.kisso.web.WebKissoConfigurer" init-
method="initKisso">
      <property name="ssoPropPath" value="properties/sso.properties" />
  </bean>
  <mvc:interceptors>
    <!-- SSO 拦截器 -->
    <!-- path 对所有的请求拦截使用/**，对某个模块下的请求拦截使用：/myPath/* -->
    <mvc:interceptor>
        <mvc:mapping path="/**" />
        <bean class="com.baomidou.kisso.web.spring.SSOInterceptor" />
    </mvc:interceptor>
  </mvc:interceptors>
` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` 
` ` `
~~~

Spring SSO 拦截器配置，此基础上支持方法注解，如下跳过该方法登录验证。

~~~
` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` 
` ` `
@Login(action = Action.Skip)
~~~

7、waf 防火墙配置【可选】 # waf 防火墙配置

Servlet 方式 SSO 防火墙滤器 web.xml 配置 【可选】

~~~
 ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` 
` ` `
  <!-- WafFilter use . -->
  <filter>
    <filter-name>WafFilter</filter-name>
    <filter-class>com.baomidou.kisso.web.filter.WafFilter</filter-class>
    <init-param>
        <param-name>over.url</param-name>
        <param-value>/test/a.html;/test/b.html</param-value>
    </init-param>
    <init-param>
      <param-name>filter_xss</param-name>
      <param-value>true</param-value>
