OpenAEV-Platform / **openaev** Public

Code Issues 625 Pull requests 67 Discussions Actions Projects

Commit `c09a4e7`

richard-julien committed on Sep 27, 2022 Verified

`[api/front] Be able to reset the password through a link on the login page (#135)`

`master` · 2.3.3 ••• 1.0.0

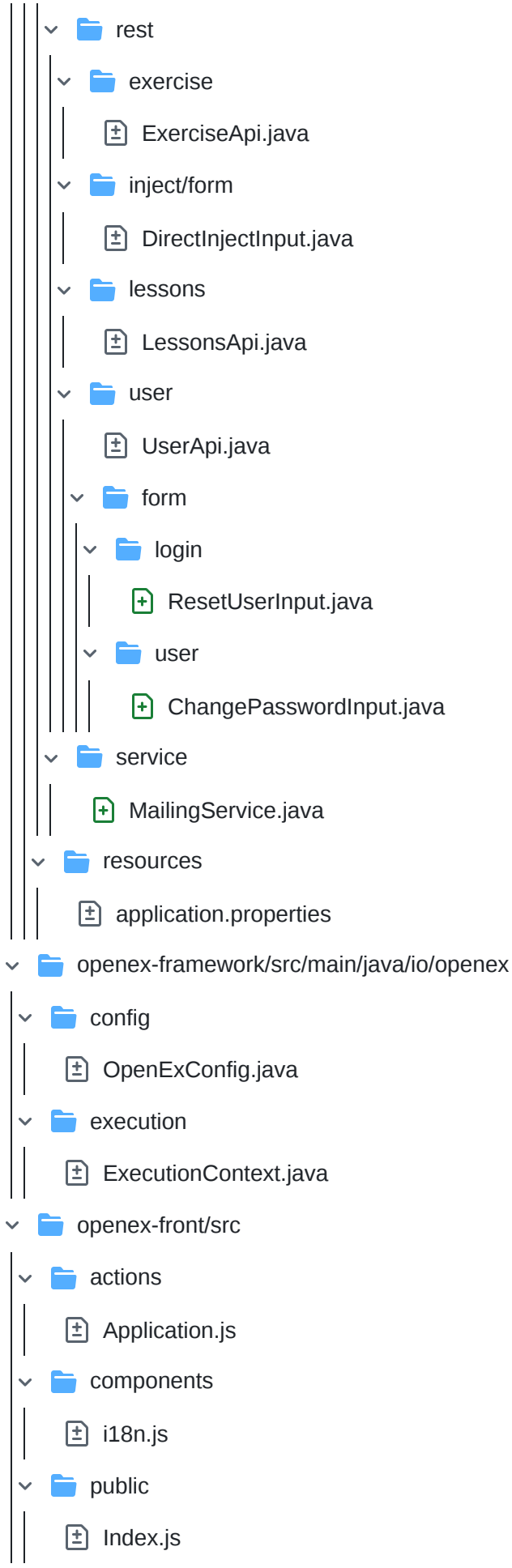
1 parent `b76c3c1` commit `c09a4e7`

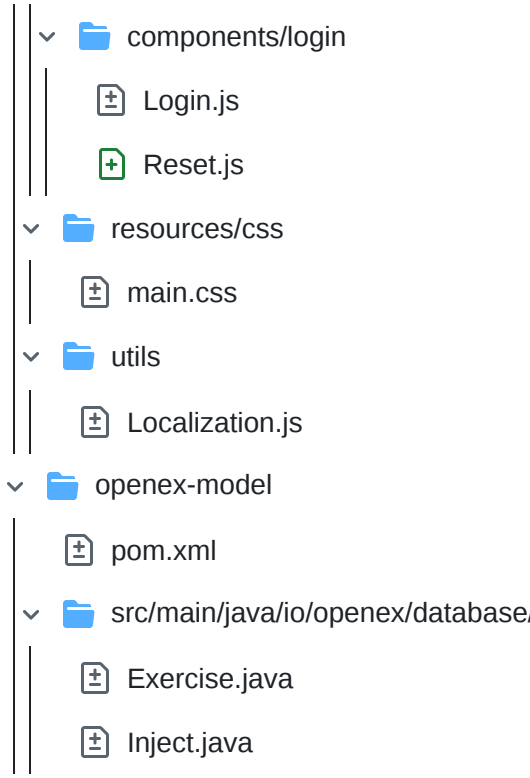
26 files changed +426 -107

Top

Filter files…

- openex-api
  - pom.xml
  - src/main
    - java/io/openex
      - config
        - AppSecurityConfig.java
      - injects
        - challenge
          - ChallengeExecutor.java
        - email
          - EmailExecutor.java
          - model
            - EmailContent.java
        - media
          - MediaExecutor.java


