From 0 to Spring Security 4.0

Rob Winch
@rob_winch

Agenda

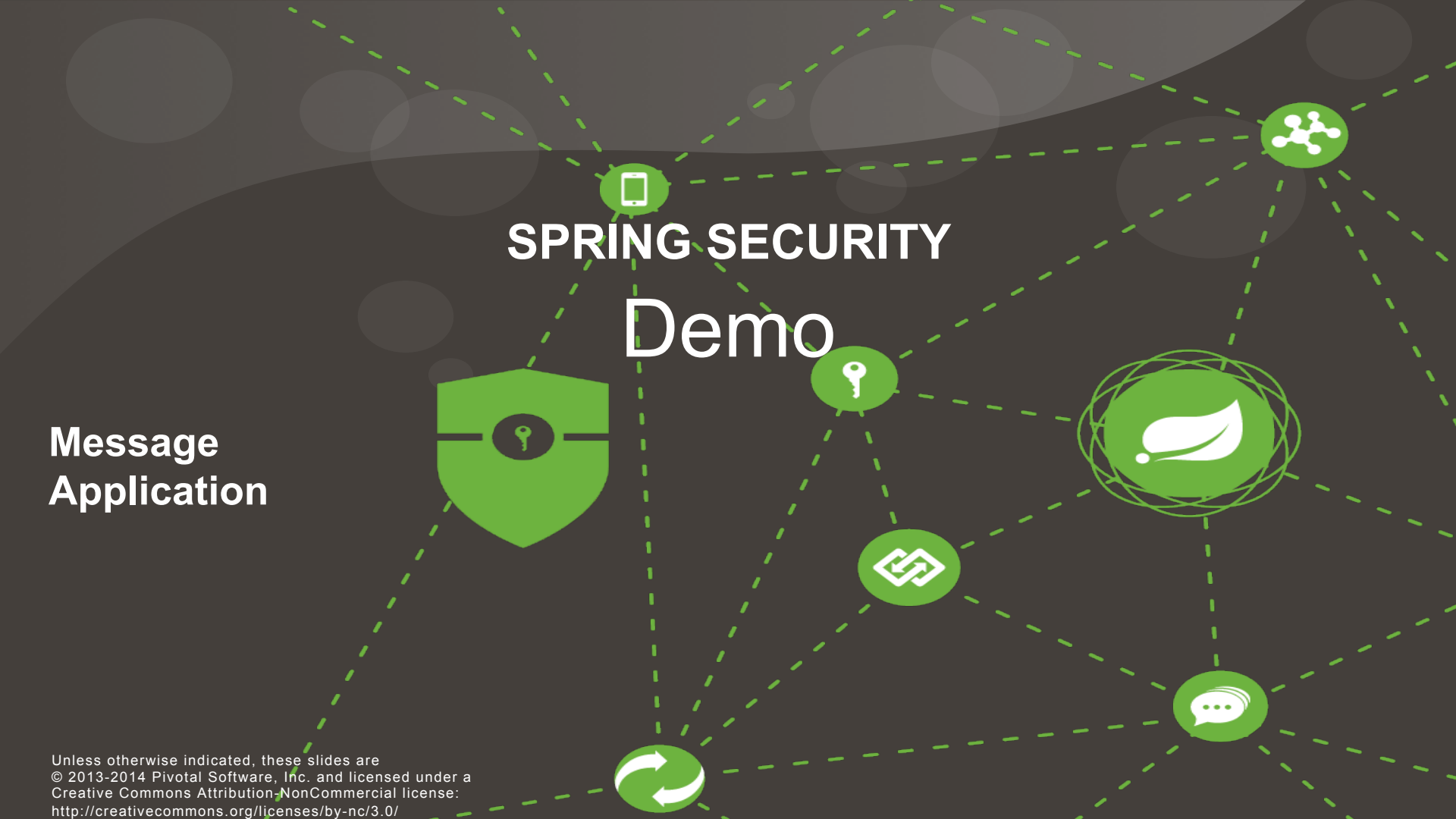
- Introductions
- Hello Spring Security (Java Config)
- Custom Authentication
- Spring Data Integration
- Testing Support
- WebSocket Support
- White Hat Hacker

About Me

- Open Source fanatic
- Spring Security & Spring Project Lead
- Committer on Spring Framework
- Co-author of Spring Security 3.1 book
- Twitter @rob_winch

What is Spring Security?

- Comprehensive support for Authentication And Authorization
- Protection against common attacks
- Servlet API Integration
- Optional integration with Spring MVC
- Optional Spring Data Integration
- WebSocket Support



SPRING SECURITY

Demo

Message Application

web.xml

```
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>
org.springframework.web.filter.DelegatingFilterProxy
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Hello Java Configuration – Replaces web.xml


```
public class SecurityWebInitializer
        extends AbstractSecurityWebApplicationInitializer {
        // optionally override methods
}
```

Hello Java Configuration – WebSecurityConfig

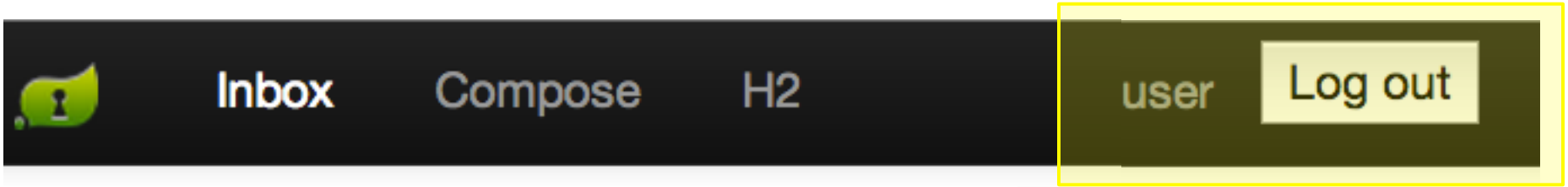

```
@Configuration
@EnableWebMvcSecurity
public class WebSecurityConfig
    extends WebSecurityConfigurerAdapter {
...
}
```

Hello Java Configuration – WebSecurityConfig