    </init-param>
    <init-param>
      <param-name>filter_sql_injection</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>WafFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
~~~

8、api 服务及 oauth2

kisso-oauth2-demo oauth2 演示 demo 由于 apache-oltu-oauth2 很灵活 kisso 只做辅助支持，不重新造轮子。

kisso_ApiServer 移动端 api 演示 demo 采用微信验证机制， 当然你也可以不采用该方法、直接在移动端维护 cookie 方式访问 api

一、 Api 授权认证方式 - 采用微信公众平台 api 认证机制

1、使用AES加密时，当密钥大于128时，代码会抛出异常：java.security.InvalidKeyException: Illegal key size

2、是指密钥长度是受限制的，java运行时环境读到的是受限的policy文件。文件位于${java_home}/jre/lib/security
这种限制是因为美国对软件出口的控制。

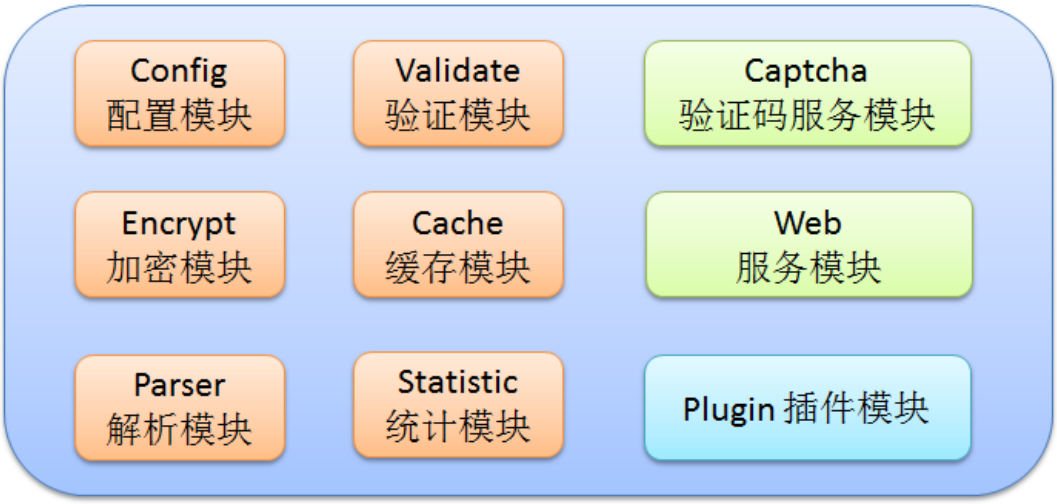
解决办法：http://www.oracle.com/technetwork/java/javaseproducts/downloads/index.html
进入 Oracle JDK 下载 Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files
for JDK/JRE 8 【下载对应 JDK 版本的 Policy 文件】
JDK8 下载地址：http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html
下载包的readme.txt 有安装说明。就是替换${java_home}/jre/lib/security/ 下面的local_policy.jar和US_export_policy.jar

二、 Api 授权认证方式 - Oauth2 认证机制

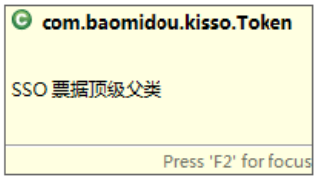
官方地址：http://oauth.net/2/
kisso 支持 oauth2 依赖 jars http://oltu.apache.org/

9、kisso 深入解析