rest
exercise
ExerciseApi.java
inject/form
DirectInjectInput.java
lessons
LessonsApi.java
user
UserApi.java
form
login
ResetUserInput.java
user
ChangePasswordInput.java
service
MailingService.java
resources
application.properties
openex-framework/src/main/java/io/openex
config
OpenExConfig.java
execution
ExecutionContext.java
openex-front/src
actions
Application.js
components
i18n.js
public
Index.js



Search within code

openex-api/pom.xml

```
@@ -154,6 +154,11 @@
154 154            <artifactId>commons-email</artifactId>
155 155            <version>1.5</version>
156 156        </dependency>
    157 +        <dependency>
    158 +            <groupId>org.apache.commons</groupId>
    159 +            <artifactId>commons-collections4</artifactId>
    160 +            <version>4.4</version>
    161 +        </dependency>
157 162        <dependency>
158 163            <groupId>org.flywaydb</groupId>
159 164            <artifactId>flyway-core</artifactId>
```

...ava/io/openex/config/AppSecurityConfig.java

```
@@ -63,6 +63,7 @@ protected void configure(HttpSecurity http) throws
Exception {
63 63                .antMatchers("/api/player/**").permitAll()
64 64                .antMatchers("/api/settings").permitAll()
```

```
65  65                  /**/.antMatchers("/api/login").permitAll()
    66  +               /**/.antMatchers("/api/reset/**").permitAll()
66  67                  /**/.antMatchers("/api/**").authenticated()
67  68                  .and()
68  69                  .logout()
```

...ex/injects/challenge/ChallengeExecutor.java

```
@@ -79,7 +79,7 @@ public List<Expectation> process(Execution execution,
ExecutableInject injection
79  79                  List<Document> documents =
injection.getInject().getDocuments().stream()
80  80
.filter(InjectDocument::isAttached).map(InjectDocument::getDocument).toList();
81  81                  List<DataAttachment> attachments = resolveAttachments(execution,
injection, documents);
82      -               String message = content.buildMessage(injection.getInject(),
imapEnabled);
    82  +               String message = content.buildMessage(injection, imapEnabled);
83  83                  boolean encrypted = content.isEncrypted();
84  84                  users.stream().parallel().forEach(userInjectContext -> {
85  85                      try {
```

.../io/openex/injects/email/EmailExecutor.java

```
@@ -71,7 +71,7 @@ public List<Expectation> process(Execution execution,
ExecutableInject injection
71  71          List<DataAttachment> attachments = resolveAttachments(execution,
injection, documents);
72  72          String inReplyTo = content.getInReplyTo();
73  73          String subject = content.getSubject();
74      -       String message = content.buildMessage(inject, imapEnabled);
    74  +       String message = content.buildMessage(injection, imapEnabled);
75  75          boolean mustBeEncrypted = content.isEncrypted();
76  76          // Resolve the attachments only once
77  77          List<ExecutionContext> users = injection.getUsers();
@@ -83,7 +83,7 @@ public List<Expectation> process(Execution execution,
ExecutableInject injection
```

```
83  83              users = users.stream().peek(context -> context.put("document_uri",
    buildDocumentUri(context, inject))).toList();
84  84          }
85  85          Exercise exercise = injection.getSource().getExercise();
86      -       String replyTo = exercise.getReplyTo();
    86  +       String replyTo = exercise != null ? exercise.getReplyTo() :
    openExConfig.getDefaultMailer();
87  87          //noinspection SwitchStatementWithTooFewBranches
88  88          switch (contract.getId()) {
89  89              case EMAIL_GLOBAL -> sendMulti(execution, users, replyTo, inReplyTo,
    subject, message, attachments);
```

