IŠ PRAEITOS PASKAITOS

- React atributų perdavimas event metu
- perduoti per funkciją

```
<input onClick={(event) => this.onClick(event, papildomas)} />
```

- įsidėti į papildomą komponentą ir perduoti per props

```
class Komponentas extends Component {
  handleClick = (event) => {
    this.props.onClick(this.props.papildomas);
  }
  render() { return (
      <input onClick={this.handleClick} /
    > ); }
}
// tuomet panaudojimas būtų toks
<Komponentas papildomas={papildomas}
  onClick={this.onKomponentoClick} />
```

AKADEMIJA.IT

INFOBALT IR TECH CITY

SPRING SECURITY. LIVE CODING. PRAKTIKA

Andrius Stašauskas

andrius@stasauskas.lt

http://stasauskas.lt/itpro2018/

TURINYS

- Spring Security
- Live Coding
- Praktika

TURINYS

- Kas yra Spring Security
- Prijungimas prie projekto
- Prijungimas prie UI

KAS YRA SPRING SECURITY

AKADEMIJA.IT
INFOBALT IR TECH CITY

KODĖL ATSIRADO?

- Saugumas - svarbus aspektas be išimties visoms aplikacijoms
- Saugumas stipriai įtakoja aplikacijos funkcionalumą
- Aplikacija saugumu pati rūpintis neturėtų
  - todėl norima atskirti saugumą nuo aplikacijos funkcijų

TRUMPA ISTORIJA

- 2003 sukurtas Acegi Security
  - saugumo servisai Spring karkasui
- Nuo 1.1.0 Acegi tapo Spring moduliu

KAS YRA SPRING SECURITY

Spring Security is a powerful and highly customizable authentication and access-control framework. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications.

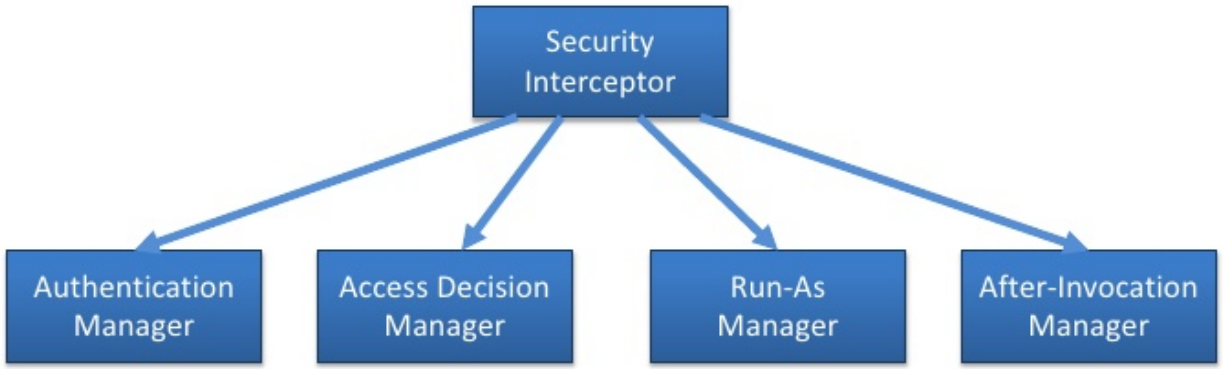
- Authentication - ar asmuo dedasi tuo, kuo sako
  - pvz. per slaptažodį
- Authorization - taisyklės, nusakančios, kas gali ką daryti
  - pvz. per roles (angl. ROLE)

PAGRINDINIAI ELEMENTAI

- Filtrai (Security Interceptor)
  - praktiškai patikrina username ir password
  - deleguoja užklausas Manager'iams
- Autentikacija
- Autorizacija
  - Web
  - Metodai