9.1、项目模块结构



kisso 模块介绍：



com.baomidou.kisso.AuthToken

com.baomidou.kisso.SSOToken

kisso 会话 cookie 内容为加密票据，SSO 票据都必须继承父类 Token。

com.baomidou.kisso.SSOConfig

com.baomidou.kisso.web.WebKissoConfigurer

config 配置模块，通过 web 模块 WebKissoConfigurer（继承 SSOConfig）加载 sso.properties 初始化 kisso 配置。

com.baomidou.kisso.common.encrypt.AES

com.baomidou.kisso.common.encrypt.SSOEncrypt

encrypt 加密模块，默认采用 AES 加密。通过实现类 SSOParser 支持自定义。

com.baomidou.kisso.common.parser.FastJsonParser

com.baomidou.kisso.common.parser.SSOParser

parser 解析模块，采用 json 方式序列化 token 对象。通过实现类 SSOEncrypt 支持自定义。

com.baomidou.kisso.common.CookieHelper

com.baomidou.kisso.common.IpHelper

com.baomidou.kisso.common.Browser

validate 验证模块，主要 cookie 合法性进行校验，包括 ip 验证、浏览器验证、加密验证... 等。

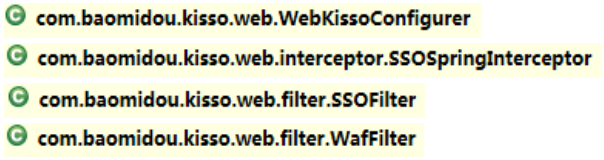
com.baomidou.kisso.SSOCache

cache 缓存模块，该模块为了支持用户心跳及踢出功能。

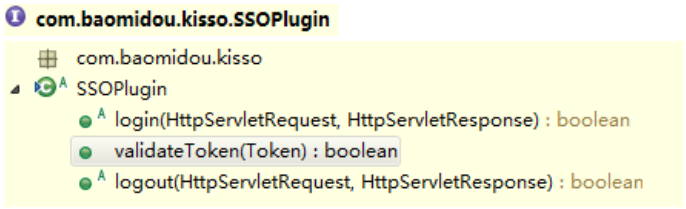
com.baomidou.kisso.SSOStatistic

statistic 统计模块，只需要实现 SSOStatistic 接口，注入实现类、或配置 sso.statistic.class = 实现类。

captcha 验证码服务模块，Java 图片验证码服务。

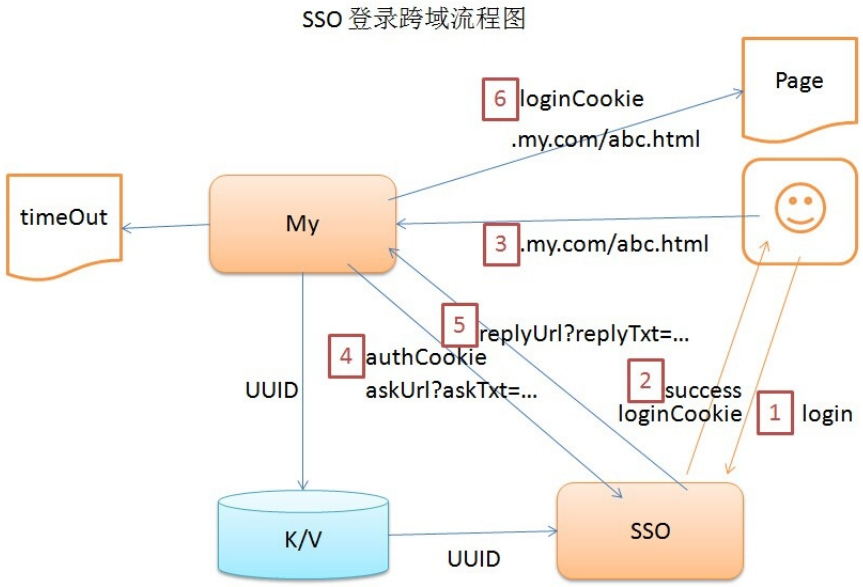


web 服务模块，负责 kisso 配置加载、服务启动、权限拦截、防火墙拦截。



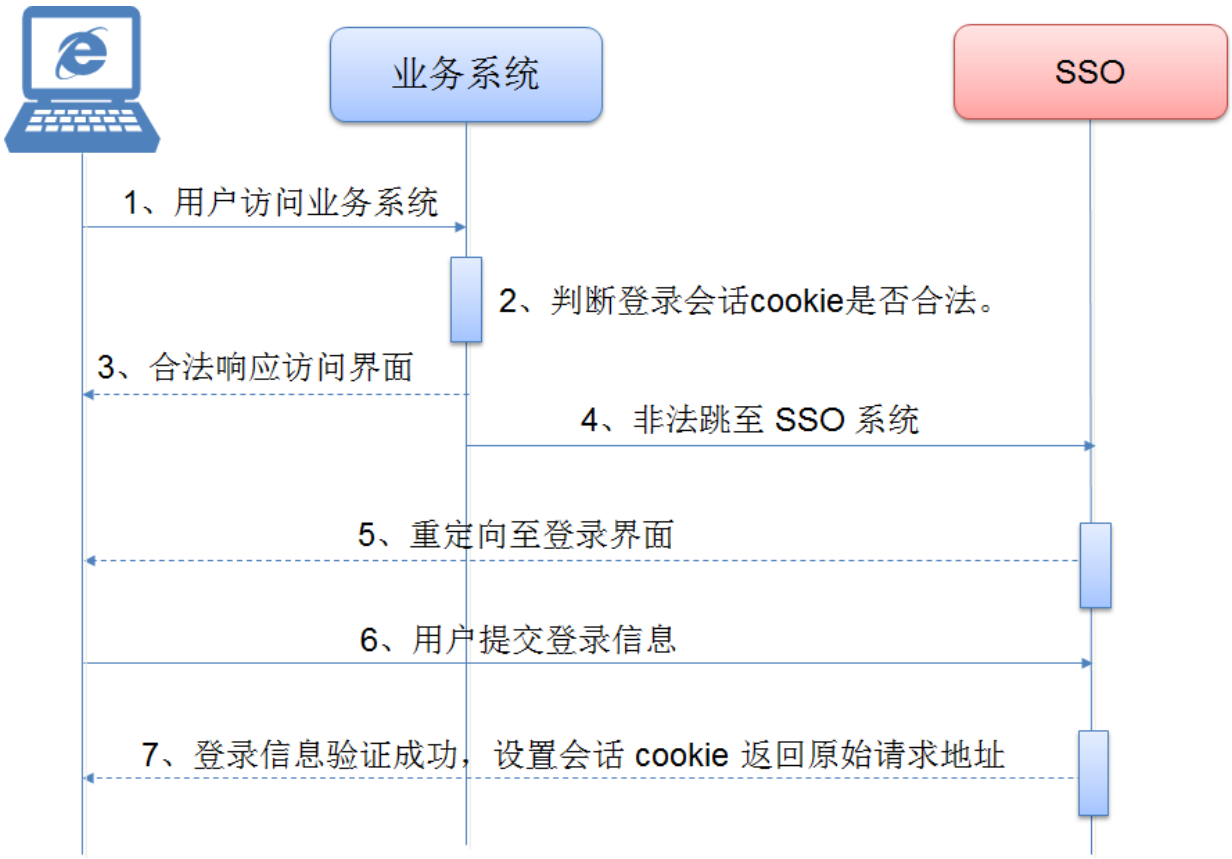