...penex/injects/email/model/EmailContent.java

```
    @@ -1,7 +1,7 @@
1   1   package io.openex.injects.email.model;
2   2
3   3   import com.fasterxml.jackson.annotation.JsonProperty;
4       - import io.openex.database.model.Inject;
    4   + import io.openex.execution.ExecutableInject;
5   5   import org.springframework.util.StringUtils;
6   6
7   7   import java.util.Objects;
    @@ -58,24 +58,21 @@ public void setEncrypted(boolean encrypted) {
58  58          this.encrypted = encrypted;
59  59      }
60  60
61      -   public String buildMessage(Inject inject, boolean imapEnabled) {
    61  +   public String buildMessage(ExecutableInject injection, boolean imapEnabled)
    {
62  62          // String footer = inject.getFooter();
63      -       String header = inject.getHeader();
    63  +       String header = injection.getInject().getHeader();
64  64          StringBuilder data = new StringBuilder();
65  65          if (StringUtils.hasLength(header)) {
66  66              data.append(HEADER_DIV).append(header).append(END_DIV);
67  67          }
68  68          data.append(START_DIV).append(body).append(END_DIV);
69      -       // if (StringUtils.hasLength(footer)) {
```

```
70      -         //    data.append(FOOTER_DIV).append(footer).append(END_DIV);
71      -         // }
72  69            // If imap is enable we need to inject the id marker
73      -         if (imapEnabled && !inject.isDirect()) {
        70  +         if (injection.isRuntime() && imapEnabled) {
74  71                    data.append(START_DIV)
75  72                            .append("<br/><br/><br/><br/>")
76  73                            .append("----------------------------------------------------
    ------------------------------<br/>")
77  74                            .append("OpenEx internal information, do not remove!<br/>")
78      -                     .append("[inject_id=").append(inject.getId()).append("]
    <br/>")
        75  +                 .append("
    [inject_id=").append(injection.getInject().getId()).append("]<br/>")
79  76                            .append("----------------------------------------------------
    ------------------------------<br/>")
80  77                            .append(END_DIV);
81  78            }
```

.../io/openex/injects/media/MediaExecutor.java

```
    @@ -79,7 +79,7 @@ public List<Expectation> process(Execution execution,
    ExecutableInject injection
79  79                        List<Document> documents =
    injection.getInject().getDocuments().stream()
80  80
    .filter(InjectDocument::isAttached).map(InjectDocument::getDocument).toList();
81  81                        List<DataAttachment> attachments =
    resolveAttachments(execution, injection, documents);
82      -                     String message = content.buildMessage(injection.getInject(),
    imapEnabled);
        82  +                 String message = content.buildMessage(injection,
    imapEnabled);
83  83                        boolean encrypted = content.isEncrypted();
84  84                        users.stream().parallel().forEach(userInjectContext -> {
85  85                            try {
```

...va/io/openex/rest/exercise/ExerciseApi.java

```
@@ -1,6 +1,7 @@
1  1  package io.openex.rest.exercise;
2  2
3  3  import com.fasterxml.jackson.databind.ObjectMapper;
   4  + import io.openex.config.OpenExConfig;
4  5  import io.openex.database.model.*;
5  6  import io.openex.database.model.Exercise.STATUS;
6  7  import io.openex.database.repository.*;
@@ -21,6 +22,7 @@
21 22  import org.springframework.web.multipart.MultipartFile;
22 23  import reactor.util.function.Tuples;
23 24
   25 + import javax.annotation.Resource;
24 26  import javax.annotation.security.RolesAllowed;
25 27  import javax.servlet.http.HttpServletResponse;
26 28  import javax.transaction.Transactional;
@@ -61,6 +63,9 @@ public class ExerciseApi extends RestBehavior {
61 63
62 64      @Value("${openex.mail.imap.username}")
63 65      private String imapUsername;
   66 +
   67 +     @Resource
   68 +     private OpenExConfig openExConfig;
64 69      // endregion
65 70
66 71      // region repositories
@@ -292,6 +297,8 @@ public Exercise createExercise(@Valid @RequestBody
    ExerciseCreateInput input) {
292 297
    exercise.setTags(fromIterable(tagRepository.findAllById(input.getTagIds())));
293 298          if (imapEnabled) {
294 299              exercise.setReplyTo(imapUsername);
   300 +         } else {
   301 +             exercise.setReplyTo(openExConfig.getDefaultMailer());
295 302          }
296 303          // Find automatic groups to grants
297 304          List<Group> groups = fromIterable(groupRepository.findAll());
```

...nex/rest/inject/form/DirectInjectInput.java

```
@@ -81,7 +81,6 @@ public Inject toInject() {
81  81          inject.setDescription(getDescription());
82  82          inject.setContent(getContent());
83  83          inject.setContract(getContract());
84      -       inject.setDirect(true);
85  84          return inject;
86  85      }
87  86  }
```

