

Sprawozdanie z ćwiczenia "Rest"

Strona początkowa

Na serwerze po uruchomieniu na serwerze tomcat endpointem początkowym jest:

http://localhost:8080/Cw_1_Servlets_war/login

Ralizacja zadania

Większość klas użytych w programie jest zgodna z opisem w instrukcji, do Dashboard dodano kod odpowiedzialny za realizację żądań *POST* oraz *DELETE* tylko dla użytkownika, którego rola jest równa "Role.ADMIN", klasa przedstawia się w następujący sposób:

```
package servlets;

import beans.Book;
import beans.Role;
import beans.User;
import com.google.gson.Gson;
import requests.NewBook;
import responses.ExceptionResponse;
import responses.GetDashboardReponse;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

@WebServlet(name = "DashboardServlet", urlPatterns = {"/dashboard",
"/dashboard/*"})
public class DashboardServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        var user = (User) request.getSession().getAttribute("User");
        response.setContentType("application/json;charset=UTF-8");
        Gson gson = new Gson();
        try {
            if (user != null && user.getRole().equals(Role.ADMIN)) {
                System.out.println("In DashboardServlet POST");

                NewBook newBookRequest = gson.fromJson(request.getReader(),
NewBook.class);
                ArrayList<Book> Books = (ArrayList<Book>)
request.getSession().getAttribute("Books");
```

```
        var newBook = new Book(newBookRequest.Title,
newBookRequest.Author, newBookRequest.Year);
        Books.add(newBook);
        request.getServletContext().setAttribute("Books", Books);
        response.setStatus(201);
        gson.toJson(newBook, response.getWriter());
    }
    else {
        throw new Exception("Unauthorized user");
    }
} catch (Exception ex) {
    ExceptionResponse exResponse = new ExceptionResponse();
    exResponse.setMessage(ex.getLocalizedMessage());
    exResponse.setStatus(401);
    ((HttpServletResponse) response).setStatus(401);
    gson.toJson(exResponse, response.getWriter());
}

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    System.out.println("In DashboardServlet GET");
    response.setContentType("application/json;charset=UTF-8");
    Gson gson = new Gson();
    try {
        List<Book> books = getBooksFromContext(request.getServletContext());
        GetDashboardReponse res = new GetDashboardReponse(books, 200);
        gson.toJson(res, response.getWriter());
    } catch (Exception ex) {
        ExceptionResponse exResponse = new ExceptionResponse();
        exResponse.setMessage(ex.getLocalizedMessage());
        exResponse.setStatus(401);
        ((HttpServletResponse) response).setStatus(401);
        gson.toJson(exResponse, response.getWriter());
    }
    System.out.println("Out DashboardServlet GET");
}

private List<Book> getBooksFromContext(ServletContext context) {
    return (ArrayList<Book>) context.getAttribute("Books");
}

protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    resp.setContentType("application/json;charset=UTF-8");
    Gson gson = new Gson();
    var user = (User) req.getSession().getAttribute("User");
    try {
        if (user != null && user.getRole().equals(Role.ADMIN)) {
            var UrlParts = req.getRequestURL().toString().split("/");
            var Id = Integer.parseInt(UrlParts[UrlParts.length - 1]);
        }
    }
}
```

```
        var books = getBooksFromContext(req.getServletContext());
        var bookToRemove = books.stream().filter(book -> (book.getId() ==
Id)).findFirst();
        if (bookToRemove.isPresent()) {
            books.removeIf(book -> (book.getId() == Id));
            req.getServletContext().setAttribute("Books", books);
            gson.toJson(bookToRemove, resp.getWriter());
        } else {
            throw new Exception("No book with id = " + Id);
        }
    } else {
        throw new Exception("Unauthorized user");
    }
} catch (Exception ex) {
    ExceptionResponse exResponse = new ExceptionResponse();
    exResponse.setMessage(ex.getLocalizedMessage());
    exResponse.setStatus(401);
    ((HttpServletResponse) resp).setStatus(401);
    gson.toJson(exResponse, resp.getWriter());
}
}
```

Działanie

Niestety z braku czasu i umiejętności nie udało mi się stworzyć odpowiedniego frontendu jednak jako, że założeniem architektury *REST* jest współpraca z dowolnym frontendem który spełnia zasady tworzenia API zdecydowałem się testować wszystkie funkcjonalności za pomocą Postmana.

API przedstawia się następująco:

Logowanie

end point = "http://localhost:8080/Cw_1_Servlets_war/login" Metoda = *POST*

```
{
  "login": "<login>"
  "password": "<password>"
}
```

Wynik: Użytkownik zostanie zalogowany oraz przekierowany na dashboard. Odpowiedź: Ok w przypadku powodzenia.