plugin 插件模块，为支持扩展 login logout token自定义验证， 二次封装提供接入点。

9.2、跨域原理

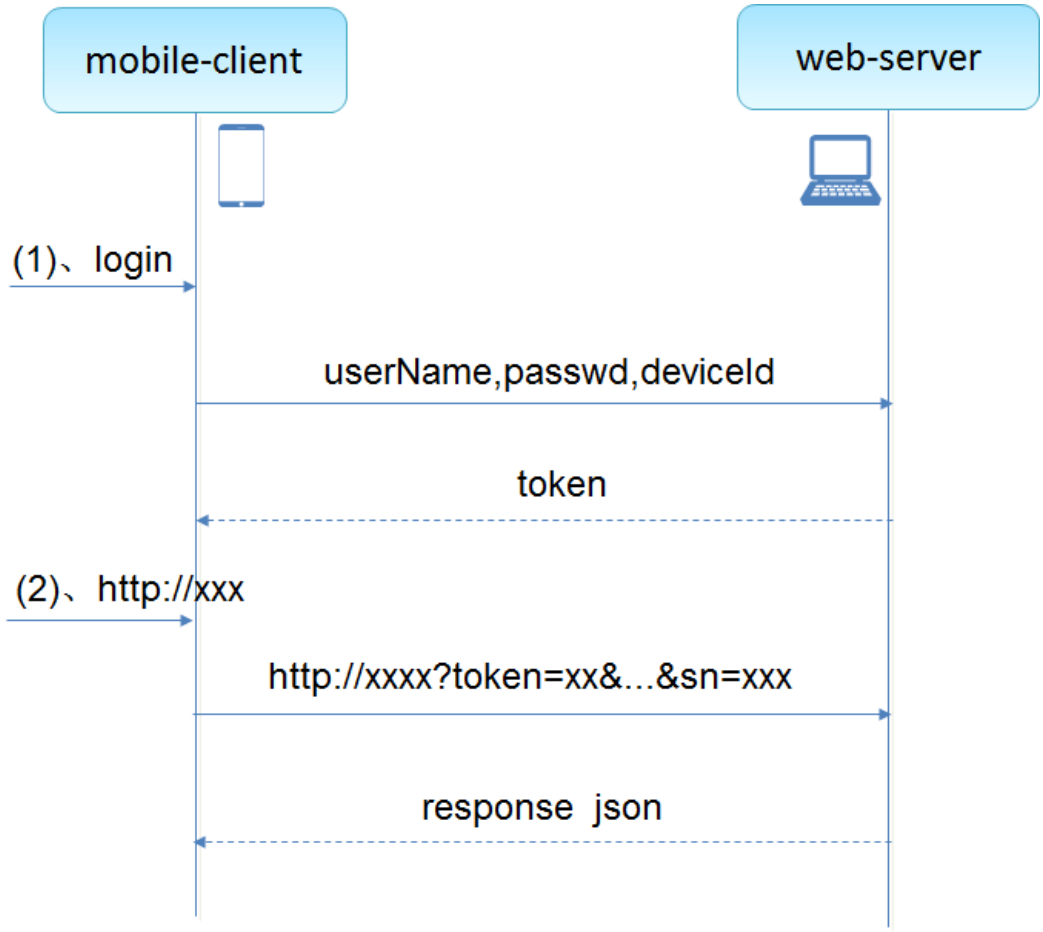


核心原理 Ajax Jsonp 跨域验证，用户在 A 站登录成功，进入 B 站时利用 Jsonp 跨域询问 SSO 是否 A 站登录？由 SSO 返回密文内容告诉 B 站是否 A 站登录，如登录在 B 站设置会话 Cookie 从而实现 B 站的登录授权，图中 UUID 部分为了安全利用分布式缓存后台做二次验证。

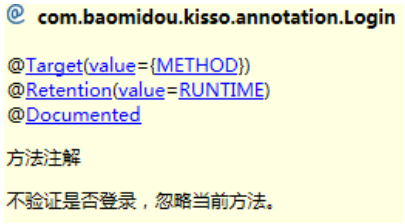
9.3、Web 端访问流程图



9.4、移动端 API 访问流程图



9.5、注解说明（当前只支持 spring 拦截器）



@Login(action = Action.Skip) 忽略拦截

Action 属性：Normal 默认拦截，Skip 跳过

9.6、登录设置 Cookie

没错！只需要一行代码：

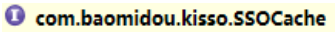
```
//记住密码，设置 cookie 时长 1 周 = 604800 秒【动态设置 maxAge 实现记住密码功能】
// if ( "on".equals(req.getParameter("rememberMe")) ) { request.setAttribute(SSOConfig.SSO_COOKIE_MAXAGE, 604800); }
SSOHelper.setSSOCookie(getRequest(), getResponse(), token, true);
```

9.7、退出登录

没错！只需要一行代码：

```
/**
 * <p>
 * SSO 退出，清空退出状态即可
 * </p>
 * <p>
 * 子系统退出 SSOHelper.logout(request, response); 注意 sso.properties 包含 退出到
 * SSO 的地址 ， 属性 sso.logout.url 的配置
 * </p>
 */
SSOHelper.clearLogin(getRequest(), getResponse());
```

9.8、SSOCache 如何维护用户心跳及踢出登录的？



cache 缓存模块，提供 SSOCache 接口。只要实现该接口 kisso 自动支持用户 心跳维护 登录踢出 功能。