```
@Autowired
public void configureGlobal(
        AuthenticationManagerBuilder auth) throws Exception {
  auth
    .inMemoryAuthentication()
      .withUser("admin”)
        .password("password”)
        .roles("ADMIN","USER")
        .and()
      .withUser("user")
        .password("password")
        .roles("USER");
}
```

Hello Java Configuration

Hello Java Configuration



Hello Java Configuration


```
<div th:with="currentUser=$
{#httpServletRequest.userPrincipal?.name}">
  <div th:if="${currentUser != null}">
      <form th:action="@{/logout}" method="post”>
          <input type="submit" value="Log out" />
      </form>
      <p th:text="${currentUser}”>
        sample_user
      </p>
</div>
```

Hello Java Configuration

```
<div th:with="currentUser=${#httpServletRequest.userPrincipal?.name}">
  <div th:if="${currentUser != null}">
    ...
  </div>
</div>
```

```
public interface HttpServletRequest … {
  Principal getUserPrincipal();
  ...
}
```

Hello Java Configuration

```
<div th:with="currentUser=$
{#httpServletRequest.userPrincipal?.name}">
  <div th:if="${currentUser
```

```
public interface Principal … {
  String getName();
  ...
}
```

```
</div>
```

Hello Java Configuration


```
<div th:with="currentUser=$
{#httpServletRequest.userPrincipal?.name}">
  <div th:if="${currentUser != null}">
      <form th:action="@{/logout}" method="post”>
          <input type="submit" value="Log out" />
      </form>
      <p th:text="${currentUser}”>
        sample_user
      </p>
 </div>
</div>
```

Custom Log in Form

Java Configuration


```
@Override
protected void configure(HttpSecurity http)
        throws Exception {
  http
    .authorizeRequests()
        .anyRequest().authenticated()
        .and()
    .formLogin().and()
    .httpBasic();
}
```

Java Configuration


```
http
    .authorizeRequests()
        .anyRequest().authenticated()
        .and()
    .formLogin().and()
    .httpBasic();
```

```
<http use-expressions="true">
    <intercept-url pattern="/**" access="authenticated"/>
    <form-login />
    <http-basic />
</http>
```

Java Configuration


```
http
   .authorizeRequests()
     .anyRequest().authenticated()
     .and()
   .formLogin()
      .loginPage("/login”)
      .permitAll()
      .and()
   .logout()
         .permitAll();
```

Java Configuration


```
http
   .authorizeRequests()
      .antMatchers("/resources/**”).permitAll()
      .anyRequest().authenticated()
      .and()
   .formLogin()
       .loginPage("/login”)
       .permitAll()
       .and()
   .logout()
         .permitAll();
```

Java Configuration


```
<form th:action="@{/login}" method="post">
    <label for="username">Username</label>
    <input type="text" id="username"
name="username"/>
    <label for="password">Password</label>
    <input type="password" id="password"
name="password"/>
    <button type="submit">Log in</button>
</form>
```

Java Configuration


```
<form th:action="@{/login}" method="post">
    <label for="username">Username</label>
    <input type="text" id="username"
name="username"/>
    <label for="password">Password</label>
    <input type="password" id="password"
name="password"/>
    <button type="submit">Log in</button>
</form>
```

Java Configuration

```
<form th:action="@{/login}" method="post">
    <label for="username
    <input type="text"
name="username"/>
    <label for="pas
    <input type="passw
name="password"/>
    <button type="submit">Log in</button>
</form>
```

```
http
    ….
    .formLogin()
        .loginPage("/login”)
```

Custom Authentication

Java Configuration – Custom Authentication


```
public interface UserDetailsService {
  UserDetails loadUserByUsername(String username)
      throws UsernameNotFoundException;
}
```

Java Configuration – Custom Authentication


```
public interface UserDetails extends Serializable {
    Collection<? extends GrantedAuthority>
getAuthorities();
    String getPassword();
    String getUsername();
    boolean isAccountNonExpired();
    boolean isAccountNonLocked();
    boolean isCredentialsNonExpired();
    boolean isEnabled();
}
```

Java Configuration – Custom Authentication


```
@Entity
public class User implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String firstName;
    private String lastName;
    private String email;
    private String password;
    ...
}
```

Java Configuration – Custom Authentication


```
pubic class CustomUserDetails extends User
    implements UserDetails {
  public CustomUserDetails(User u) {
    super(user);
  }
  public Collection getAuthorities() {
    return AuthorityUtils.createAuthorityList("ROLE_USER");
  }
  public String getUsername() {
    return getEmail();
  }
  public boolean isEnabled() { return true; }
  ...
```

Java Configuration – Custom Authentication


