数据校验

给项目添加拦截器，拦截所有请求，并对请求进行各种校验

Spring Security，Shiro

认证，授权，防护，资源权限管理

将校验逻辑分发到不同的过滤器，一个过滤器负责处理一种认证方式，

UsernamePassword 过滤器验证用户和密码

1.pom.xml添加依赖

```
<!--        spring security-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>

    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
            <version>2.4.5</version>
        </dependency>

    <resources>
            <resource>
                <directory>src/main/java</directory>
                <includes>
                    <include>**/*.xml</include>
                </includes>
            </resource>
        </resources>
```

2.创建 index.html

3.自定义用户名和密码 application.xml

```
spring:
  security:
    user:
      name: admin
      password: 123123
```

资源权限管理

页面：index.html、admin.html

角色：ADMIN、USER

权限：ADMIN 可以访问 index.html 和 admin.html

　　User 只能访问 index.html

1.创建一个配置类

```
package com.southwind.configure;

import org.springframework.security.crypto.password.PasswordEncoder;

public class PasswordEncoderImpl implements PasswordEncoder {
    @Override
    public String encode(CharSequence rawPassword) {
        return rawPassword.toString();  //对securityConfiguration里面设置的密码进行
编码
    }

    @Override
    public boolean matches(CharSequence rawPassword, String encodedPassword) {
        return encodedPassword.equals(rawPassword.toString()); //前端传来的密码和设
置的密码进行校验
    }
}
```

2.SecurityConfiguration

```
package com.southwind.configure;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authentic
ationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;


@Configuration
@EnableWebSecurity
public class SecurityConfiguration  extends WebSecurityConfigurerAdapter {
    @Override
//    设置账户用户名和密码，以及对应的角色
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{
        auth.inMemoryAuthentication().passwordEncoder(new PasswordEncoderImpl())
                .withUser("user") //设置用户名
                .password(new PasswordEncoderImpl().encode("123456"))  //设置密码
                .roles("User") //设置对应的角色
                .and()
                .withUser("admin")
                .password(new PasswordEncoderImpl().encode("123456"))
                .roles("ADMIN","USER"); //一个账号可以拥有多个角色权限
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
```

```
                .antMatchers("/admin").hasRole("ADMIN") //admin页面与角色admin绑定
                .antMatchers("/index")
                .access("hasRole('ADMIN') or hasRole('index')") //index页面与角色
admin,user绑定
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login") //login不用过滤
                .permitAll()
                .and()
                .logout()
                .permitAll()
                .and()
                .csrf() //关闭
                .disable();
```

3.controller

```
@Controller
public class IndexController {

    @GetMapping("/index")
    public String index(){
        return "index";
    }

    @GetMapping("/admin")
    public String admin(){
        return "admin";
    }

    @GetMapping("/login")
    public String login(){
        return "login";
    }
}
```

4.login.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="shortcut icon" href="#"/>
</head>
<body>
<form action="/login" method="post">
    <table>
<!--        将错误信息展示出来-->
        <span th:text="${param.error}" style="color: red"></span>
        <tr>
            <td>用户名：</td>
            <td>
                <input type="text" name="username"/>
```

```
            </td>
        </tr>
        <tr>
            <td>密码：</td>
            <td>
                <input type="password" name="password"/>
            </td>
        </tr>
        <tr>
            <td>
                <input type="submit" value="登录"/>
            </td>
        </tr>
    </table>
</form>
</body>
</html>
```

index.html

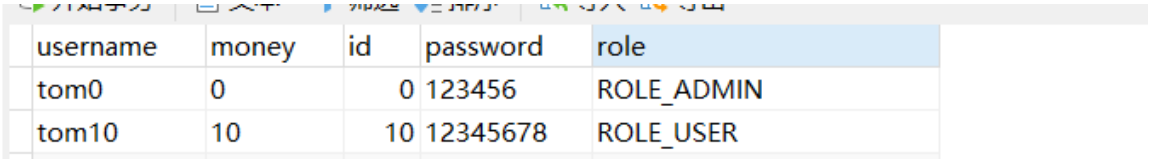
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>index</p>
    <form action="/logout" method="post">
        <button type="submit">退出</button>
    </form>
</body>
</html>
```

admin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>admin</p>
    <form action="/logout" method="post">
        <button type="submit">退出</button>
    </form>
</body>
</html>
```

Spring Security整合数据库

1.创建数据库 角色必须使用ROLE_XXX (都必须大写)



username	money	id	password	role
tom0	0	0	123456	ROLE_ADMIN
tom10	10	10	12345678	ROLE_USER

2.创建实体类

```
import lombok.Data;

