Rozproszone systemy internetowe

JAX-RS (RESTful web services)

cz.2 Rozwój aplikacji CRUD

Sources and help:

What are RESTful Web Services (ORACLE)

Create a RESTful Web Service Using NetBeans IDE

JAX-RS Tutorial by mkyong

JAX-RS Tutorial by javaTpoint

Budowa aplikacji CRUD z użyciem RESTful web serwisu

HTTP Service Requests

RESTful web services are implemented using one or more of the following **four HTTP request types** depending on the design of the system. These services *loosely* map to the so-called *CRUD* operations: **Create, Retrieve, Update and Delete.**

- **POST** – Creates a resource on the server. The resource is contained in the body of the POST request. POST is analogous to an SQL *insert* statement.
- **GET** – Retrieves a resource from the server. The resource is specified with a URL and may include a ? to delineate the request from the request parameters. GET is analogous to an SQL *select* statement.
- **PUT** – Updates a resource on the server. The resource is contained in the body of the PUT request. PUT is analogous to an SQL *update* statement.
- **DELETE** – Deletes a resource on the server. The resource is specified in the URL only. DELETE is analogous to an SQL *delete* command.

```
@Consumes(MediaType.APPLICATION_JSON)

@Produces(MediaType.APPLICATION_JSON)
```

Czynność	**HTTP Metoda**	**URI**
Wyświetlenie wszystkich wiadomości	**GET**	**http://localhost:8080/RestWS4/webresources/messages**
Wyświetlenie jednej wiadomości	**GET**	**http://localhost:8080/RestWS4/webresources/messages/{messageId}**
Utworzenie nowej wiadomości	**POST**	**http://localhost:8080/RestWS4/webresources/messages**
Edycja wiadomości	**PUT**	**http://localhost:8080/RestWS4/webresources/messages/{messageId}**

Kasowanie wiadomości	**DELETE**	**http://localhost:8080/RestWS4/webresources/messages/{messageId}**

Ćwiczenie 1. Pobranie jednego rekordu

GET http://localhost:8080/RestWS4/webresources/messages/{messageId}

Przebudowa MessageService - dane reprezentowane są w formie Map<Long, Message>. Dane z DB symulowane w formie Map.

MessageResource.java | MessageService.java

Source | History

```
6   import java.util.Map;
7   import model.Message;
8
9   public class MessageService {
10
    static private Map<Long, Message> messages = new HashMap<Long, Message>();
12
13      public MessageService() {
14          messages.put(1L, new Message(1L, "Pierwsza wiadomość", "Jacek"));
15          messages.put(2L, new Message(2L, "Druga wiadomość", "Marek"));
16          messages.put(3L, new Message(3L, "Trzecia wiadomość", "Ewa"));
17      }
18
19      public List<Message> getAllMessages() {
            return new ArrayList<Message>(messages.values());
21      }
22
23      public Message getMessage(Long id) {
24          return messages.get(id);
25      }
26
27      public Message createMessage(Message message) {
28          message.setId(messages.size() + 1L);
29          messages.put(messages.size() + 1L, message);
30
```

```
@GET

@Path("/{messageId}")

public Message getMessage(@PathParam("messageId") Long id) {

return messageService.getMessage(id);

}
```

GET http://localhost:8080/RestWS4/webresources/messages/2

GET http://localhost:8080/RestWS4/webresources/messages/2

Authorization | Headers (0) | Body | Pre-request script | Tes

Header | Value

Body | Cookies | Headers (5) | Tests (0/0) | Status 200 OK | Time 323 ms

Pretty | Raw | Preview | JSON


```
1 {
2   "author": "Marek",
3   "created": "2016-02-19T11:53:24.345+01:00",
4   "id": 2,
5   "message": "Druga wiadomość"
6 }
```


Ćwiczenie 2. Tworzenie nowego rekordu (@POST)

POST http://localhost:8080/RestWS4/webresources/messages

```
@POST

@Consumes(MediaType.APPLICATION_JSON)

public Message createMessage( Message message) {

//return "post test";

return messageService.createMessage(message);

}
```

Testowanie REST API:

Stan przed utworzeniem nowego rekordu:

Builder Runner Import

http://localhost:808...