SECURITY INTERCEPTOR

Fundamental elements of Spring Security

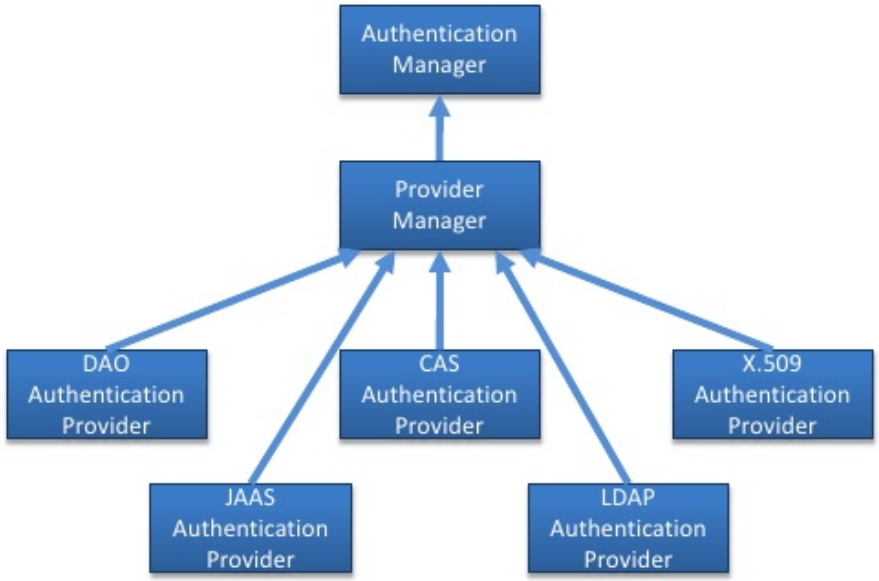


AUTHENTICATION MANAGER

Authentication Manager

• verifies *principal (typically a username) and credentials* (typically a password)

• Spring Security comes with a handful of flexible authentication managers that cover the most common authentication strategies



PRIJUNGIMAS PRIE PROJEKTO

PRIKLAUSOMYBĖ

- jungiame prie Spring Boot projekto

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

- application.properties

```
server.session.cookie.name = SECURITYID
```

KLAIDOS IŠMETIMAS REST

- SecurityEntryPoint.java kad išmestų 401 klaidą

```
@Component("restAuthenticationEntryPoint")
public class SecurityEntryPoint implements AuthenticationEntryPoint {
  @Override
  public void commence(HttpServletRequest request, HttpServletResponse response,
      AuthenticationException authException) throws IOException, ServletException
    response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");
  }

}
```

DUOMENŲ SAUGOJIMAS

- jau turimoje klasėje prisidedam slaptažodį

```
@Entity
@Table(name = "Naudotojas")
public class User {
  <..>
  @NotBlank
  private String password;
  <..>
  @Email
  @Size(min=6)
  private String email;
  @ManyToOne(cascade = {CascadeType.MERGE,
  CascadeType.DETACH}) @JoinColumn(name = "ROLE_ID")
  private Role role;
}
```

- aišku vietoje email galima naudoti tiesiog pvz username

NAUDOTOJŲ SERVISAS

- savo naudotojų servise implementinti UserDetailsService

```
@Service
public class UserService implements UserDetailsService { // <..>
  @Override
  public UserDetails loadUserByUsername(String username)
    throws UsernameNotFoundException {
      User user = findByEmail(username);
      if (user == null)
        throw new UsernameNotFoundException(username + " not found.")
      return new org.springframework.security.core.userdetails.User(
          user.getEmail(), user.getPassword(),
          AuthorityUtils.createAuthorityList(
           new String[] { "ROLE_" + user.getRole().getName() }) );
  } // <..>
  @Transactional(readOnly = true) public
  User findByEmail(String email) {
    return userRepository.findByEmail(email); } }
```

NAUDOTOJŲ SERVISAS

- kuriant naują naudotoją reikia užkoduoti slaptažodį su PasswordEndcoder, šiuo atveju su numatytuoju Spring

```
User newUser = new User();
newUser.setUsername(username);
PasswordEncoder encoder =
  PasswordEncoderFactories.createDelegatingPasswordEncoder();
newUser.setPassword(encoder.encode(password));
Role r = new Role();
r.setName("CALC");
newUser.setRole(r);
User saved = userRepository.save(newUser);
```

KONFIGŪRACIJA

- sukurti naują SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled=true,prePostEnabled=true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private SecurityEntryPoint securityEntryPoint;
    @Autowired
    private UserDetailsService userService;
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth)
      throws Exception {
      auth.userDetailsService(userService);
      //
      auth.inMemoryAuthentication().withUser("uu") /
    } /   .password("pp").roles("USER", "CALC");
}
```

KONFIGŪRACIJA

- taip pat SecurityConfig.java prijungti:

