СБЕРБАНК ТЕХНОЛОГИИ

Spring: MVC + Security

ЧТО ЭТО И ЗАЧЕМ?

СБЕРБАНК ТЕХНОЛОГИИ

это Java/JavaEE framework, предоставляющий механизмы построения систем **аутентификации** и **авторизации**, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework.

ОСОБЕННОСТИ

СБЕРБАНК ТЕХНОЛОГИИ

- Всесторонняя и расширяемая поддержка как аутентификации, так и авторизации
- Защита от атак, таких как фиксация сеанса, clickjacking, подделка запроса и т. д.
- Интеграция с сервлетами
- Интеграция с Spring Web MVC
- Многое другое…

ДОПОЛНИТЕЛЬНЫЕ МОДУЛИ SPRING SECURITY

СБЕРБАНК ТЕХНОЛОГИИ

- **spring-security-web -** Содержит фильтры и код поддержки инфраструктуры веб-безопасности. Все, что связано с зависимостями servlet-API.
- **spring-security-ldap -** LDAP-аутентификация
- **spring-security-oauth2-* -** поддержка авторизации OAuth 2.0 и OpenID 1.0
- **spring-security-cas –** Central Authentication Service, протокола управления доступом, обеспечивающего технологию единого входа (SSO) в веб-сервисах**.**
- **spring-security-openid -** поддержка аутентификации в сети OpenID

ОСНОВНЫЕ КОМПОНЕНТЫ

СБЕРБАНК ТЕХНОЛОГИИ

- **SecurityContextHolder** – «держатель» контекста безопасности, по умолчанию использует **ThreadLocal**.
- **SecurityContext**
  - **Authentication** – состояние аутентификации пользователя (Principal)
  - **GrantedAuthority** – разрешения (роли), выданные пользователю (ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN)
- **UserDetails** – используется для построения объекта Authentication. Содержит в себе имя, пароль, флаги состояния, коллекцию ролей.
- **UserDetailsService** – сервис, реализация, которого должна получать данные пользователя (**UserDetails**) по его имени.

ПРОЦЕСС АУТЕНТИФИКАЦИИ

СБЕРБАНК ТЕХНОЛОГИИ

1. Пользователь указывает идентификатор и пароль. Данные объединяются в **UsernamePasswordAuthenticationToken** *(экземпляр интерфейса Authentication)*
2. Токен передается экземпляру **AuthenticationManager** для проверки.
   a) При неудачной попытке аутентификации будет выброшено исключение **BadCredentialsException** с сообщением “Bad Credentials”.
   b) Если аутентификация прошла успешно возвращает полностью заполненный экземпляр **Authentication**.
3. Устанавливается контекст безопасности через **SecurityContextHolder.getContext()** **.setAuthentication(…)**, куда передается объект, полученный из **AuthenticationManager**.

HELLO WEB SECURITY CONFIGURATION

СБЕРБАНК ТЕХНОЛОГИИ

```
@EnableWebSecurity
public class WebSecurityConfig implements WebMvcConfigurer {

    @Bean
    public UserDetailsService userDetailsService() throws Exception {
        InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();
        manager.createUser(User.withDefaultPasswordEncoder()
                .username("user").password("password").roles("USER").build());

        return manager;
    }
}
```

- Требует аутентификацию для каждого URL-адреса
- Создаёт форму входа
- Реализует выход пользователя из системы
- Предотвращение атаки CSRF
- Защита фиксации сеанса
- Интегрирует заголовок безопасности
- Интегрирует X-XSS-Protection и др.

WEB SECURITY CONFIG: HTTP SECURITY

СБЕРБАНК ТЕХНОЛОГИИ

```
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
            .antMatchers("/resources/**", "/signup", "/about")
            .permitAll()
            .antMatchers("/admin/**")
            .hasRole("ADMIN")
            .antMatchers("/db/**")
            .access("hasRole('ADMIN') and hasRole('DBA')")
            .anyRequest().authenticated()
            .and().formLogin();
}
```

- Настраиваем доступ к конкретным ресурсам по URL
  - требуем конкретные роли
  - разрешаем/запрещаем анонимный доступ
  - разрешаем доступ аутентифицированному пользователю
  - и т.д.

