

Bookstore

2



Baza H2

H2 to otwartoźródłowa lekka i szybka baza danych napisana w javie- sprawdzi się jako baza tesowa.

H2 może być używana w trybie wbudowanym, gdzie działa jako biblioteka wewnątrz aplikacji Java, lub w trybie serwerowym, umożliwiając zdalny dostęp przez sieć.

H2 może być zapisywać dane w pamięci lub również umieszczać je w pliku.

<https://www.h2database.com/html/tutorial.html>

Ustawienia aplikacji:

Baza H2

```
spring.datasource.url=jdbc:h2:file:C:/Users/luke/test.db
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password
spring.h2.console.enabled=true
```

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Ustawienia reguł spring security:

http

```
.authorizeHttpRequests(auth -> auth
```

```
.requestMatchers("/login", "/register").permitAll()
.requestMatchers("/h2-console/**").permitAll()
.anyRequest().authenticated()
```

```
).csrf(csrf->csrf.ignoringRequestMatchers("/h2-console/**"))
.headers(headers->
```

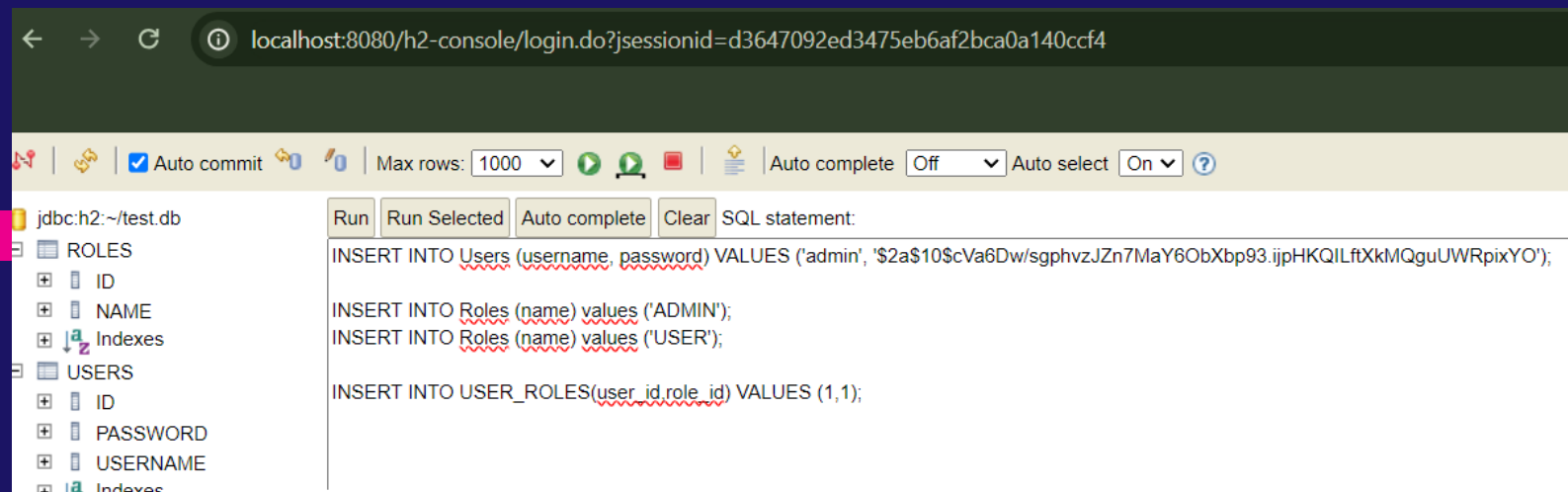
```
>headers.frameOptions(HeadersConfigurer.FrameOptionsConfig::sameOrigin))
```



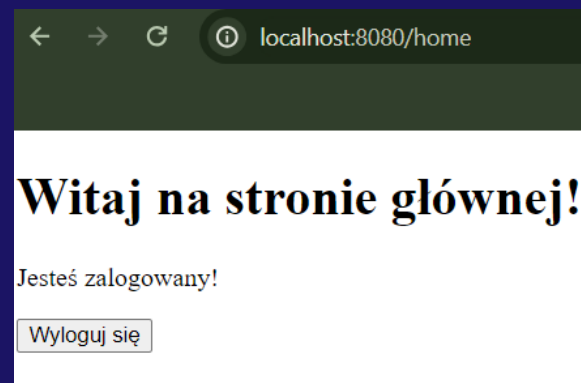
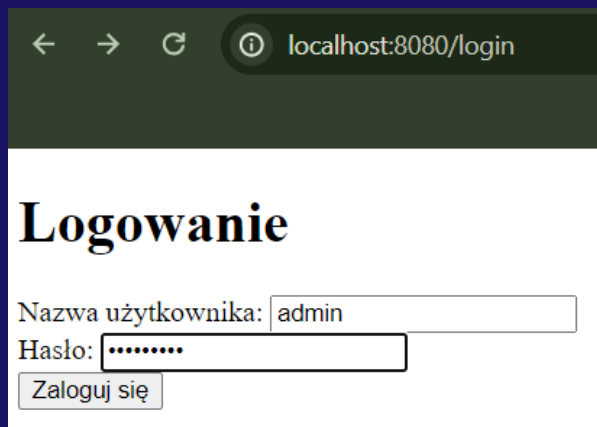
Dostęp przez przeglądarkę: localhost:8080/h2-console

