自定义登录逻辑

当进行自定义登录逻辑时需要用到之前讲解的`UserDetailsService`和`PasswordEncoder`。但是Spring Security 要求：当进行自定义登录逻辑时容器内必须有`PasswordEncoder`实例。所以不能直接 new 对象。

编写配置类

```
package com.xxxx.springsecuritydemo.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

/**
 * @author zhoubin
 * @since 1.0.0
 */
@Configuration
public class SecurityConfig {

   @Bean
   public PasswordEncoder getPw(){
      return new BCryptPasswordEncoder();
   }
}
```

自定义逻辑

在 Spring Security 中实现 UserDetailService 就表示为用户详情服务。在这个类中编写用户认证逻辑。

```
package com.xxxx.springsecuritydemo.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.authority.AuthorityUtils;
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.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

/**
 * @author zhoubin
 * @since 1.0.0
 */
@Service
public class UserServiceImpl implements UserDetailsService {
   @Autowired
   private PasswordEncoder pw;
```

```
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException
{
        //1.查询数据库判断用户名是否存在，如果不存在抛出UsernameNotFoundException异常
        if (!"admin".equals(username)){
            throw new UsernameNotFoundException("用户名不存在");
        }
        //2.把查询出来的密码（注册时已经加密过）进行解析，或直接把密码放入构造方法中
        String password = pw.encode("123");
        return new User(username,password,
AuthorityUtils.commaSeparatedStringToAuthorityList("admin,normal"));
    }
}
```

查看效果

重启项目后，在浏览器中输入账号：admin，密码：123。后可以正确进入到 login.html 页面。