```
public UserDetails loadUserByUsername(String username)
        throws UsernameNotFoundException {
    User user = userRepository.findByEmail(username);
    if(user == null) {
        throw new UsernameNotFoundException(…);
    }
    return new CustomUserDetails(user);
}
```

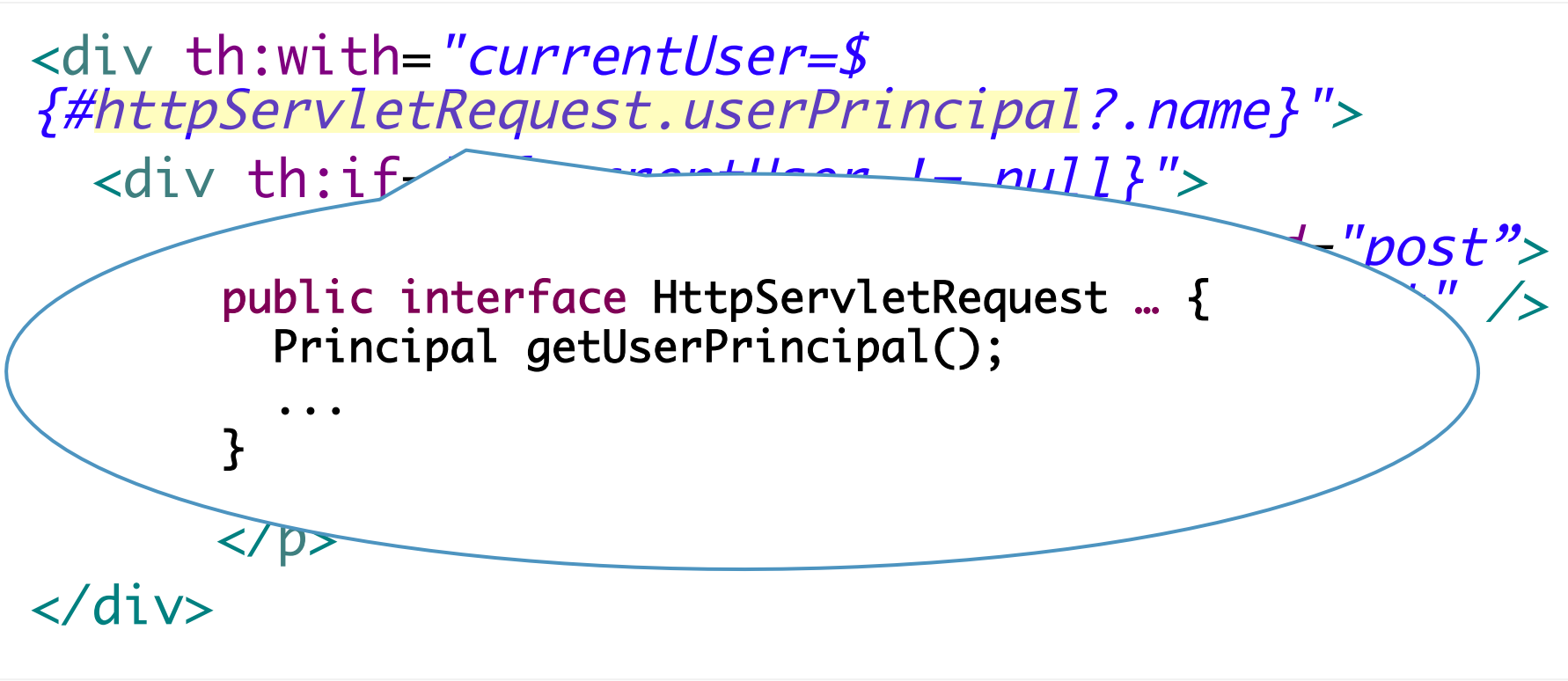
Java Configuration – Custom Authentication


```
@Autowired
public void configureGlobal(
    AuthenticationManagerBuilder auth,
    UserDetailsService userDetailsService)
      throws Exception {
  auth
    .userDetailsService(userDetailsService);
}
```

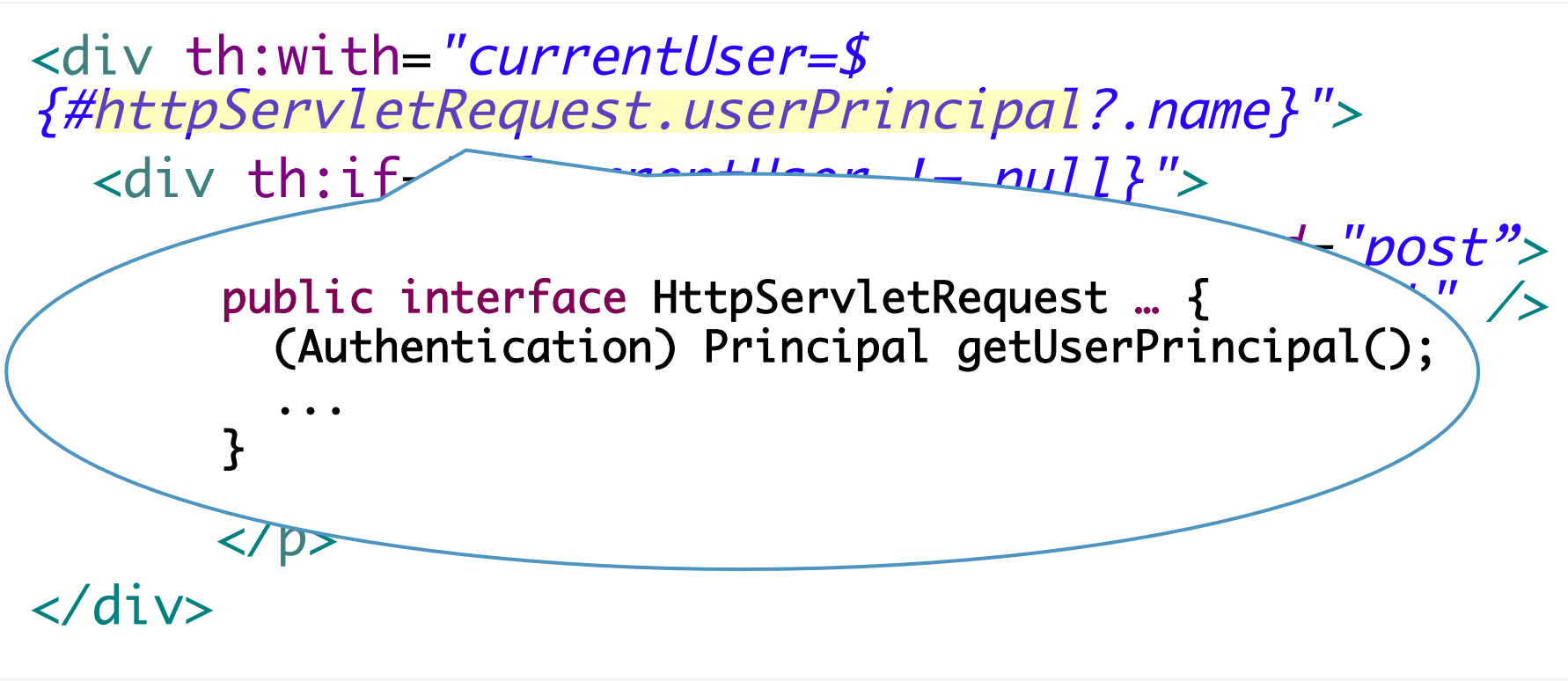
Java Configuration – Custom Authentication