Wyświetlanie dashboardu

end point = "http://localhost:8080/Cw_1_Servlets_war/dashboard*" Metoda = *GET* Wynik: Odpowiedź będzie zawierać w sobie listę książek. Odpowiedź: lista książek.

Dodawanie Książki

end point = "http://localhost:8080/Cw_1_Servlets_war/dashboard*" Metoda = *POST*

```
{
  "Title": "<Title>",
  "Author": "<Author>",
  "Year": "<Year>"
}
```

Wynik: Do zbioru książek zostanie dodana nowa zgodna z jsonem książka. Odpowiedź: Nowo dodana książka.

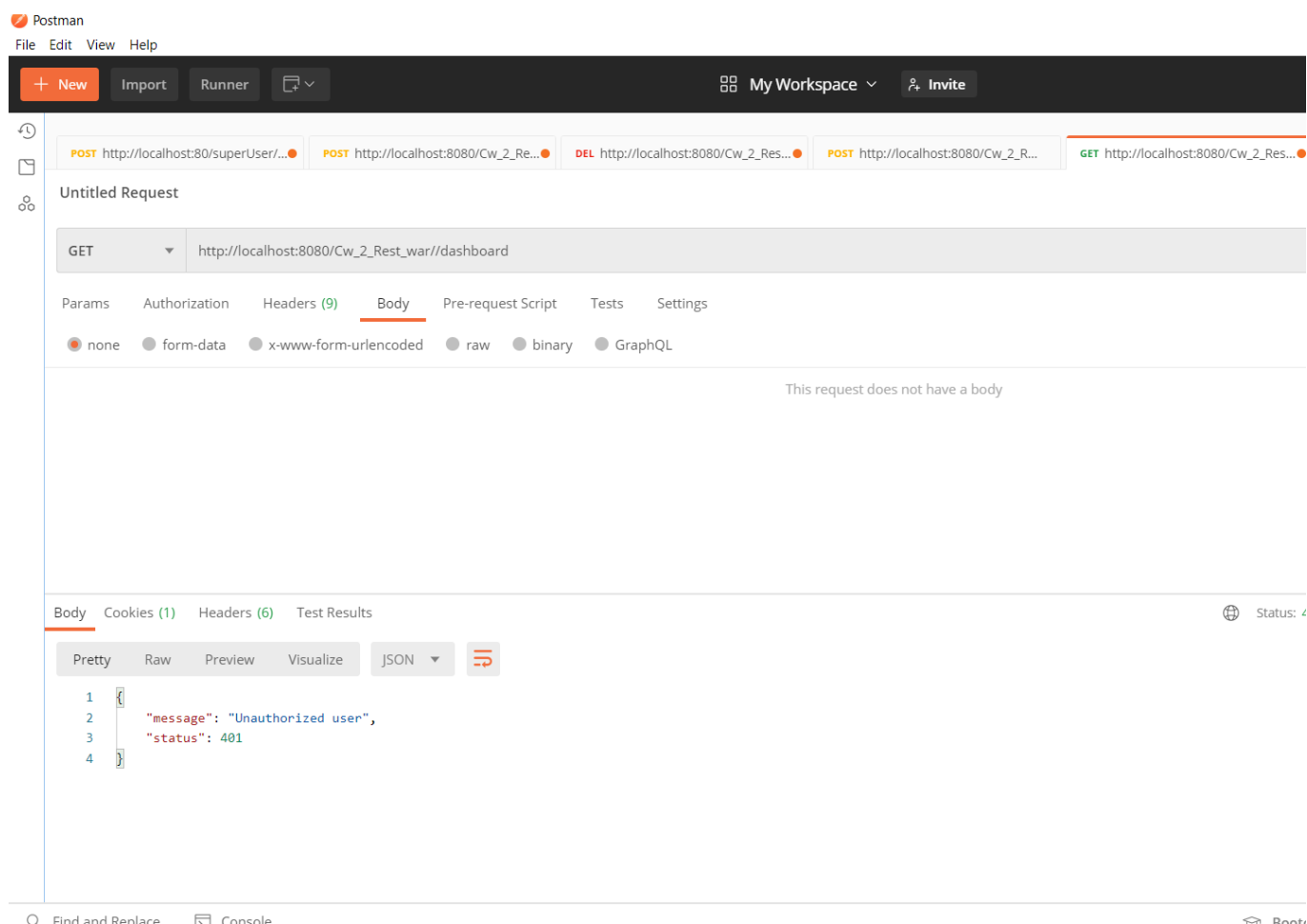
Dodawanie Książki

end point = "http://localhost:8080/Cw_1_Servlets_war/dashboard/{ID}" Metoda = *DELETE*

Wynik: Usunięta zostanie książka od Id == {ID}. Odpowiedź: Usunięta książka.