@Data
public class Account {

   private String username;
   private String money;
   private Integer id;
   private  String  password;
   private String role;
}
```

3.创建mapper

```
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.ishang.entity.Account;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Repository;


@Repository
public interface AccountMapper extends BaseMapper<Account> {

    public Account findByName(@Param("username") String username);

}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.ishang.mapper.AccountMapper">

    <select id="findAll" resultType="com.ishang.entity.Account">
        select * from account
    </select>
    <select id="findByName" resultType="com.ishang.entity.Account">
        select * from account where username= #{username}
    </select>

</mapper>
```

5.创建Use'rDetailServieImpl继承spring security中提供的接口UserDeatilService

```
import com.ishang.entity.Account;
import com.ishang.mapper.AccountMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.Collection;

@Service
public class UserDetailServiceImpl implements UserDetailsService {
   @Autowired
   private AccountMapper accountMapper;
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
//       1.通过loadUserByUsername将前台输入的username获取出来，去数据库查找是否有该用户名
        Account account = this.accountMapper.findByName(username);
        if (account == null ){

                throw  new UsernameNotFoundException("用户名"+username+"不存在");
        }
//        2.如果用户名存在，则将数据库中真正的密码和角色取出来，装到集合中，因为一个用户可能有
多个角色
        Collection<GrantedAuthority> grantedAuthorities= new ArrayList<>();
        GrantedAuthority grantedAuthority = new
SimpleGrantedAuthority(account.getRole());
        grantedAuthorities.add(grantedAuthority);
//        返回一个user
        return new User(username,account.getPassword(),grantedAuthorities);
    }
}
```

6.创建configuration

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authentic
ationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;


@Configuration
```

```
//@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

//    3.将service注入，完成用户登录逻辑

    @Qualifier("userDetailServiceImpl")
    @Autowired
    private UserDetailsService userDetailsService;

    @Override
//    设置账户用户名和密码，以及对应的角色
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{


 auth.userDetailsService(this.userDetailsService).passwordEncoder(passwordEncode
r())

    }

//    注入一个PasswordEncoder实例
    @Bean
    public PasswordEncoder passwordEncoder(){
        return NoOpPasswordEncoder.getInstance();
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/admin")
                .hasRole("ADMIN")
                .antMatchers("/index")
                .access("hasRole('ADMIN') or hasRole('index')")
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .permitAll()
                .and()
                .logout()
                .permitAll()
                .and()
                .csrf()
                .disable();
    }
}
```

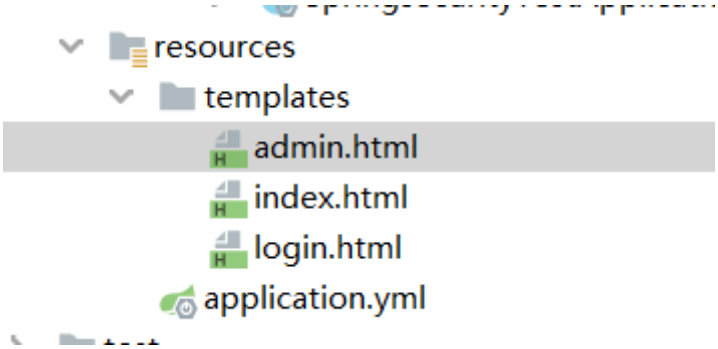
7.配置application.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/test
    username: root
    password: 123456
```

```
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
  security:
    user:
      name: root
      password: 123456

mybatis-plus:
  configuration:
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
  mapper-locations: com/ishang/mapper/*.xml
```

8.创建静态文件



logout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>admin</p>
    <form action="/logout" method="post">
        <button type="submit">退出</button>
    </form>
</body>
</html>
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>index</p>
    <form action="/logout" method="post">
        <button type="submit">退出</button>
    </form>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="shortcut icon" href="#"/>
</head>
<body>
<form action="/login" method="post">
    <table>
<!--        将错误信息展示出来-->
        <span th:text="${param.error}" style="color: red"></span>
        <tr>
            <td>用户名：</td>
            <td>
                <input type="text" name="username"/>
            </td>
        </tr>
        <tr>
            <td>密码：</td>
            <td>
                <input type="password" name="password"/>
            </td>
        </tr>
        <tr>
            <td>
                <input type="submit" value="登录"/>
            </td>
        </tr>
    </table>
</form>
</body>
</html>
```

9.创建controller

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class IndexController {

    @GetMapping("/index")
    public String index(){
        return "index";
    }

    @GetMapping("/admin")
    public String admin(){
        return "admin";
    }

    @GetMapping("/login")
    public String login(){
        return "login";
```

```
    }


}
```