```
<div th:with="currentUser=$
{#httpServletRequest.userPrincipal?.name}">
  <div th:if="${currentUser != null}">
      <form th:action="@{/logout}" method="post”>
          <input type="submit" value="Log out" />
      </form>
      <p th:text="${currentUser}”>
        sample_user
      </p>
</div>
```

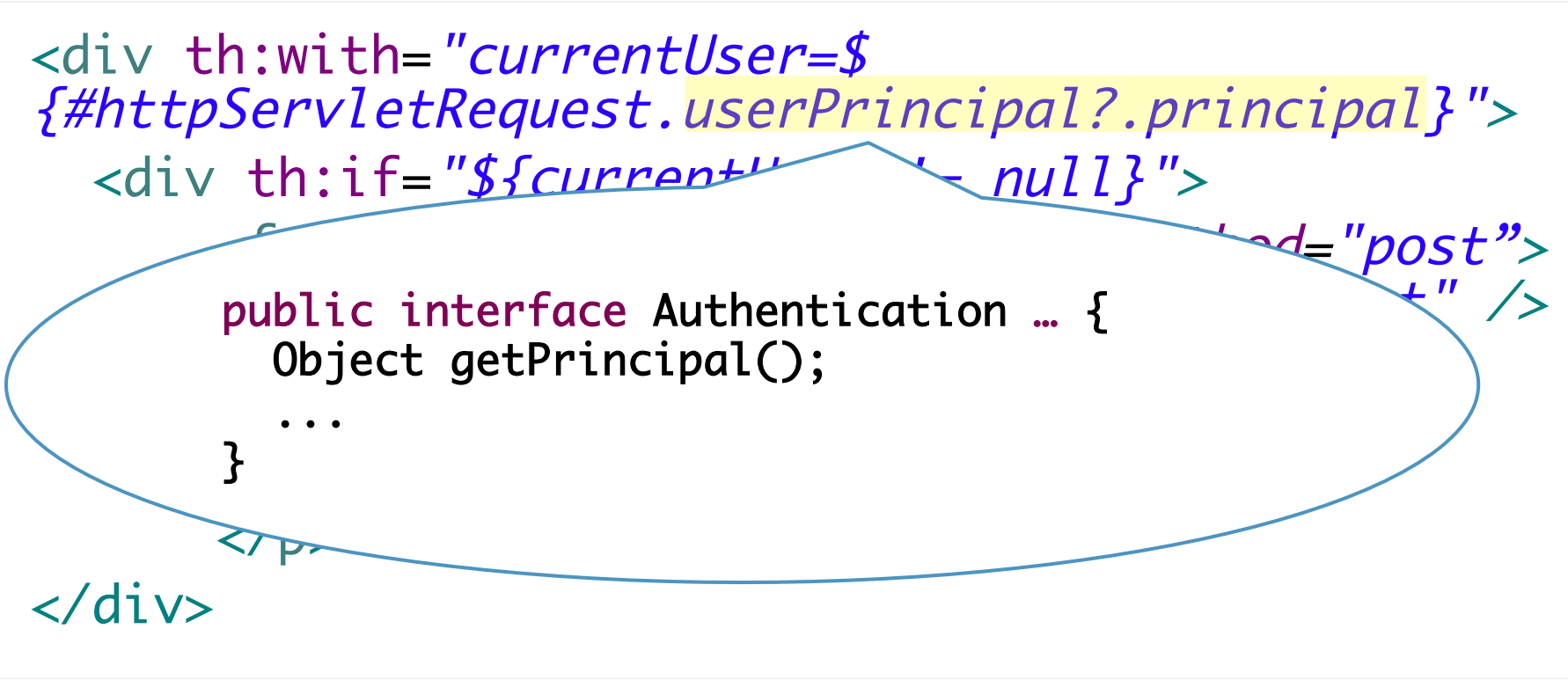
Java Configuration – Custom Authentication



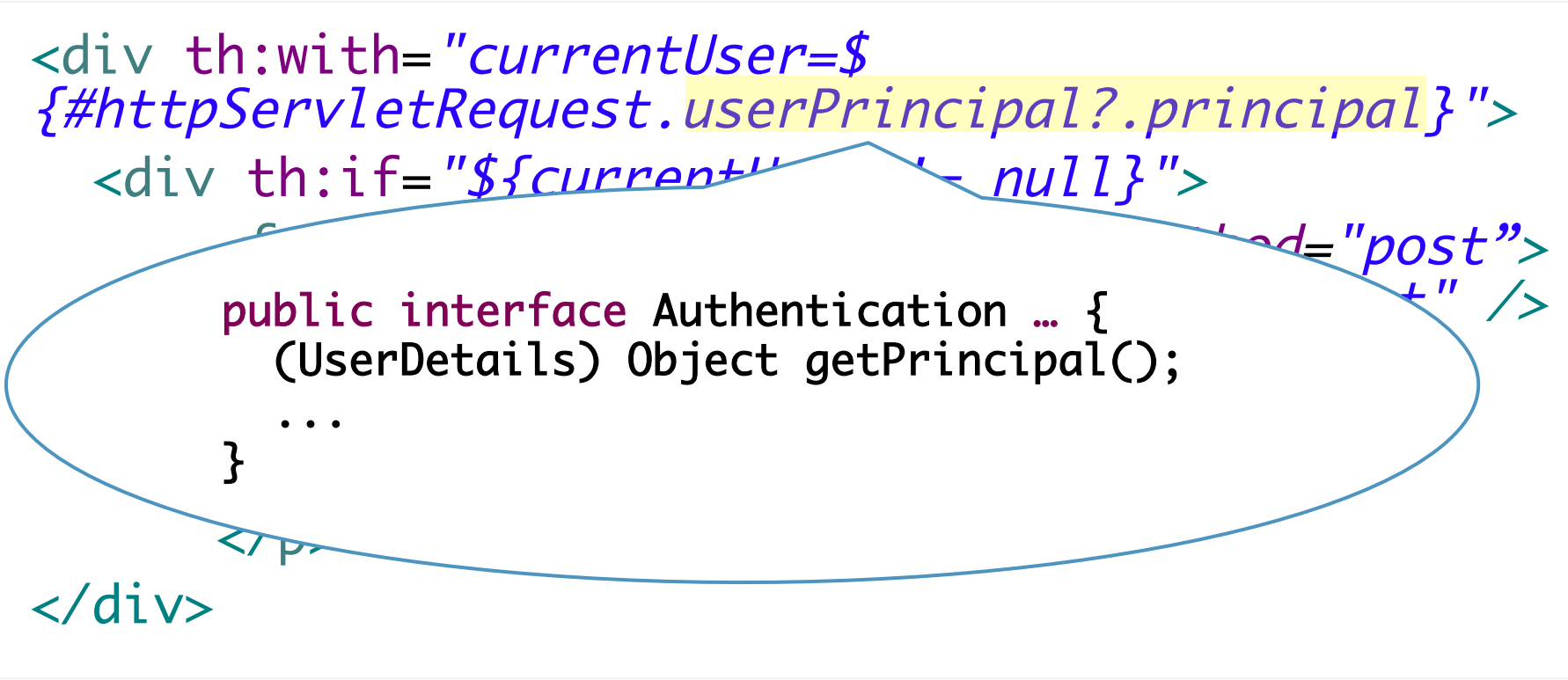
Java Configuration – Custom Authentication



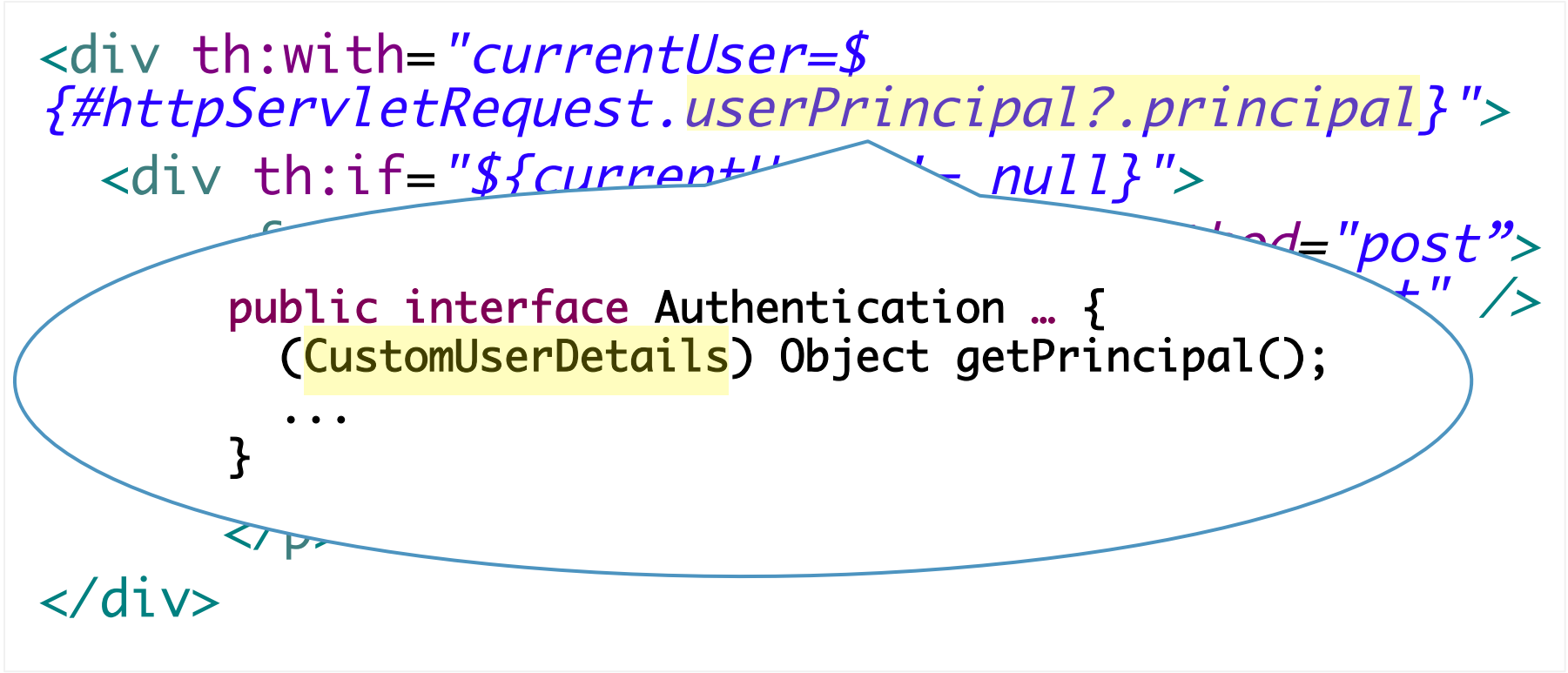
Java Configuration – Custom Authentication



Java Configuration – Custom Authentication



Java Configuration – Custom Authentication



Java Configuration – Custom Authentication