...java/io/openex/rest/lessons/LessonsApi.java

```
@@ -1,26 +1,17 @@
1   1   package io.openex.rest.lessons;
2   2
3       - import io.openex.config.OpenExConfig;
4       - import io.openex.contract.Contract;
5   3   import io.openex.database.model.*;
6   4   import io.openex.database.repository.*;
7   5   import io.openex.database.specification.LessonsAnswerSpecification;
8   6   import io.openex.database.specification.LessonsCategorySpecification;
9   7   import io.openex.database.specification.LessonsQuestionSpecification;
10      - import io.openex.execution.ExecutableInject;
11      - import io.openex.execution.ExecutionContext;
12      - import io.openex.execution.Injector;
13      - import io.openex.injects.email.EmailContract;
14      - import io.openex.injects.email.model.EmailContent;
15  8   import io.openex.rest.helper.RestBehavior;
16  9   import io.openex.rest.lessons.form.*;
17      - import io.openex.service.ContractService;
    10  + import io.openex.service.MailingService;
18  11  import org.springframework.beans.factory.annotation.Autowired;
19      - import org.springframework.context.ApplicationContext;
20  12  import org.springframework.security.access.prepost.PreAuthorize;
21  13  import org.springframework.web.bind.annotation.*;
22  14
23      - import javax.annotation.Resource;
24  15  import javax.validation.Valid;
25  16  import java.util.List;
```

```
26  17    import java.util.Optional;
          @@ -32,19 +23,19 @@
32  23
33  24    @RestController
34  25    public class LessonsApi extends RestBehavior {
35        -
36        -     @Resource
37        -     private OpenExConfig openExConfig;
38        -
39  26        private ExerciseRepository exerciseRepository;
40  27        private AudienceRepository audienceRepository;
41  28        private LessonsTemplateRepository lessonsTemplateRepository;
42  29        private LessonsCategoryRepository lessonsCategoryRepository;
43  30        private LessonsQuestionRepository lessonsQuestionRepository;
44  31        private LessonsAnswerRepository lessonsAnswerRepository;
45        -     private ContractService contractService;
46        -     private ApplicationContext context;
47  32        private UserRepository userRepository;
      33  +     private MailingService mailingService;
      34  +
      35  +     @Autowired
      36  +     public void setMailingService(MailingService mailingService) {
      37  +         this.mailingService = mailingService;
      38  +     }
48  39
49  40        @Autowired
50  41        public void setUserRepository(UserRepository userRepository) {
          @@ -81,16 +72,6 @@ public void
          setLessonsAnswerRepository(LessonsAnswerRepository lessonsAnswerRepo
81  72            this.lessonsAnswerRepository = lessonsAnswerRepository;
82  73        }
83  74
84        -     @Autowired
85        -     public void setContractService(ContractService contractService) {
86        -         this.contractService = contractService;
87        -     }
88        -
89        -     @Autowired
90        -     public void setContext(ApplicationContext context) {
91        -         this.context = context;
```

```
92        -     }
93        -
94    75          @GetMapping("/api/exercises/{exerciseId}/lessons_categories")
95    76          @PreAuthorize("isExerciseObserver(#exerciseId)")
96    77          public Iterable<LessonsCategory> exerciseLessonsCategories(@PathVariable
                  String exerciseId) {
          ↓ ↑     @@ -220,28 +201,9 @@ public void deleteExerciseLessonsQuestion(@PathVariable
                  String lessonsQuestionId
220   201         public void sendExerciseLessons(@PathVariable String exerciseId, @Valid
                  @RequestBody LessonsSendInput input) {
221   202                 Exercise exercise =
                  exerciseRepository.findById(exerciseId).orElseThrow();
222   203                 List<LessonsCategory> lessonsCategories =
                  lessonsCategoryRepository.findAll(LessonsCategorySpecification.fromExercise(exe
                  rciseId)).stream().toList();
223       -             EmailContent emailContent = new EmailContent();
224       -             emailContent.setSubject(input.getSubject());
225       -             emailContent.setBody(input.getBody());
226       -             Inject inject = new Inject();
227       -             inject.setTitle("Lessons learned campaign");
228       -             inject.setDescription("Direct inject for lessons learned
                  questionnaire");
229       -             inject.setContent(mapper.valueToTree(emailContent));
230       -             inject.setContract(EmailContract.EMAIL_DEFAULT);
231       -             inject.setUser(currentUser());
232       -             inject.setDirect(true);
233       -             Contract contract = contractService.resolveContract(inject);
234       -             if (contract == null) {
235       -                 throw new UnsupportedOperationException("Unknown inject contract "
                  + inject.getContract());
236       -             }
237       -             inject.setType(contract.getConfig().getType());
238       -             inject.setExercise(exercise);
239       -             List<ExecutionContext> userInjectContexts =
                  lessonsCategories.stream().flatMap(lessonsCategory ->
                  lessonsCategory.getAudiences().stream()
240       -                         .flatMap(audience ->
                  audience.getUsers().stream())).distinct()
241       -                 .map(user -> new ExecutionContext(openExConfig, user, inject,
                  "Direct execution")).toList();
```

```
242     -         ExecutableInject injection = new ExecutableInject(true, true, inject,
          contract, List.of(), userInjectContexts);
243     -         Injector executor = context.getBean(contract.getConfig().getType(),
          Injector.class);
244     -         executor.executeInjection(injection);
        204 +         List<User> users = lessonsCategories.stream().flatMap(lessonsCategory -
          > lessonsCategory.getAudiences().stream()
        205 +                 .flatMap(audience ->
          audience.getUsers().stream())).distinct().toList();
        206 +         mailingService.sendEmail(input.getSubject(), input.getBody(), users,
          Optional.of(exercise));
245     207       }
246     208
247     209       @GetMapping("/api/exercises/{exerciseId}/lessons_answers")
```