GET http://localhost:8080/RestWS4/webresources/messages

URL Parameter Key Value

Authorization Headers (0) Body Pre-request script Tests

No Auth

Body Cookies Headers (5) Tests (0/0) Status 200 OK Time 385 ms

Pretty Raw Preview XML


```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messages>
    <message>
        <author>Jacek</author>
        <created>2016-02-19T12:27:03.182+01:00</created>
        <id>1</id>
        <message>Pierwsza wiadomość</message>
    </message>
    <message>
        <author>Marek</author>
        <created>2016-02-19T12:27:03.182+01:00</created>
        <id>2</id>
        <message>Druga wiadomość</message>
    </message>
    <message>
        <author>Ewa
        <created>2016-02-19T12:27:03.182+01:00</created>
        <id>3</id>
        <message>Trzecia wiadomość</message>
    </message>
</messages>
```


Tworzenie nowego rekordu z użyciem Postman Rest Client:

http://localhost:808...

POST http://localhost:8080/RestWS4/webresources/messages/ Params

Ac Value

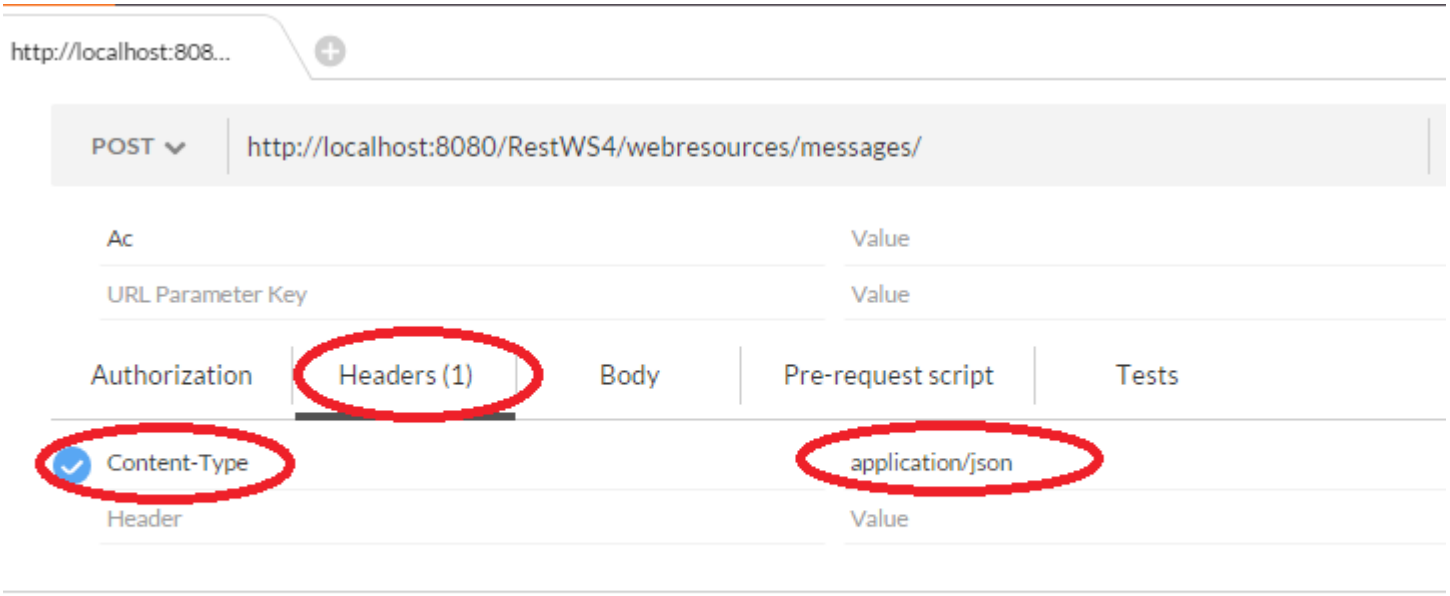
URL Parameter Key Value

Authorization Headers (1) Body Pre-request script Tests

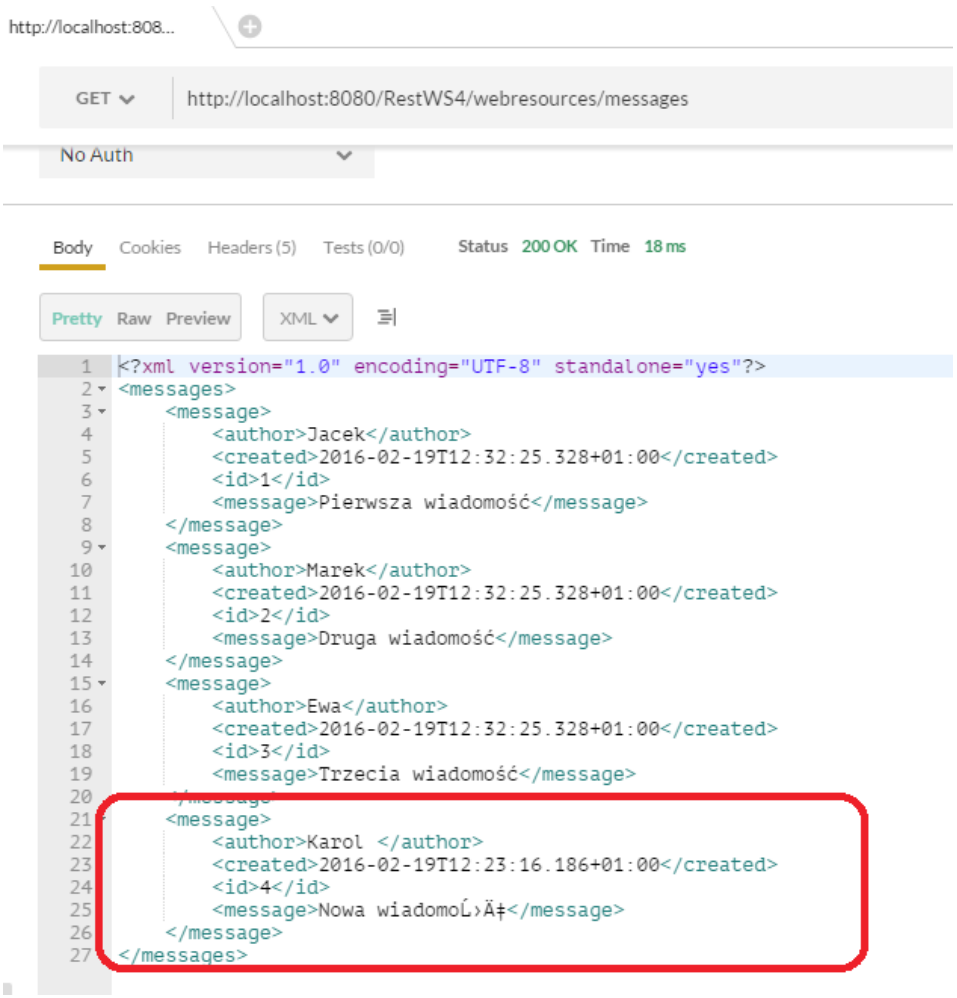
form-data x-www-form-urlencoded raw binary JSON (application/json)


```
{
  "author": "Karol ",
  "created": "2016-02-19T12:23:16.186+01:00",

  "message": "Nowa wiadomość"
},
```