```
public class CustomUserDetails … {
  String getFirstName();
  ...
}
```

```
<di
 
  <d
    <form th:action=" e                method="post”>
      <input type="subm            lue="Log out" />
    </form>
    <p th:text="${currentUser.firstName}”>
      sample_user
    </p>
</div>
```

Java Configuration – Custom Authentication


```
@RequestMapping(method=RequestMethod.GET)
public ModelAndView list() {
    SecurityContext ctx =
        SecurityContextHolder.getContext();
    Authentication authentication =
ctx.getAuthentication();
    User custom = authentication == null ?
        null : (User) authentication.getPrincipal();


    ...
}
```

Java Configuration – Custom Authentication


```
@RequestMapping(method=RequestMethod.GET)
public ModelAndView list(Authentication
authentication) {
    User custom = authentication == null ?
        null : (User)
authentication.getPrincipal();


  ...
}
```

Java Configuration – Custom Authentication


```
@RequestMapping(method=RequestMethod.GET)
public ModelAndView list(
    @AuthenticationPrincipal User
currentUser) {

...
}
```

Java Configuration – Custom Authentication


```
@Target(ElementType.PARAMETER)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@AuthenticationPrincipal
public @interface CurrentUser { }
```

Java Configuration – Custom Authentication


```
@RequestMapping(method=RequestMethod.GET)
public ModelAndView list(
    @CurrentUser User currentUser) {

Iterable<Message> messages =

messageRepository.findByToId(currentUser.getId())
;
  ...
}
```

Spring Security / Spring Data

SpEL Support

Spring Security / Spring Data


```
@Bean
public SecurityEvaluationContextExtension
      securityEvaluationContextExtension() {
  return new SecurityEvaluationContextExtension();
}
```

Spring Security / Spring Data


```
public interface MessageRepository
    extends CrudRepository<Message, Long> {

  @Query("select m from Message m where m.to.id = " +
         "?#{principal.id}")
  Iterable<Message> findAllToCurrentUser();
}
```

Spring Security / Spring Data


```
public interface MessageRepository
    extends CrudRepository<Message, Long> {

  @Query("select m from Message m where m.to.id = " +
         "?#{hasRole('ROLE_ADMIN') ? '%' :
principal.id}”)
  Iterable<Message> findAll();
}
```

Spring Security / Spring Data

In the year 2000….

