1、shiro原理图如下:

框架解释：

subject：主体，可以是用户也可以是程序，主体要访问系统，系统需要对主体进行认证、授权。

securityManager：安全管理器，主体进行认证和授权都 是通过securityManager进行。它包含下面的认证器和授权器。

authenticator：认证器，主体进行认证最终通过authenticator进行的。

authorizer：授权器，主体进行授权最终通过authorizer进行的。

sessionManager：web应用中一般是用web容器对session进行管理，shiro也提供一套session管理的方式。可以实现单点登录。

SessionDao： 通过SessionDao管理session数据，针对个性化的session数据存储需要使用sessionDao。

cache Manager：缓存管理器，主要对session和授权数据进行缓存，比如将授权数据通过cacheManager进行缓存管理，和ehcache整合对缓存数据进行管理。

realm：域，领域，相当于数据源，通过realm存取认证、授权相关数据。（它的主要目的是与数据库打交道，查询数据库中的认证的信息（比如用户名和密码），查询授权的信息（比如权限的code等，所以这里可以理解为调用数据库查询一系列的信息，一般情况下在项目中采用自定义的realm，因为不同的业务需求不一样））

2.Shiro

2.1 Shiro是什么?

> java中的安全框架,执行身份验证、授权、密码学和会话管理

2.2 Shiro的使用

2.2.1 添加依赖

```
<dependency>
  <groupId>org.apache.shiro</groupId>
  <artifactId>shiro-core</artifactId>
  <version>${shiro-core.version}</version>
</dependency>
<dependency>
  <groupId>commons-logging</groupId>
  <artifactId>commons-logging</artifactId>
  <version>${commons-logging.version}</version>
</dependency>
```

2.2.2 身份的验证(登录)

准备数据,需要一个ini配置文件,输入以下内容

```
[users]
admin=123
zhang=234
```

2.2.3 进行登录

```
//1.获取SecurityManager
Factory<SecurityManager> factory = new
IniSecurityManagerFactory("classpath:shiro01/shiro.ini");
SecurityManager securityManager = factory.getInstance();

//2.包裹SecurityManager ,方便在代码的任意地点调用
SecurityUtils.setSecurityManager(securityManager);

//3.获取用户主体
Subject subject = SecurityUtils.getSubject();

//  判断用户是否认证
System.out.println(subject.isAuthenticated());

//4.获取Token
UsernamePasswordToken usernamePasswordToken = new UsernamePasswordToken("admin",
"1234");

//5.进行登录
subject.login(usernamePasswordToken);

System.out.println(subject.isAuthenticated());
```

2.3 身份认证的流程01

- SecurityManager:核心管理器，管理Shiro中的所有的组键；请求分发
- Authenticator:认证器 认证业务操作
- Realm: 安全数据源 (数据查询的方式)
- Subject:用户主体

2.4 JdbcRealm的使用

2.4.1 导入数据库的依赖

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>${mysql-connector-java.version}</version>
</dependency>

<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid</artifactId>
  <version>${druid.version}</version>
</dependency>
```

2.4.2 添加配置ini文件

```
# 配置dataSource
dataSource=com.alibaba.druid.pool.DruidDataSource
dataSource.driverClassName=com.mysql.jdbc.Driver
dataSource.url=jdbc:mysql://127.0.0.1:3306/shiro?useUnicode=true&characterEncoding=utf8
dataSource.username=root
# dataSource.username=

# 创建一个jdbcRealm
realm=org.apache.shiro.realm.jdbc.JdbcRealm
realm.dataSource=$dataSource

# 使用自定义的查询语句
# realm.authenticationQuery= select pwd from t_user where name=?

# 将jdbcRealm交个SecurityManager
securityManager.realm=$realm
```

2.4.3 数据库表

```
默认shiro查询的语句为 select password from users where username=?
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

如果想要自定表的话：
    可以在ini配置文件中通过:
    realm.authenticationQuery= select pwd from t_user where name=?
    去修改默认的查询语句
```

2.5 Shiro的授权

2.5.1 基于角色的授权

可以理解为权限的集合。

2.5.2 基于权限的授权

权限的规则: 资源的表示符:操作:实例的ID

```
user:*:* =  user:*(建议)  = user            对用户资源拥有所有操作
user:update:*  = user:update (建议)         对所有用户资源拥有update权限
user:view:2                                  对id为2的用户拥有查看权限
user:update,user:view  = "user:update,view"
*:*:*
```

> jdbcRealm中默认不会查询权限数据,如果需要查询权限数据，需要添加配置。
>
> realm.permissionsLookupEnabled=true