Rozproszone systemy internetowe

JAX-RS (RESTful web services)

cz.1

Sources and help:

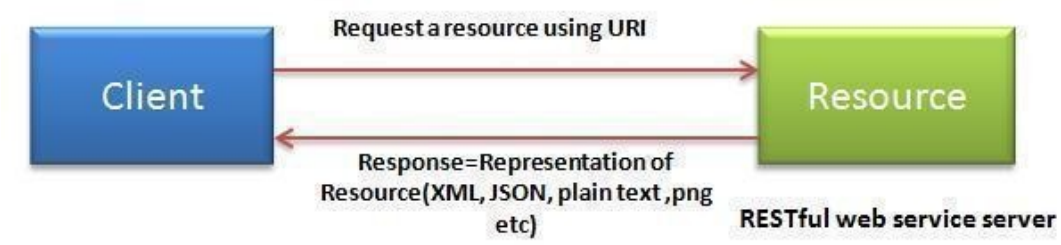
What are RESTful Web Services (ORACLE)

Create a RESTful Web Service Using NetBeans IDE

JAX-RS Tutorial by mkyong

JAX-RS Tutorial by javaTpoint

Representational State Transfer (REST) is an architectural style. In the REST architectural style, data and functionality are considered resources and are accessed using *Uniform Resource Identifiers (URIs)*, typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically **HTTP**. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.



Ćwiczenie 1. Tworzenie prostego RESTful web serwisu

Simple REST Style Web Service using NetBeans and GlassFish Server Server.

Usually NetBeans comes along with the GlassFish Server and also in build support for generating REST Services using the Jersey Framework.

File-->New -> New Project -> Java Web -> Web Application

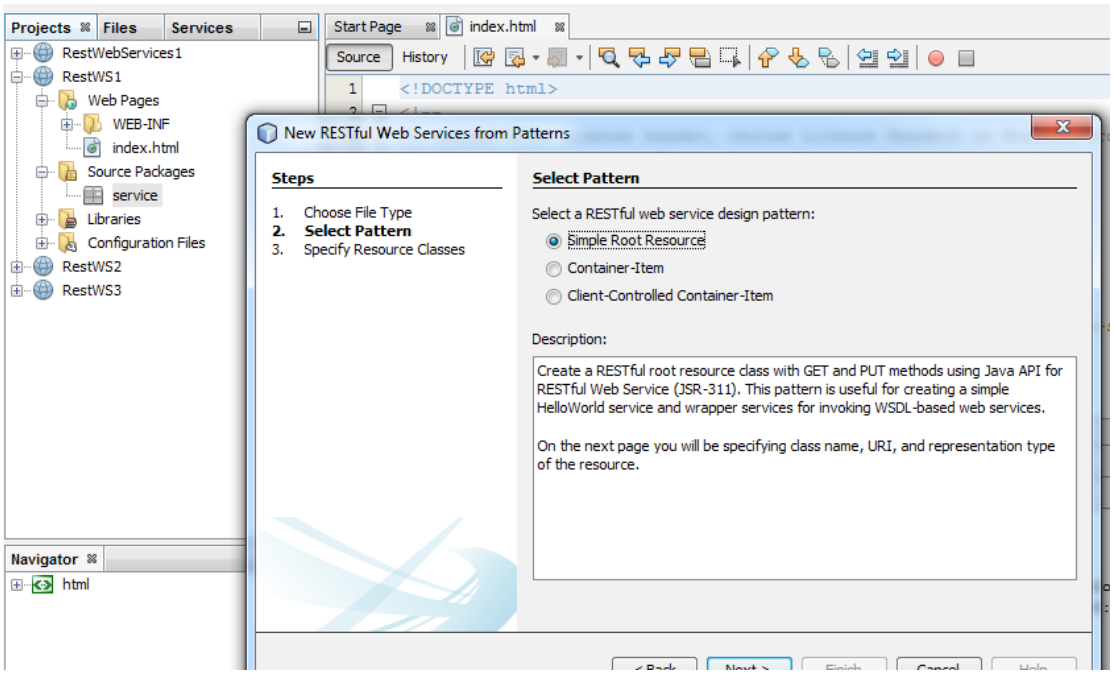
Basic and commonly used annotations:

@Path - Mention the Path from which you want to access a REST Service either class level or method level.

@GET - Performs HTTP get operation useful for getting info read only.

@Produces - This Produces the Respective output in different format such as XML, JSON, TEXT, HTML etc to the client.