.../main/java/io/openex/rest/user/UserApi.java

```
          @@ -5,21 +5,26 @@
5       5     import io.openex.database.repository.OrganizationRepository;
6       6     import io.openex.database.repository.TagRepository;
7       7     import io.openex.database.repository.UserRepository;
        8   + import io.openex.rest.exception.InputValidationException;
8       9     import io.openex.rest.helper.RestBehavior;
9       10    import io.openex.rest.user.form.login.LoginUserInput;
        11  + import io.openex.rest.user.form.login.ResetUserInput;
        12  + import io.openex.rest.user.form.user.ChangePasswordInput;
10      13    import io.openex.rest.user.form.user.CreateUserInput;
11      -   - import io.openex.rest.user.form.user.UpdatePasswordInput;
12      14    import io.openex.rest.user.form.user.UpdateUserInput;
        15  + import io.openex.service.MailingService;
13      16    import io.openex.service.UserService;
        17  + import org.apache.commons.collections4.map.PassiveExpiringMap;
        18  + import org.apache.commons.lang3.RandomStringUtils;
14      19    import org.springframework.beans.factory.annotation.Autowired;
15      -   - import org.springframework.http.ResponseEntity;
16      20    import org.springframework.security.access.AccessDeniedException;
17      21    import org.springframework.web.bind.annotation.*;
18      22
19      23    import javax.annotation.Resource;
20      24    import javax.annotation.security.RolesAllowed;
```

```
21  25    import javax.transaction.Transactional;
22  26    import javax.validation.Valid;
    27  + import java.util.List;
23  28    import java.util.Optional;
24  29
25  30    import static io.openex.database.model.User.ROLE_ADMIN;
          @@ -28,14 +33,19 @@
28  33
29  34    @RestController
30  35    public class UserApi extends RestBehavior {
31      -
    36  +     PassiveExpiringMap<String, String> resetTokenMap = new PassiveExpiringMap<>
          (1000 * 60 * 10);
32  37        @Resource
33  38        private SessionManager sessionManager;
34      -
35  39        private OrganizationRepository organizationRepository;
36  40        private UserRepository userRepository;
37  41        private TagRepository tagRepository;
38  42        private UserService userService;
    43  +     private MailingService mailingService;
    44  +
    45  +     @Autowired
    46  +     public void setMailingService(MailingService mailingService) {
    47  +         this.mailingService = mailingService;
    48  +     }
39  49
40  50        @Autowired
41  51        public void setTagRepository(TagRepository tagRepository) {
          @@ -58,18 +68,67 @@ public void setUserRepository(UserRepository
          userRepository) {
58  68        }
59  69
60  70        @PostMapping("/api/login")
61      -     public ResponseEntity<User> login(@Valid @RequestBody LoginUserInput input)
          {
    71  +     public User login(@Valid @RequestBody LoginUserInput input) {
62  72            Optional<User> optionalUser =
          userRepository.findByEmail(input.getLogin());
63  73            if (optionalUser.isPresent()) {
```

```
64  74                  User user = optionalUser.get();
65  75                  if (userService.isUserPasswordValid(user, input.getPassword())) {
66  76                      userService.createUserSession(user);
67      -                   return ResponseEntity.ok().body(user);
        77  +               return user;
68  78                  }
69  79              }
70  80              throw new AccessDeniedException("Invalid credentials");
71  81          }
72  82
        83  +       @PostMapping("/api/reset")
        84  +       public void passwordReset(@Valid @RequestBody ResetUserInput input) {
        85  +           Optional<User> optionalUser =
                    userRepository.findByEmail(input.getLogin());
        86  +           if (optionalUser.isPresent()) {
        87  +               User user = optionalUser.get();
        88  +               String resetToken = RandomStringUtils.randomNumeric(8);