A screenshot of the H2 Console web interface. The interface is titled 'Login' and shows the 'Generic H2 (Embedded)' driver class selected. The JDBC URL is 'jdbc:h2:~/test.db', the user name is 'sa', and the password is empty. There are 'Connect' and 'Test Connection' buttons at the bottom. The interface is in English and has a menu bar with 'Preferences', 'Tools', and 'Help'.

Baza H2



Hash generator (domyślnie w spring bcrypt
ma round = 10):
<https://bcrypt-generator.com/>



Strona rejestracji

`th:object="${user}"` - obiekt
przekazywany przez model

`th:field="*{password}"` - pole
obiektu przekazywany=ego przez
model

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Rejestracja</title>
</head>
<body>
<h1>Rejestracja</h1>
<form th:action="@{/register}" method="post" th:object="${user}">
  <div th:text="${message}"></div>
  <div>
    <label for="username">Nazwa użytkownika:</label>
    <input type="text" id="username" th:field="*{username}"
required />
  </div>
  <div>
    <label for="password">Hasło:</label>
    <input type="password" id="password" th:field="*{password}"
required />
  </div>
  <div>
    <button type="submit">Zarejestruj się</button>
  </div>
</form>
</body>
</html>
```

Kontroler

```
@Autowired
private UserService userService;
```

```
@GetMapping("/register")
public String showRegistrationForm(Model model) {
    model.addAttribute("user", new User());
    return "register";
}
```

```
@PostMapping("/register")
public String registerUser(@ModelAttribute("user") User user, Model model) {
    String result = userService.registerUser(user);
    model.addAttribute("message", result);
    if (result.equals("success")) {
        return "redirect:/login";
    }
    return "register";
}
```

UserService



```
public interface UserService {  
    Optional<User> findByUsername(String username);  
    String registerUser(User user);  
}
```

UserService

```
@Autowired
private UserRepository userRepository;

@Autowired
private PasswordEncoder passwordEncoder;

@Autowired
private RoleRepository roleRepository;
```

```
@Transactional
public Optional<User> findByUsername(String username) {
    return userRepository.findByUsername(username);
}
```

```
@Transactional
public String registerUser(User user) {
    if (userRepository.findByUsername(user.getUsername()).isPresent()) {
        return "failure";
    }
    user.setPassword(passwordEncoder.encode(user.getPassword()));

    Role userRole = roleRepository.findByName("USER").orElseGet(null);
    if (userRole != null) {
        user.getRoles().add(userRole);
    }
    else {
        Role role = new Role();
        role.setName("USER");
        user.getRoles().add(role);
    }

    userRepository.save(user);
    return "success";
}
```


Repozytoria



Przypomnienie gotowych repozytoriów:

```
public interface RoleRepository extends JpaRepository<Role, Long> {  
    Optional<Role> findByName(String name);  
}
```

```
public interface UserRepository extends JpaRepository<User, Long> {  
    Optional<User> findByUsername(String username);  
    User save(User user);  
}
```

Endpoint tylko dla admina

Reguła – wszystkie endpointy przeznaczone dla admina – user musi posiadać rolę „ADMIN”:

```
.requestMatchers("/admin/**").hasAuthority("ADMIN")
```

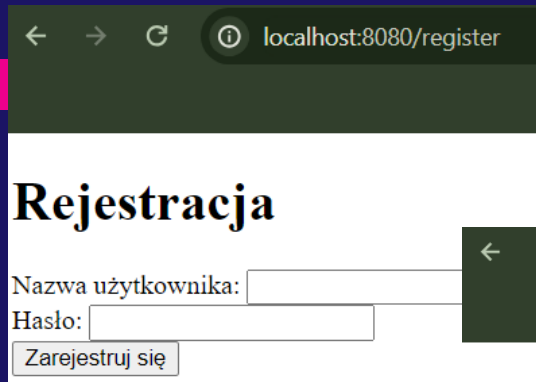
Kontroler z panelem admina:

```
@GetMapping("/admin/adminpanel")
public String adminpanel() {
    return "adminpanel";
}
```

Strona adminpanel.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Strona główna</title>
</head>
<body>
<h1>Admin panel!</h1>
<p>admin</p>
<form th:action="@{/logout}" method="post">
    <button type="submit">Wyloguj się</button>
</form>
</body>
</html>
```