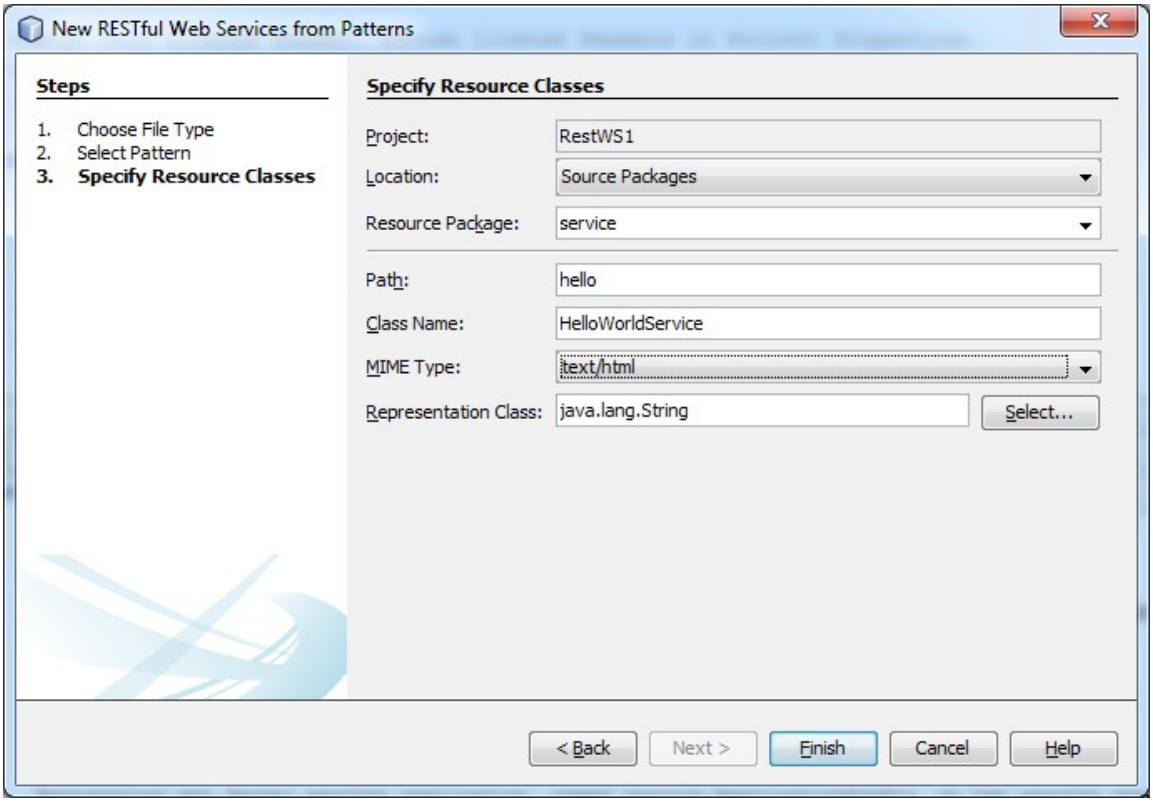
New->Other->WebService->RESTful WebServices from Patterns.