```
public class SecurityConfig <..> {
  // <..>
  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http
      .authorizeRequests()
        // be saugumo UI dalis ir swaggeris
        .antMatchers("/", "/swagger-
        ui.html").permitAll() // visi /api/ saugus (dar
        galima .anyRequest() ) .antMatchers("/api/**", "/
      .and()
      api/**").authenticated()
      // <..>
```

KONFIGŪRACIJA

- taip pat SecurityConfig.java prijungti:

```
// <..>
      .formLogin() // leidziam login
      // prisijungus
      .successHandler(new
      SimpleUrlAuthenticationSuccessHandler()) // esant blogiems
      user/pass
      .failureHandler(new
      SimpleUrlAuthenticationFailureHandler()) .loginPage("/
    .and()
//  <..>
      login").permitAll() // jis jau egzistuoja !
```

KONFIGŪRACIJA

- taip pat SecurityConfig.java prijungti:

```
  // <..>
        .logout().permitAll() // leidziam /
      logout .and()
        .csrf().disable() // nenaudojam tokenu
         // toliau forbidden klaidai
        .exceptionHandling()
          .authenticationEntryPoint(securityEntryPoint)
      .and()
        .headers().frameOptions().disable(); // H2 konsolei
  }
}
```

KONFIGŪRACIJA

- aišku prisijungus galima grąžinti username:

```
.successHandler(new AuthenticationSuccessHandler() {
  @Override
  public void onAuthenticationSuccess(HttpServletRequest request,
      HttpServletResponse response, Authentication authentication)
      throws IOException, ServletException {
    response.setHeader("Access-Control-Allow-Credentials", "true");
    response.setHeader("Access-Control-Allow-Origin",
      request.getHeader("Origin"));
    response.setHeader("Content-Type",
      "application/json;charset=UTF-8");
    response.getWriter().print("{\"username\": \""+
      SecurityContextHolder.getContext().getAuthentication().getName
      +"\"}");
  }
})
```

SAUGUMO ĮJUNGIMAS SERVISE

- bet kuriame servise, pvz. kalkuliatoriaus:

```
@RestController
public class HelloController {
    @RequestMapping(value = "calc", method = RequestMethod.GET)
    // Preauthorized galima or, secured - tik and
    @PreAuthorize("hasRole('CALC')") // @Secured("ROLE_CALC") public
    Result calc(@RequestParam int left, @RequestParam int righ
     //
    <..> }
}
```

- Neprisijungus bus Forbidden
  - Prisijungus be CALC rolės - access denied
- O kaip prisijungti per UI?

KAIP GAUTI PRISIJUNGUSĮ NAUDOTOJĄ

- bet kuriame servise, pvz. kalkuliatoriaus:

```
@RestController
public class HelloController {
    @RequestMapping(path = "/loggedUsername", method = RequestMethod.
    public String getLoggedInUsername() {
        Authentication authentication =
          SecurityContextHolder.getContext().getAuthentication(); if
        (!(authentication instanceof AnonymousAuthenticationToken)
            String currentUserName = authentication.getName();
            return currentUserName;
        }
        return "not logged";
    }
}
```

PRIJUNGIMAS PRIE UI

FORMA

- Login formos pvz.:

```
const Forma = ({email, pass, onPassChange, onEmailChange, onSubmit},
      context) => {
    return <form onSubmit={onSubmit}>
       <input type="text" value={email} onChange={onEmailChange}/>
       <input type="password" value={pass} onChange={onPassChange}/>
       <input type="submit"/>
    </form>;
  }
```

FORMACONTAINER

- Kaip siunčiame užklausas iš http://localhost:3000/:

```
import axios from 'axios';
axios.defaults.withCredentials = true; // leidzia dalintis cookies
class FormaContainer extends Component {
  onEmailChange=(event)=>{this.setState({email:event.target.value})}
  onPassChange=(event)=>{this.setState({pass:event.target.value})}
  onSubmit = (event) => {
    let userData = new URLSearchParams();
    userData.append('username', this.state.email);
    userData.append('password', this.state.pass);
    axios.post('http://localhost:8081/login', userData,
      {headers:{'Content-type':'application/x-www-form-urlencoded'}})
      .then((resp) => {
        console.log("user "+resp.data.username+" logged in") })
      .catch((e) => { console.log(e); });
    event.preventDefault();
  } }
```

FORMACONTAINER

```
class FormaContainer extends Component {
  // <..>
  render() {
    return <Forma email={this.state.email} pass={this.state.pass}
      onEmailChange={this.onEmailChange}
      onPassChange={this.onPassChange}
      onSubmit={this.onSubmit} />;
  }
  onCalc = (event) => {
      axios.get('http://localhost:8081/calc?left=1&right=2')
        .then((response) => { console.log(response); })
        .catch((e) => { console.log(e); });
      event.preventDefault();
  }
}
```

KAIP PADIDINTI SAUGUMĄ?

SLAPTAŽODŽIO KODAVIMAS

- Slaptažodžio kodavimas
- HTTPS tiek Rest, tiek UI
- CSRF tokenai formoms
- išorinė duomenų bazė (LDAP ar kitokia)
- ir kiti būdai

SLAPTAŽODŽIO KODAVIMAS

```
public class SecurityConfig <..> {
  @Bean
  public PasswordEncoder passwordEncoder() {
      return new BCryptPasswordEncoder(); // galima pakeisti
  }
  //<..>
  @Autowired
  public void configureGlobal(AuthenticationManagerBuilder auth)
    throws Exception {
       auth.userDetailsService(userService)
         .passwordEncoder(passwordEncoder());
  }
}
```

UŽDUOTIS #1

- jei niekada nedarėte, nukopijuokite React/Npm aplikacijos katalogą ir pakeiskite aplikacijos pavadinimą
- nusikopijuokite Spring Boot/Maven katalogą ir pakeiskite pavadinimą, paketą ir versiją savo Java projektui
- įsitikinkite, kad pakeitus pavadinimus projektai kompiliuojasi ir aplikacijos pasileidžia
  - egzamino metu bus nurodyta, kaip turi būti suformuoti aplikacijų pavadinimai, kokie paketai naudojami, kokia turi būti versija

UŽDUOTIS #2

- jau esate susikūrę ManyToMany ryšį (Krepšelis<->Prekė)
- sukurkite Rest servisą, kurio pagalba galima viena Rest užklausa pridėti keletą prekių į krepšelį
- sukurkite Rest servisą, kuris galėtų priimti visas krepšelyje pakeistas Quantity reikšmes ir atnaujinti jas visiems produktams iškart viena užklausa
  - aišku turi būti patikra, ar kiekvienos prekės norimas pirkti kiekis quantity neviršija ant prekės nustatytojo (neviršija visų prekių sandėlyje skaičiaus)

UŽDUOTIS #3

- taip pat patikrinimas turėtų tikrinti, kad į kitus krepšelius nėra pridėta prekių daugiau nei quantity, ir jeigu dabartiniam krepšeliui prašoma per daug prekių, - neleisti
- turi būti suformuotas klaidos pranešimas su sąrašu įvykusių klaidų
  - galiausiai pranešimas turi pasirodyti React aplikacijoje gražiai suformatuotu tekstiniu pavidalu
  - ir raudonai nuspalvinti atitinkami quantity langeliai
- jeigu tai įgyvendinsite Java priemonėmis, vietoje Java perrašykite į vieną ar kelias sudėtingesnes JPQL užklausas su JOIN sakiniais patikrinimams

PAPILDOMA UŽDUOTIS #4

- vėl nusikopijuokite React ir Spring Boot aplikacijas
- įjunkite spring boot aplikacijoje saugumą taip, kaip aprašyta skaidrėse
- apsaugokite su anotacijomis egzistuojančius Rest servisus
- React susikurkite prisijungimo formą bei atsijungimo mygtuką
- prisidėkite keletą naudotojų su skirtingomis rolėmis
- pabandykite gauti forbidden ir accesss denied klaidas
- egzamino praktinėje užduotyje Spring Security nebus

KITOJE PASKAITOJE

Praktika