Obrazki Prezentująca działanie

Próba dostania się do dashboarda bez logowania



Logowanie użytkownika

POST http://localhost:80/superUser/...
POST http://localhost:8080/Cw_2_Re...
DEL http://localhost:8080/Cw_2_Res...
POST http://localhost:8080/Cw_2_R...

Untitled Request

POST
http://localhost:8080/Cw_2_Rest_war/login

Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON

```

1 {
2   "login": "user",
3   "password": "user"
4 }

```

Body
Cookies (2)
Headers (5)
Test Results

Pretty
Raw
Preview
Visualize
JSON

```

1 {
2   "books": [
3     {
4       "Id": 1,
5       "Title": "Title0",
6       "Author": "Author0",
7       "Year": "2000"
8     },
9     {
10      "Id": 2,

```

Próba usunięcia książki przez użytkownika

POST http://localhost:80/superUser/...

POST http://localhost:8080/Cw_2_Re...

DEL http://localhost:8080/Cw_2_Res...

POST http://localhost:8080/Cw_2_R...

Untitled Request

DELETE

http://localhost:8080/Cw_2_Rest_war//dashboard/1

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

Body

Cookies (2)

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"message": "Unauthorized user",

3

"status": 401

4

}

Logowanie admina

POST http://localhost:80/superUser/...
POST http://localhost:8080/Cw_2_R... X
DEL http://localhost:8080/Cw_2_Res...
POST http://localhost:8080/Cw_2_R...

Untitled Request

POST
http://localhost:8080/Cw_2_Rest_war/login

Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON

```

1 {
2   "login": "admin",
3   "password": "admin"
4 }

```

Body
Cookies (2)
Headers (5)
Test Results

Pretty
Raw
Preview
Visualize
JSON

```

1 {
2   "books": [
3     {
4       "Id": 1,
5       "Title": "Title0",
6       "Author": "Author0",
7       "Year": "2000"
8     },
9     {
10      "Id": 2,

```

Usuwanie książki przez admina

Untitled Request

DELETE


http://localhost:8080/Cw_2_Rest_war//dashboard/1

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies (2) Headers (5) Test Results

Pretty Raw Preview Visualize JSON 

```

1 {
2   "value": {
3     "Id": 1,
4     "Title": "Title0",
5     "Author": "Author0",
6     "Year": "2000"
7   }
8 }
```

Dashboard po usunięciu książki

Untitled Request

GET


http://localhost:8080/Cw_2_Rest_war//dashboard

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies (2) Headers (5) Test Results

Pretty Raw Preview Visualize JSON 

```

1 {
2   "books": [
3     {
4       "Id": 2,
5       "Title": "Title1",
6       "Author": "Author1",
7       "Year": "2001"
8     },
9     {
10      "Id": 3,
```

Dodanie książki przez admina

POST http://localhost:80/superUser/...
POST http://localhost:8080/Cw_2_R...
DEL http://localhost:8080/Cw_2_Res...
POST http://localhost:8080/Cw_2_R...
POST http://localhost:8080/Cw_2_R...

Untitled Request

POST

http://localhost:8080/Cw_2_Rest_war//dashboard

Params

Authorization

Headers (11)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```

1 {
2   "Title": "SuperKsiazka",
3   "Author": "Ktos",
4   "Year": "1996"
5 }

```

Body

Cookies (2)

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```

1 {
2   "Id": 11,
3   "Title": "SuperKsiazka",
4   "Author": "Ktos",
5   "Year": "1996"
6 }

```

Dashboard po dodaniu książki

Untitled Request

GET
http://localhost:8080/Cw_2_Rest_war//dashboard

Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON

```

1  {
2    "Title": "SuperKsiazka",
3    "Author": "Ktos",
4    "Year": "1996"
5  }

```

Body
Cookies (2)
Headers (5)
Test Results

Pretty
Raw
Preview
Visualize
JSON

```

56  },
57  {
58    "Id": 11,
59    "Title": "SuperKsiazka",
60    "Author": "Ktos",
61    "Year": "1996"
62  }
63  ],
64  "status": 200
65  }

```

Dane Logowania

Dane Logowania w programie to

- dla Admina:

```
login: admin
hasło: admin
```

- dla usera:

```
login: user
hasło: user
```

Dane do logowania dla usera są też domyślnymi w panelu logowania.

Imię Nazwisko: *Kacper Szczygiel*

Numer Albumu: 140453