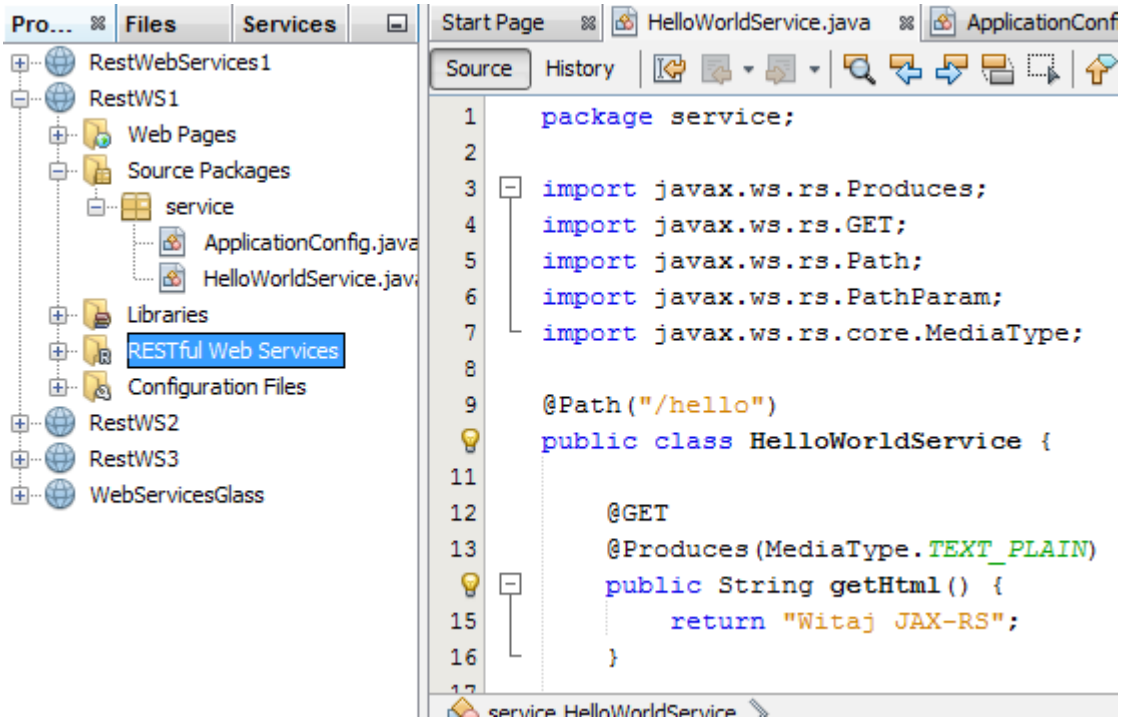
HelloWorldService

/hello

Give Path name as "hello" and class Name as"HelloWorldService" and select the MIME Type as "text/plain".

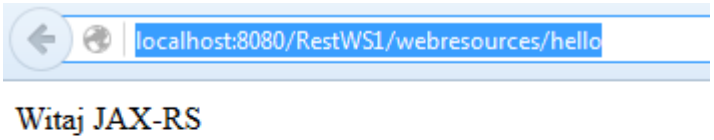


Można uprościć wygenerowany kod do formy przedstawionej na rysunku:



Deploy project

Testowanie: http://localhost:8080/RestWS1/webresources/hello



Witaj JAX-RS

Wywołanie serwisu (REST API) przez przeglądarkę zwraca tekst (rezultat wykonania metody getHtml() z klasy HelloWorldService)

PS. Zamiast @Produces("**text/plain**") mozna użyć @Produces(**MediaType.TEXT_PLAIN**), gdzie MediaType.TEXT_PLAIN jest stałą tekstową "text/**plain**"

MediaType.**APPLICATION_JSON** = "application/json"

MediaType.TEXT_XML = "text/xml"

Ćwiczenie 2. Ustawianie @Path dla metody

Dodaj dodatkową metodę oznaczoną adnotacją @Path("…") i @GET tak aby zwracała tekst "Witaj Echo"

http://localhost:8080/RestWS1/webresources/hello/echo

Witaj Echo

Ćwiczenie 3. Pobieranie parametru - @PathParam

Help

Zaimplementować nową REST API (metodę w serwisie) pobierającą parametr z URI

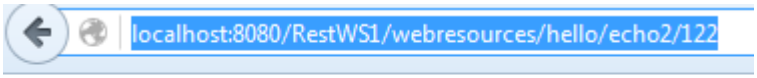
http://localhost:8080/RestWS1/webresources/hello/echo2/122

```
@GET

@Path("/echo2/{parametr}")

public String echo(@PathParam("parametr") String name) {

...
```