Headers



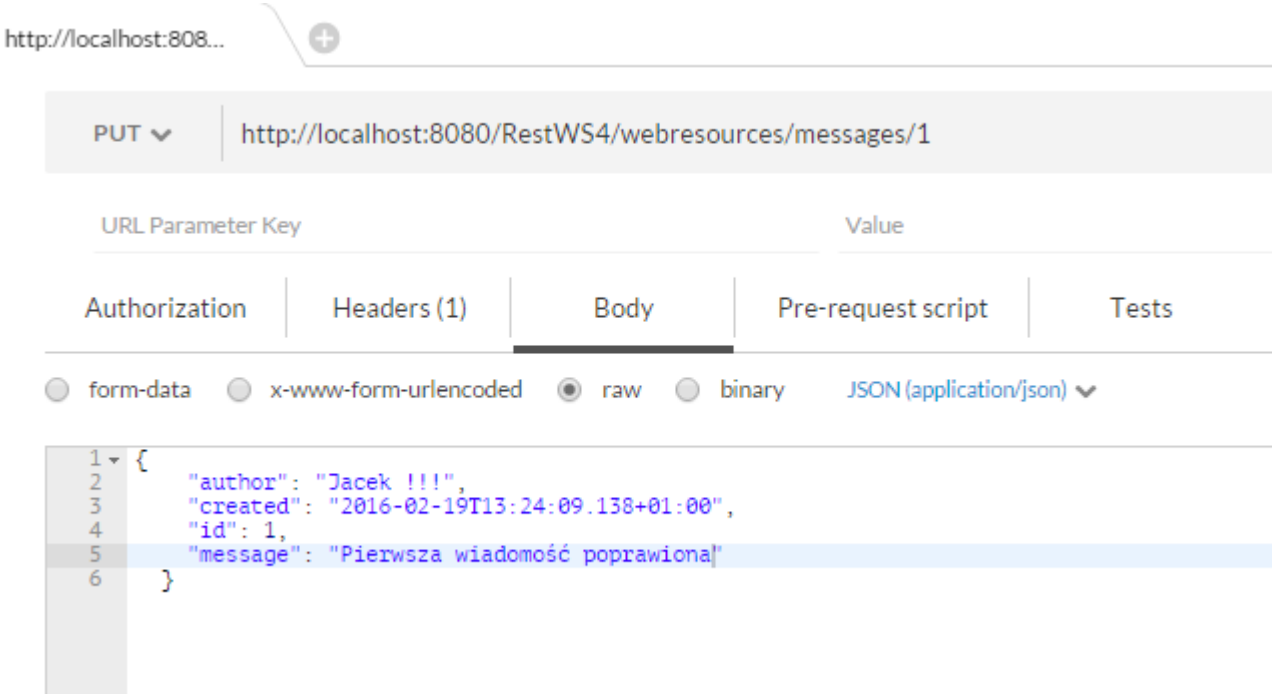
Stan po utworzeniu rekordu:



Ćwiczenie 3. Update record (@PUT)

Dodaj metodę updateMessage

```
@PUT

@Path("/{messageId}")

@Consumes(MediaType.APPLICATION_JSON)

@Produces(MediaType.APPLICATION_JSON)

public Message updateMessage(Message message) {

//return "post test";

return messageService.updateMessage(message);

}
```