• НАСТРАИВАЕМ МЕХАНИЗМ LOGOUT

СБЕРБАНК ТЕХНОЛОГИИ

```
http.logout()
        .logoutUrl("/my/logout")
        .logoutSuccessUrl("/my/index")
        .logoutSuccessHandler(logoutSuccessHandler)
        .invalidateHttpSession(true)
        .addLogoutHandler(logoutHandler)
        .deleteCookies(cookieNamesToClear)
        .and()…
```

ПРИМЕР НАСТРОЙКИ JDBC-АУТЕНТИФИКАЦИИ

СБЕРБАНК ТЕХНОЛОГИИ

```
@Autowired private DataSource dataSource;
@Autowired public void configureGlobal(
AuthenticationManagerBuilder auth) throws Exception {
    UserBuilder users = User.withDefaultPasswordEncoder();
    auth.jdbcAuthentication()
            .dataSource(dataSource)
            .withDefaultSchema()
            .withUser(users.username("user")
                    .password("password")
                    .roles("USER"))
            .withUser(users.username("admin")
                    .password("password")
                    .roles("USER", "ADMIN"));
}
```

METHOD SECURITY

СБЕРБАНК ТЕХНОЛОГИИ

@EnableGlobalMethodSecurity(securedEnabled = true, prePostEnabled = true) – включает поддержку «защиты» методов аннотацией **@Secured** и @PreAuthorize, @PreFilter, @PostAuthorize, @PostFilter. Рассмотрим их подробнее…

@SECURED

СБЕРБАНК ТЕХНОЛОГИИ

Позволяет ограничить доступ пользователя к методам, помеченным этой аннотацией. При отсутствии прав у пользователя будет брошено исключение.

```
@Secured({ "ROLE_USER" })
public void create(Contact contact);

@Secured({ "ROLE_USER", "ROLE_ADMIN" })
public void update(Contact contact);

@Secured({ "ROLE_ADMIN" })
public void delete(Contact contact);
```

@SECURED

СБЕРБАНК ТЕХНОЛОГИИ

Позволяет ограничить доступ пользователя к методам, помеченным этой аннотацией. При отсутствии прав у пользователя будет брошено исключение.

```
@Secured({ "ROLE_USER" })
public void create(Contact contact);

@Secured({ "ROLE_USER", "ROLE_ADMIN" })
public void update(Contact contact);

@Secured({ "ROLE_ADMIN" })
public void delete(Contact contact);
```

@PREAUTHORIZE, @POSTAUTHORIZE

СБЕРБАНК ТЕХНОЛОГИИ

Не просто ограничивает доступ пользователя к методам, но и позволяет указывать белее сложные выражения (возможности расширяются модулями), что, правда, требует дополнительной настройки…

```
@PreAuthorize("hasRole('ROLE_USER')")
public void create(Contact contact);
…
@PreAuthorize("hasPermission(#contact, 'admin')")
public void deletePermission(Contact contact, Sid
recipient, Permission permission);
```

@PREFILTER, @POSTFILTER

СБЕРБАНК ТЕХНОЛОГИИ

Позволяет фильтровать данные, основываясь на правах пользователя. При использовании **@PreFilter** – Spring Security итерирует коллекцию передаваемого аргумента, а в случае **@PostFilter** – коллекцию возвращаемого типа и удаляет неподходящие элементы.

```
@PreAuthorize("hasRole('ROLE_USER')")
@PostFilter("hasPermission(filterObject, 'read') or
hasPermission(filterObject, 'admin')")
public List<Contact> getAll();
```

LINKS

СБЕРБАНК ТЕХНОЛОГИИ

https://docs.spring.io/spring-security/site/docs/5.0.3.RELEASE/reference/htmlsingle/