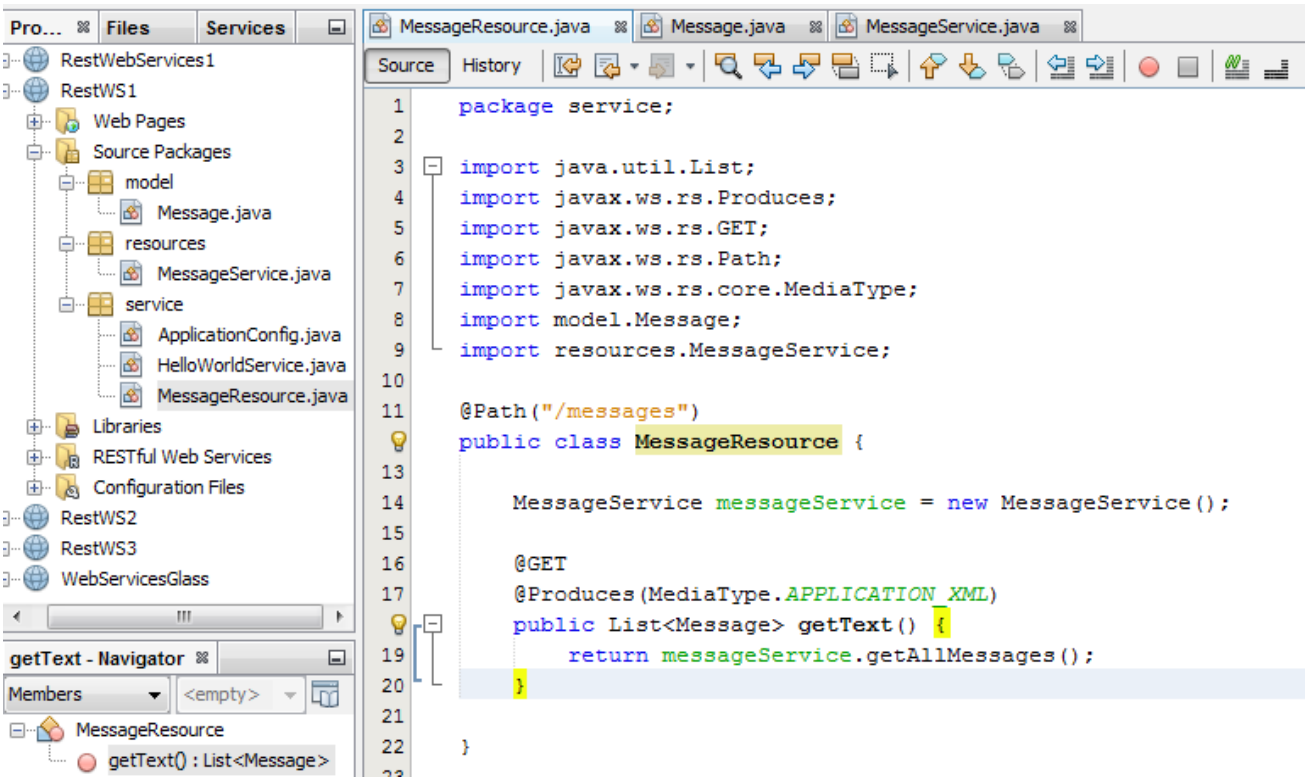
Witaj Echo: 122

Ćwiczenie 4. Zwrot lity obiektów w XML

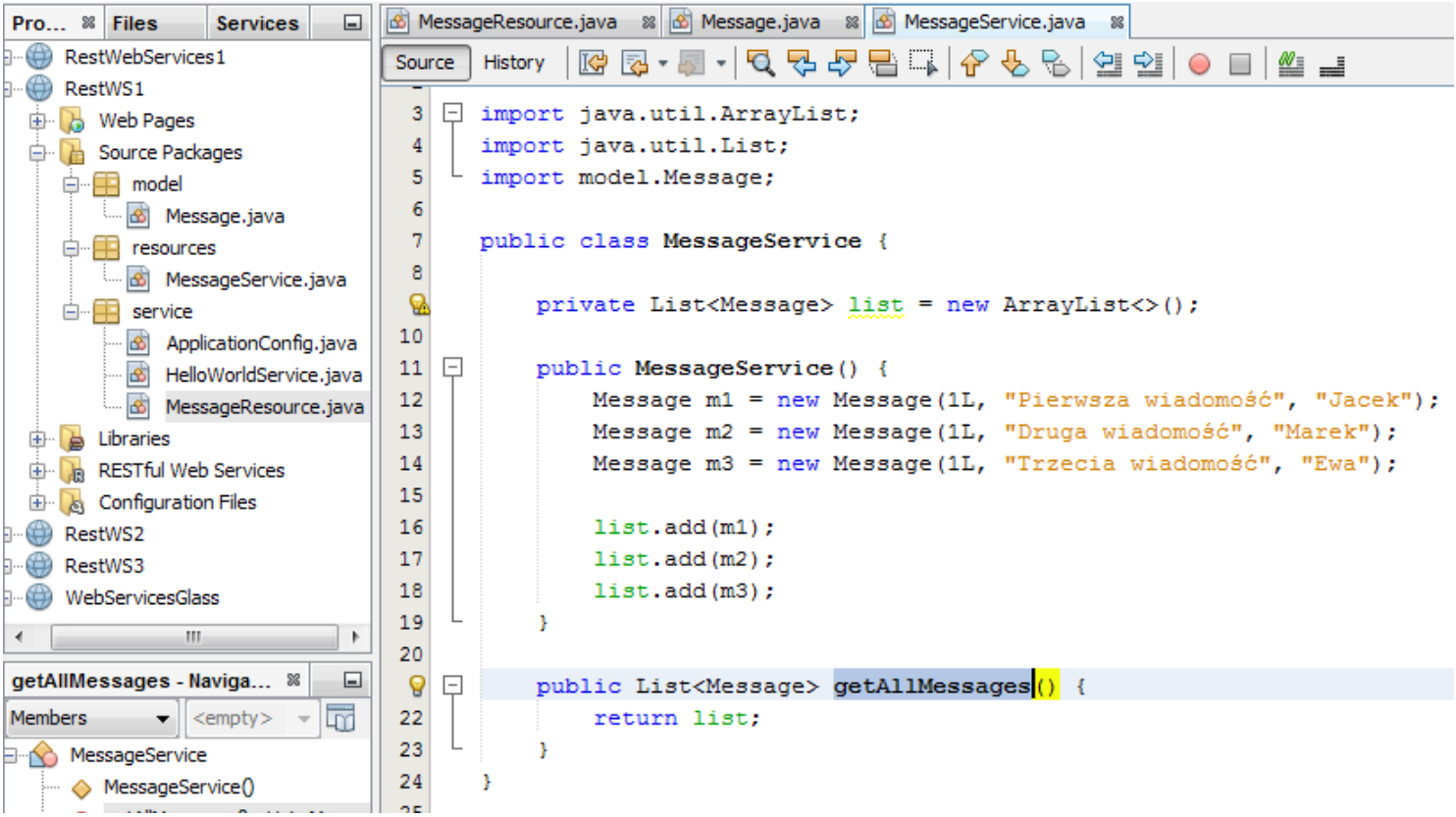
Zaimplementować REST API (metodę w serwisie) zwracającą **listę** obiektów w **formacie XML.**