```
@EnableAclSecurity
public interface SecuredMessageRepository
    extends MessageRepository {}
```

Password Storage

Password Storage


```
auth
    .userDetailsService(userDetailsService)
        .passwordEncoder(new BCryptPasswordEncoder());
```

CSRF Protection

SPRING SECURITY

Demo

CSRF Protection

CSRF Protection


```
POST /sample/110 HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
Cookie: JSESSIONID=8B00C27E0962E363CBAC814F19E51C1D

_method=delete&_csrf=7d281f9f-55d9-4663-88f8-42827f3d2c12
```

CSRF Protection


```
POST /sample/110 HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
Cookie: JSESSIONID=8B00C27E0962E363CBAC814F19E51C1D

_method=delete&_csrf=7d281f9f-55d9-4663-88f8-42827f3d2c12
```

CSRF Protection

> When do I use CSRF protection?

CSRF Protection

> ... but my application uses JSON

CSRF Protection


```
<form ... method="post" enctype="text/plain">
  <input type='hidden'
     name='{"summary":"Hi", … "ignore_me":"'
value='test"}'
  />
</form>
```

CSRF Protection


```
{
  "summary": "Hi",
  "message": "New Message",
  "to": "luke@example.com",
  "ignore_me": "=test"
}
```

CSRF Protection

> … but my application is stateless

CSRF Protection

```
POST /sample/100 HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
Authorization: Basic cm9iQGV4YW1wbGUuY29tOnBhc3N3b3Jk

_method=delete
```

CSRF Protection

> …and I use a custom header for authentication and ignore cookies

CSRF Protection

- Use proper HTTP Verbs
- Configure CSRF Protection
- Include the CSRF Token

CSRF Protection – Providing the Token

```
<form ... method="post">
  ...
  <input type="hidden"
    name="${_csrf.parameterName}"
    value="${_csrf.token}"/>
</form>
```

CSRF Protection – Providing the Token

```
<form ... method="post">
  ...
  <sec:csrfInput />
</form>
```

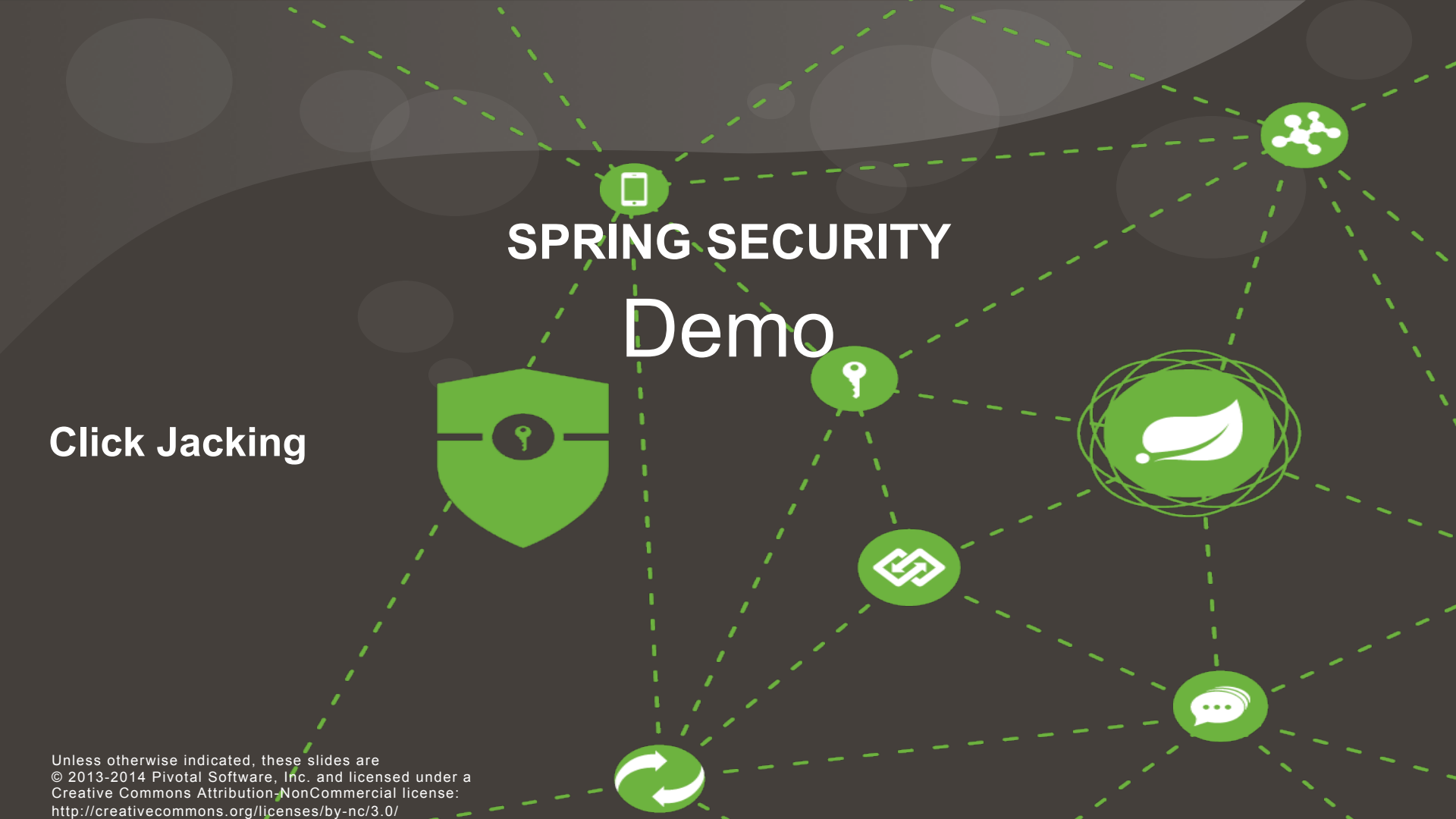
CSRF Protection – Providing the Token