Rejestracja i logowanie



localhost:8080/register

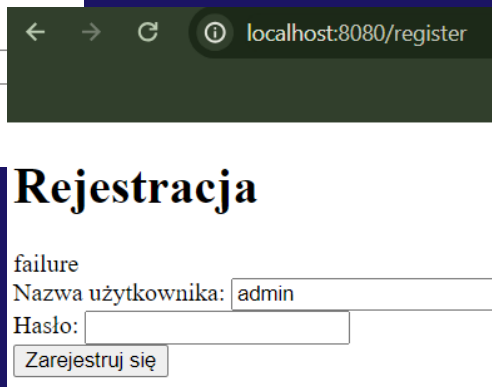
Rejestracja

Nazwa użytkownika:

Hasło:

Zarejestruj się

Strona rejestracji



localhost:8080/register

Rejestracja

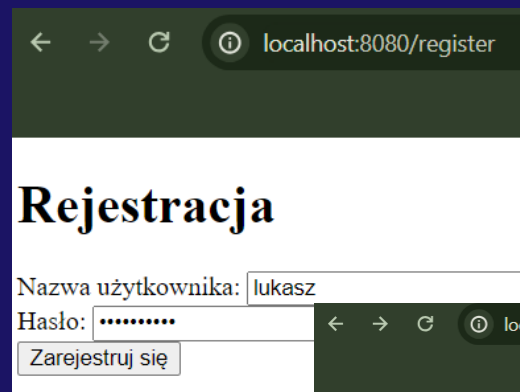
failure

Nazwa użytkownika:

Hasło:

Zarejestruj się

admin już istnieje



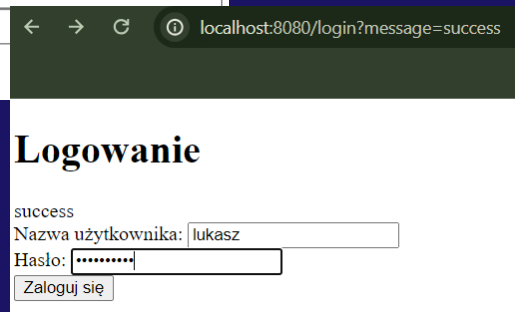
localhost:8080/register

Rejestracja

Nazwa użytkownika:

Hasło:

Zarejestruj się



localhost:8080/login?message=success

Logowanie

success

Nazwa użytkownika:

Hasło:

Zaloguj się

Utworzono usera, przekierowano na stronę logowania. Uwaga! Message o sukcesie w modelu po przekierowaniu zostaje utracony – aby wyświetlić „sukces” trzeba użyć innego sposobu!

Rejestracja i logowanie

Utworzono usera, przekierowano na stronę logowania. Uwaga! Message o sukcesie w modelu po przekierowaniu zostaje utracony – aby wyświetlić „sukces” trzeba użyć innego sposobu!

```
@PostMapping("/register")
public String registerUser(@ModelAttribute("user") User user, Model model, RedirectAttributes redirectAttributes) {
    String result = userService.registerUser(user);
    model.addAttribute("message", result);
    redirectAttributes.addAttribute("message", result);
    if (result.equals("success")) {
        return "redirect:/login";
    }
    return "register";
}
```

Dodatkowa wiadomość w kontrolerze logowania i na stronie:

```
@RequestParam(value = "message", required = false) String message,
Model model) {
if (message!=null){
    model.addAttribute("message", message);
}

<div th:if="${message}" th:text="${message}"></div>
```

Endpoint tylko dla admina

← → ↻ ⓘ localhost:8080/login?logout

Logowanie

Pomyślnie wylogowano!

Nazwa użytkownika:

Hasło:



← → ↻ ⓘ localhost:8080/admin/adminpanel

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Mon May 13 13:29:35 CEST 2024

There was an unexpected error (type=Forbidden, status=403).

← → ↻ ⓘ localhost:8080/login?logout

Logowanie

Pomyślnie wylogowano!

Nazwa użytkownika:

Hasło:



← → ↻ ⓘ localhost:8080/admin/adminpanel

Admin panel!

admin

Przekierowanie po logowaniu

W zależności od roli user zostanie przekierowany na inną stronę po logowaniu:

Podmieniamy:

```
.defaultSuccessUrl("/home", true)
```

na:

```
.successHandler(new CustomLoginSuccessHandler())
```

Przekierowanie po logowaniu

CustomLoginSuccessHandler:

```
public class CustomLoginSuccessHandler implements AuthenticationSuccessHandler {

    @Override
    public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse response, Authentication authentication)
    throws IOException, ServletException {
        Set<String> roles = AuthorityUtils.authorityListToSet(authentication.getAuthorities());
        if (roles.contains("ADMIN")) {
            response.sendRedirect("/admin/adminpanel");
        } else {
            response.sendRedirect("/home");
        }
    }
}
```