http://localhost:8080/RestWS1/webresources/messages

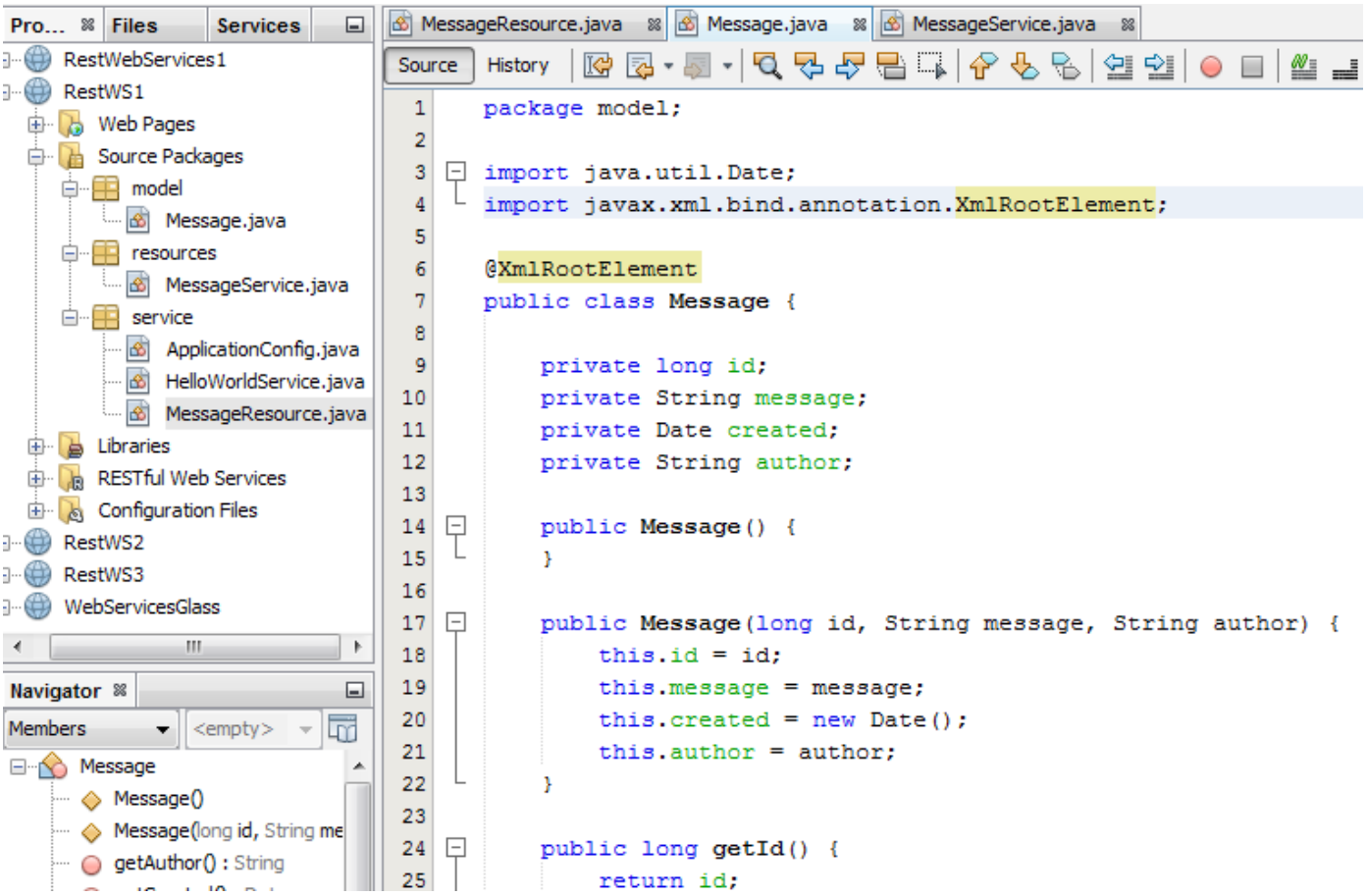
1. Zaimplementować nową klasę serwisu MessageResource.java. Będzie ona miała jedną metodę zwracającą listę rekordów w formacie XML.