GET http://localhost:8080/RestWS4/webresources/messages

URL Parameter Key | Value

Authorization | Headers (1) | Body | Pre-request script | T

No Auth

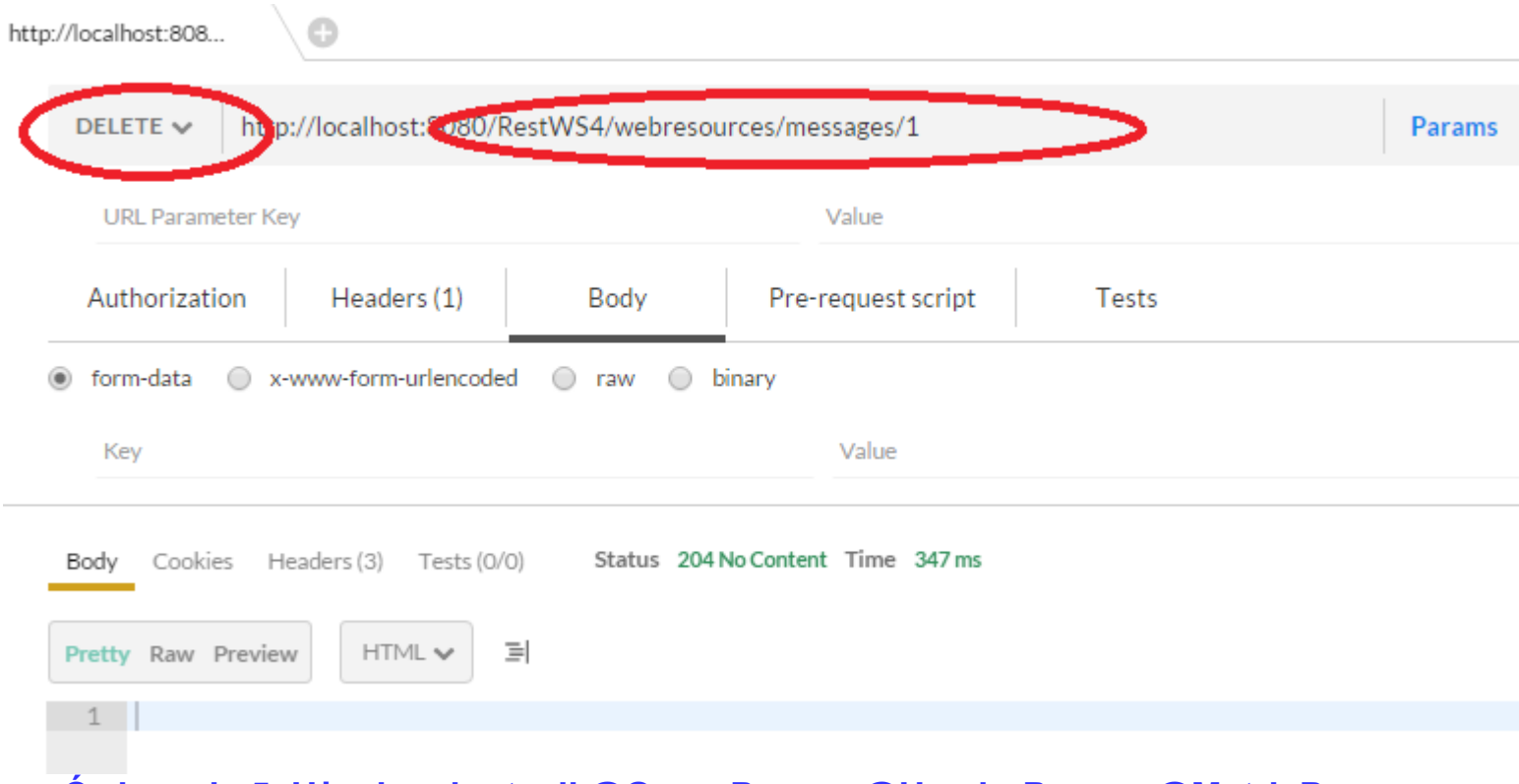
Body | Cookies | Headers (5) | Tests (0/0) | Status 200 OK | Time 23 ms

Pretty | Raw | Preview | JSON


```
[
  {
    "author": "Jacek !!!",
    "created": "2016-02-19T13:24:09.138+01:00",
    "id": 1,
    "message": "Pierwsza wiadomoĹ›Ä‡ poprawiona"
  },
  {
    "author": "Marek",
    "created": "2016-02-19T13:24:09.138+01:00",
    "id": 2,
    "message": "Druga wiadomość"
  },
  {
    "author": "Ewa",
    "created": "2016-02-19T13:24:09.138+01:00",
    "id": 3,
    "message": "Trzecia wiadomość"
  }
]
```


Ćwiczenie 4. Kasowanie rekordu (@DELETE)

Dodaj metodę kasowania rekordu



Ćwiczenie 5. Użycie adnotacji @QueryParam, @HeaderParam, @MatrixParam

A. Dodaj do metod serwisu adnotacje i przetestuj ich działanie

- @QueryParam,
- @HeaderParam,
- @MatrixParam

B. Przetestuj adnotację **@Context**

@Context UriInfo uriInfo

@Context HttpHeaders headers

Utwórz przykładową metodę korzystającą z UriInfo zwaracającą np.
uriInfo.getAbsolutePath().toString();