```
<form:form … method="post”>
  ...
</form:form>
```

CSRF Protection – Providing the Token

```
<form ... method="post">
  ...
  <input type="hidden" name="_csrf"
value="f81d4fae-…"/>
</form>
```

Security HTTP Response Headers



SPRING SECURITY

Demo

Click Jacking

Security HTTP Response Headers


```
HTTP/1.1 200 OK
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
```

Security HTTP Response Headers


```
HTTP/1.1 200 OK
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
```

Test Support

Testing Support


```
@Before
public void setup() {
  Authentication auth =
new TestingAuthenticationToken("user",”pass","ROLE_USER");
  SecurityContext ctx =
    SecurityContextHolder.getContext();
  ctx.setAuthentication(auth);
  SecurityContextHolder.setContext(ctx);
}
@After
public void cleanup() {
  SecurityContextHolder.clearContext();
}
```

Testing Support

```
UserDetails user = ...
List<GrantedAuthority> roles =
    AuthorityUtils.createAuthorityList("ROLE_USER");
Authentication auth =
    new UsernamePasswordAuthenticationToken(user,”pass",
roles);
SecurityContext ctx =
    SecurityContextHolder.getContext();
ctx.setAuthentication(auth);
```

Testing Support

```
User user = ...
List<GrantedAuthority> roles =
    AuthorityUtils.createAuthorityList("ROLE_USER");
Authentication auth =
    new UsernamePasswordAuthenticationToken(user,”pass",
roles);
SecurityContext ctx =
    SecurityContextHolder.getContext();
ctx.setAuthentication(auth);
```

Testing Support

```
...
@WithMockUser
public class SecurityMethodTests {

    ...
}
```

Testing Support

```
...
public class SecurityMethodTests {
   @Test
   @WithMockUser
   public void findAllMessages() {
      ...
   }
}
```

Testing Support

```
...
public class SecurityMethodTests {
  @Test

@WithMockUser(username="admin",roles="ADMIN”)
  public void findAllMessages() {
    repository.findAll();
  }
}
```

Testing Support


```
...
public class SecurityMethodTests {
   @Test
   @WithUserDetails(”rob@example.com")
   public void findAllMessages() {
      repository.findAll();
   }
}
```

Testing Support


```
@Target({ ElementType.METHOD, ElementType.TYPE })
@Retention(RetentionPolicy.RUNTIME)
@Inherited
@Documented
@WithSecurityContext(factory =
WithCustomUserSecurityContextFactory.class)
public @interface WithCustomUser {
    String email() default "rob@example.com";
    String firstName() default "Rob";
    String lastName() default "Winch";
    long id() default 0L;
}
```

Testing Support


```
public class WithCustomUserSecurityContextFactory
    implements WithSecurityContextFactory<WithCustomUser> {
  public SecurityContext
      createSecurityContext(WithCustomUser customUser) {
    User principal = new User();
    principal.setEmail(customUser.email());
    ...
    return ctx;
  }
}
```

Testing Support

```
...
public class SecurityMethodTests {
   @Test
   @WithCustomUser
   public void findAllMessages() {
      repository.findAll();
   }
}
```

Testing Support

```
...
public class SecurityMethodTests {
   @Test
   @WithCustomUser(id=1,email="luke@example.com")
   public void findAllMessages() {
      repository.findAll();
   }
}
```

Testing Support

> …what about Spring Test MVC?

Testing Support

```
...
public class SecurityMockMvcTests {
   @Before
   public void setup() {
      mvc = MockMvcBuilders
          .webAppContextSetup(context)
          .apply(springSecurity())
          .build();
      }
```

Testing Support