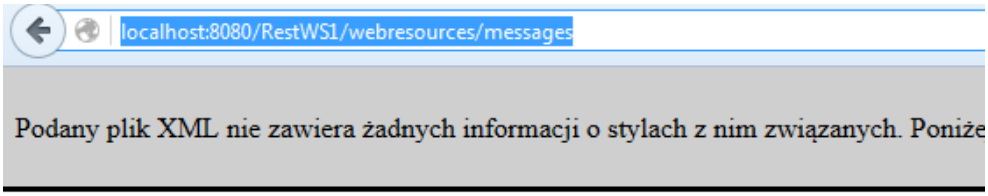
2. Utworzyć klasę modelu Message.java, z 2 konstruktorami, seterami, geterami, adnotacją **@XmlRootElement** (potrzebną dla **JAXB** do konwersji obiektu na XML)



3. Utworzyć klasę MessageService.java do przechowywania w pamięci listy obiektów Message, z metodą zwracającą listę (getAllMessages)



Testowanie: http://localhost:8080/RestWS1/webresources/messages

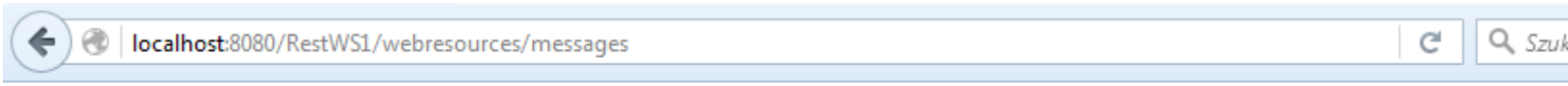



```
- <messages>
  - <message>
      <author>Jacek</author>
      <created>2016-02-19T10:42:51.048+01:00</created>
      <id>1</id>
      <message>Pierwsza wiadomość</message>
    </message>
  - <message>
      <author>Marek</author>
      <created>2016-02-19T10:42:51.048+01:00</created>
      <id>1</id>
      <message>Druga wiadomość</message>
    </message>
  - <message>
      <author>Ewa</author>
      <created>2016-02-19T10:42:51.048+01:00</created>
      <id>1</id>
      <message>Trzecia wiadomość</message>
    </message>
</messages>
```

Ćwiczenie 5. Zwracanie obiektów w formacie JSON

Zaimplementować REST API (metodę w serwisie) zwracającą listę obiektów w formacie **JSON**

http://localhost:8080/RestWS1/webresources/messages



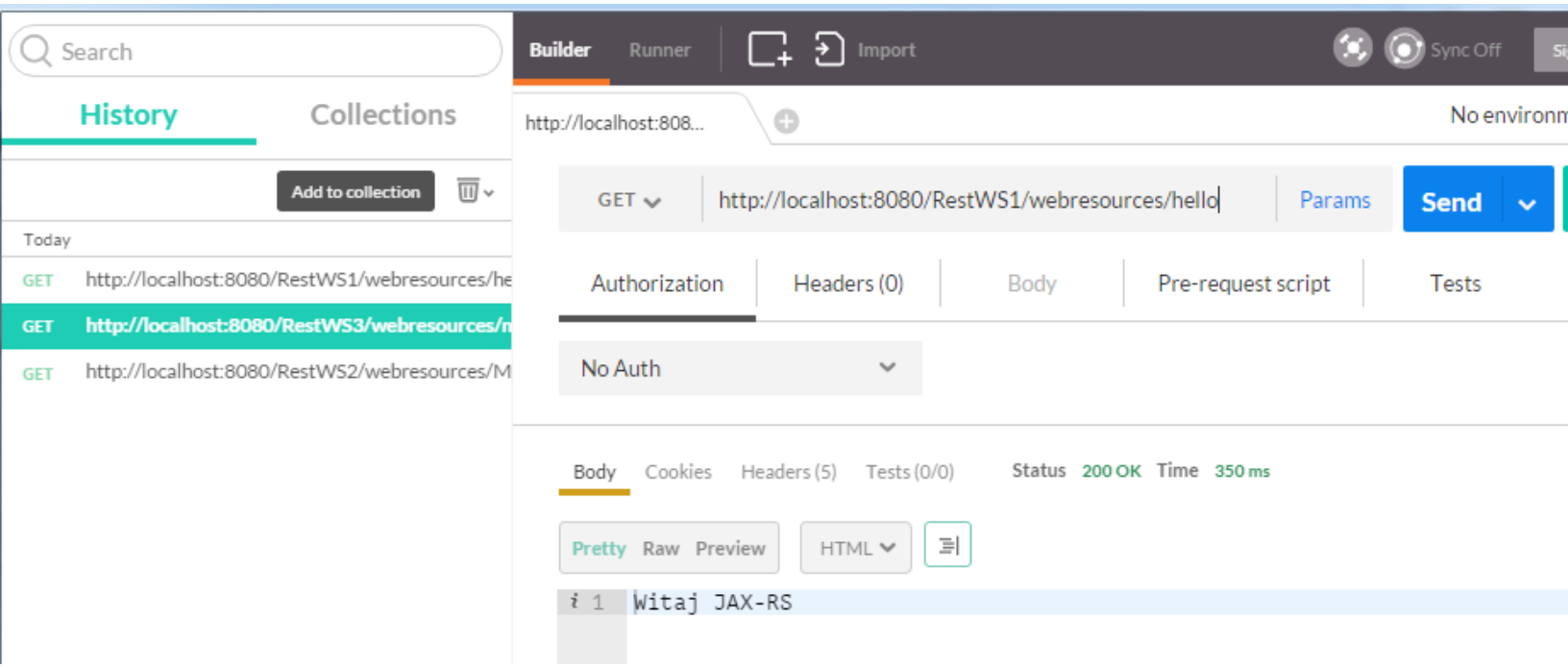

```
[{"author":"Jacek","created":"2016-02-19T10:45:32.085+01:00","id":1,"message":"Pierwsza wiadomość"},
{"author":"Marek","created":"2016-02-19T10:45:32.085+01:00","id":1,"message":"Druga wiadomość"},
{"author":"Ewa","created":"2016-02-19T10:45:32.085+01:00","id":1,"message":"Trzecia wiadomość"}]
```