        89  +               String username = user.getName() != null ? user.getName() :
                    user.getEmail();
        90  +               if ("fr".equals(input.getLang())) {
        91  +                   String subject = resetToken + " est votre code de récupération
                    de compte OpenEx";
        92  +                   String body = "Bonjour " + username + ",</br>" +
        93  +                           "Nous avons reçu une demande de réinitialisation de
                    votre mot de passe OpenEx.</br>" +
        94  +                           "Entrez le code de réinitialisation du mot de passe
                    suivant : " + resetToken;
        95  +                   mailingService.sendEmail(subject, body, List.of(user));
        96  +               } else {
        97  +                   String subject = resetToken + " is your recovery code of your
                    OpenEx account";
        98  +                   String body = "Hi " + username + ",</br>" +
        99  +                           "A request has been made to reset your OpenEx password.
                    </br>" +
        100 +                           "Enter the following password recovery code: " +
                    resetToken;
        101 +                   mailingService.sendEmail(subject, body, List.of(user));
        102 +               }
        103 +               // Store in memory reset token
        104 +               resetTokenMap.put(resetToken, user.getId());
```

```
        }
    }

    @PostMapping("/api/reset/{token}")
    public User changePasswordReset(@PathVariable String token, @Valid
@RequestBody ChangePasswordInput input) throws InputValidationException {
        String userId = resetTokenMap.get(token);
        if (userId != null) {
            String password = input.getPassword();
            String passwordValidation = input.getPasswordValidation();
            if (!passwordValidation.equals(password)) {
                throw new InputValidationException("password_validation", "Bad
password validation");
            }
            User changeUser = userRepository.findById(userId).orElseThrow();
            changeUser.setPassword(userService.encodeUserPassword(password));
            User savedUser = userRepository.save(changeUser);
            resetTokenMap.remove(token);
            return savedUser;
        }
        // Bad token or expired token
        throw new AccessDeniedException("Invalid credentials");
    }

    @GetMapping("/api/reset/{token}")
    public boolean validatePasswordResetToken(@PathVariable String token) {
        return resetTokenMap.get(token) != null;
    }

    @RolesAllowed(ROLE_ADMIN)
    @GetMapping("/api/users")
    public Iterable<User> users() {
@@ -79,7 +138,7 @@ public Iterable<User> users() {
    @RolesAllowed(ROLE_ADMIN)
    @PutMapping("/api/users/{userId}/password")
    public User changePassword(@PathVariable String userId,
-                               @Valid @RequestBody UpdatePasswordInput input) {
+                               @Valid @RequestBody ChangePasswordInput input) {
        User user = userRepository.findById(userId).orElseThrow();
        user.setPassword(userService.encodeUserPassword(input.getPassword()));
```

```
85  144             return userRepository.save(user);
```

...ex/rest/user/form/login/ResetUserInput.java

```
@@ -0,0 +1,29 @@
+ package io.openex.rest.user.form.login;
+
+ import javax.validation.constraints.NotBlank;
+
+ import static io.openex.config.AppConfig.MANDATORY_MESSAGE;
+
+ public class ResetUserInput {
+
+     @NotBlank(message = MANDATORY_MESSAGE)
+     private String login;
+
+     private String lang;
+
+     public String getLogin() {
+         return login;
+     }
+
+     public void setLogin(String login) {
+         this.login = login;
+     }
+
+     public String getLang() {
+         return lang;
+     }
+
+     public void setLang(String lang) {
+         this.lang = lang;
+     }
+ }
```

Comments 0

Please sign in to comment.