```
@Test
@WithCustomUser
public void inboxShowsOnlyTo() throws Exception {
   ...
}
```

Testing Support

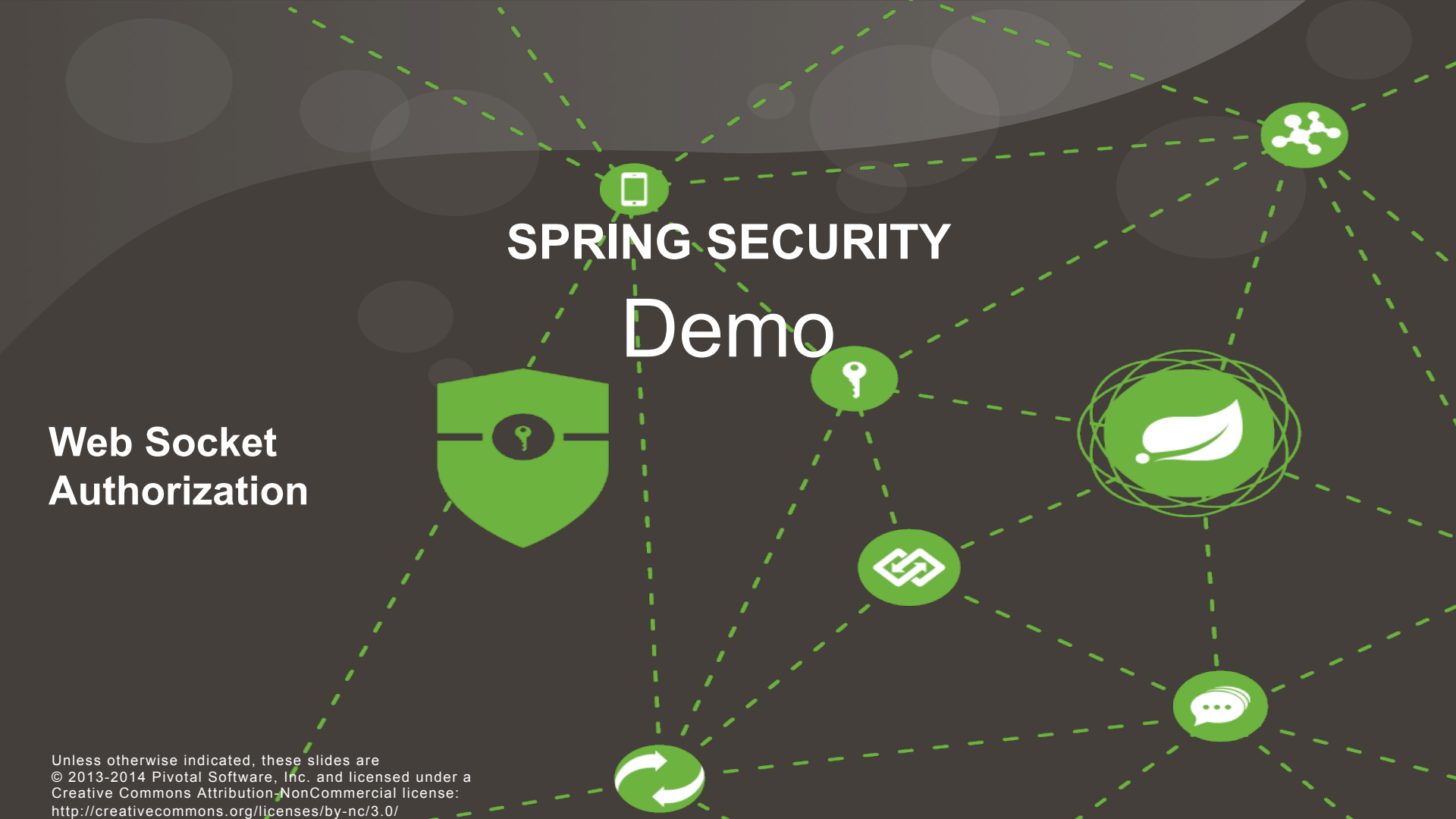

```
@Test
@WithCustomUser(id=1,email="luke@example.com")
public void inboxShowsOnlyTo() throws Exception {
   ...
}
```

Testing Support


```
@Test
@WithCustomUser
public void compose() throws Exception {
  MockHttpServletRequestBuilder compose = post("/")
      .param("summary", "Hello Luke")
      .param("message", "This is my message")
      .with(csrf());

  mvc
    .perform(compose)
    .andExpect(status().is2xxSuccessful());
}
```

WebSocket Security

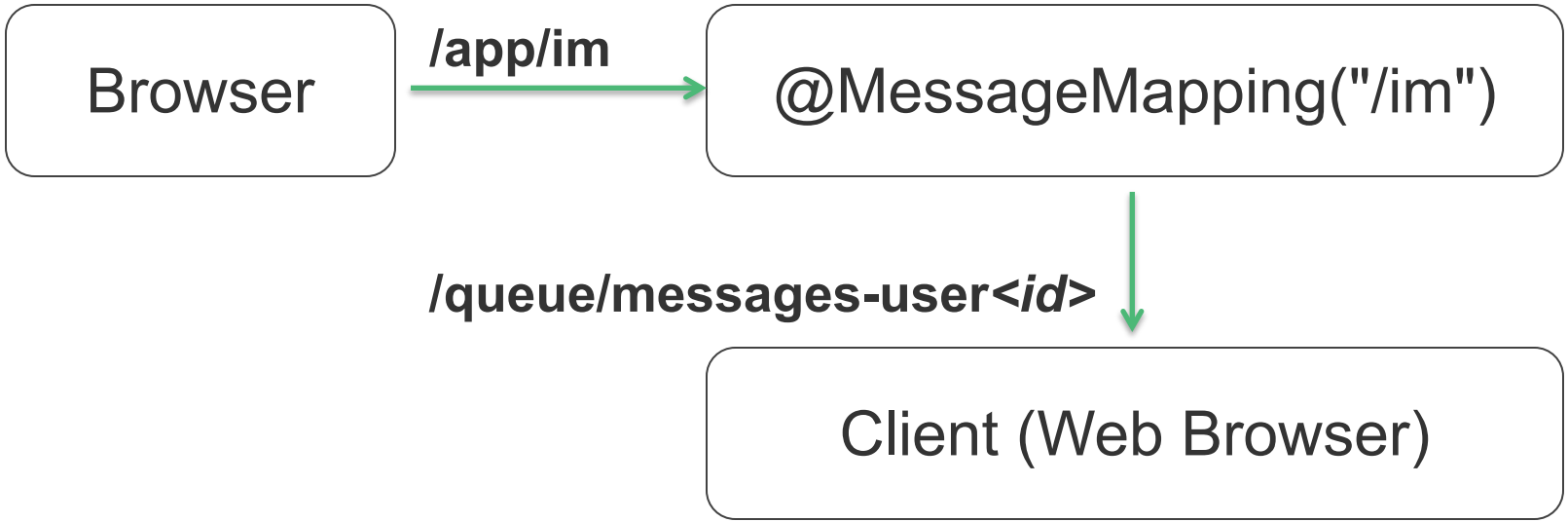


SPRING SECURITY

Demo

Web Socket Authorization

WebSocket Authorization



WebSocket Authorization


```
@Configuration
public class WebSocketSecurityConfig extends
    AbstractSecurityWebSocketMessageBrokerConfigurer {
```

WebSocket Authorization


```
protected void configure(
    MessageSecurityMetadataSourceRegistry messages) {
  messages
    .matchers(message("/topic/**","/queue/**")).denyAll()
    .anyMessage().hasRole("USER");
}
```

WebSocket Authorization


```
// avoid processing outbound channel
public void configureClientOutboundChannel(
    ChannelRegistration registration) {}
```

WebSocket Security

Spring Session

Learn More. Stay Connected.

- Source http://github.com/rwinch/spring-security-0-to-4.0
- http://spring.io/spring-security
- Twitter: @rob_winch

Security for Microservices with Spring & OAuth2 – 4:30 Today