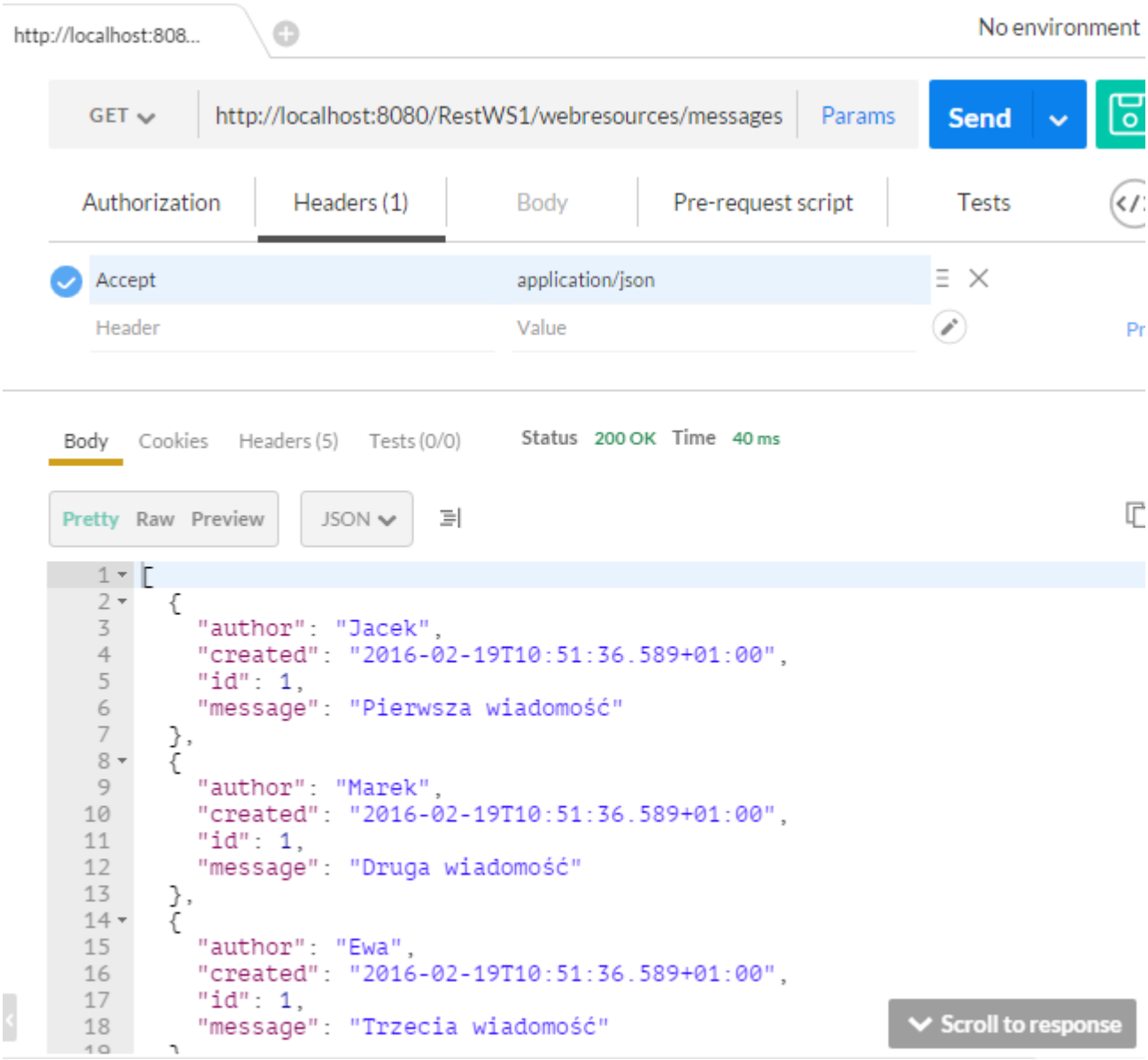

W przypadku problemu (Glasfish 4.1.1 i JSON) skorzystaj z Glassfish 4.1 lub użyj rozwiązania -> Rozwiązanie

Ćwiczenie 6. Testowanie REST API serwisu. Korzystanie z klienta REST Client – Postman

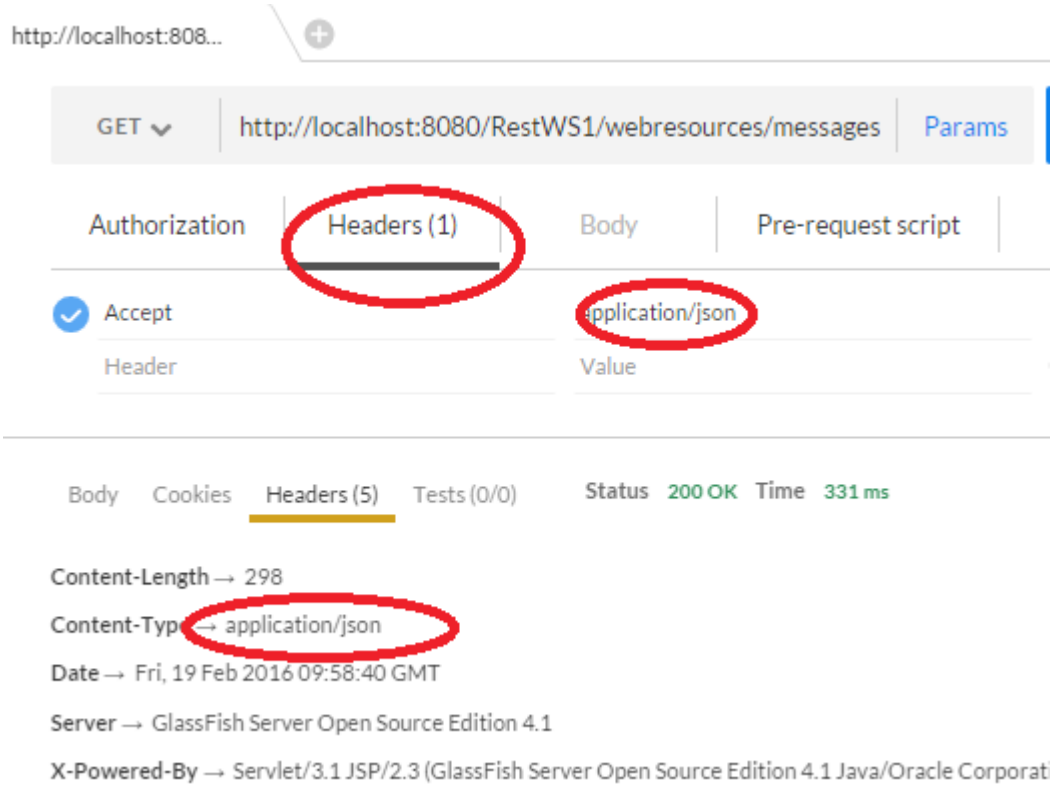
Przetestuj wszystkie stworzone REST API za pomocą **Postman REST Clienta.**

Postman REST Client (plugin przeglądarki Chrome). Można skorzystać z innego programu typu REST Client.





Uwaga: Do odbioru JSON ustawiono **Header** Accept : application/json


http://localhost:808...
GET
http://localhost:8080/RestWS1/webresources/messages
Params
Authorization
Headers (1)
Body
Pre-request script
Accept
application/json
Header
Value
Body
Cookies
Headers (5)
Tests (0/0)
Status 200 OK Time 331 ms
Content-Length → 298
Content-Type → application/json
Date → Fri, 19 Feb 2016 09:58:40 GMT
Server → GlassFish Server Open Source Edition 4.1
X-Powered-By → Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.1 Java/Oracle Corporat