原理：kisso 登录会话 cookie 与 cache 后台状态自动绑定，只要后台状态发生变化，会话 cookie 自动清

除，退出登录状态。

注意：实现缓存接口后需要配置 sso.properties 缓存部分，在 cache.get 中自己添加缓存延时设置。

9.9、如何生成新的密钥？

AES 密钥， 调用方法： SSOHelper.getSecretKey()
RSA 密钥，查看测试方法：/kisso/src/test/java/com/baomidou/kisso/TestRSA.java

8、常见问题

1、无法登录？

常见原因！错误的使用 localhost 访问，导致 sso.properties 配置 sso.login.url 地址不一致，此时如果无法登录请勿使用 localhost 。

2、使用 IP 如何配置？

注意访问时候使用 ip 访问，domain 的配置 ip 即可，不要是 .192.168.1.3 注意此时不要在前面加个 . 点。

3、javax.crypto.BadPaddingException: Given final block not properly padded

注意 1、是否加入Jar 包库 bcprov ，2、是否使用过程修改过 密钥 是删除浏览器 cookie 重启即可。

4、sso.cookie.secure=true 问题？

注意 非 https 下勿开启该属性。

5、cookie 被盗用怎么办？

怕盗用！ 开启 **https** 及 **ip** 验证**（建议后台系统开启）**， 关于 token 销毁 问题，
解决方法：

1、实现 SSOCache 自动处理心跳逻辑，设置时间不使用自动失效
2、自定义拦截器或者自定义插件判断 time 登录时间 ip 等信息，手动清理
3、cookie 超时设置

6、Web Api 版本如何管理？

大版本号： /api/v1/xxx ， /api/v2/xxx 区分。

小版本号：spring 框架推荐版本号写到 header 里面， 然后 spring 方法注解：@RequestMapping(headers="apt-version=1.0", value = "/api/user/create", method = RequestMethod.POST) ，其他框架可以采用 url 请求参数， content type 内容等区分。

7、第三方如何接入？

很简单查看跨域逻辑，使用 RSA 加密询问机制认证授权。

8、都谁在用 kisso？

不要问我星星有几颗， 我会告诉你很多很多！！（保密）

kisso 已过线上考验，由于 SSO 系统的重要性故此保密。