Rest API with Jax-RS:

Web services types:

REST. Representational State Transfer. Mainly depends on Architectural design.

Soap. Simple object access protocol

Characteristics:

* Http Exchange: Client send Request to webserver and get the response in the form of xml/json/text. Can pick any HTTP method to call services. No rules to use any methods.
* Protocol: client send request to server with standard format. That message format can be   
  xml, json, text etc.
* Service definition: No document details for REST. WADL is equivalent for WSDL.  
  (for SOAP: WSDL, it has all the library details).

**REST and HTTP is works together to build webservices:**

Few Details on HTTP:

* HTTP: Hyper Text Transfer Protocol to exchange the data between client and server.
* Resource Allocation: like http url web services has url type like **resource based URI**  
  weather.com/zipcode/12345678
* After defining the URI next is to interact with server using HTTP Methods like **Get, Post, PUT and DELETE**. PUT is similar to Post but with bit different in implementation.
* Metadata: contains details such as headers, response data with other additional data.
* HTTP status code: 200-success, 500- server error, 404- not found.
* **Format Type :** when the client sent request to server, server need to understand type of request whether it is xml/json/text. Format type is mentioned in **Content-Type** which is also   
  part of meta data.  
  eg: text/xml, application/json.

Every REST webservice should have below details:

* Resource based URI
* HTTP Method
* HTTP status code
* Message header (to define format).

**Best Practices to Write Resource Based URI:**

* Identify the Resources, for example if there is access to social media app such as facebook  
  to get the profile information/message id below resource would be the good approach  
  **profiles/{profileName} : profiles/manjunath**

**Messages/{messageId} : messages/20 :** here resource is messages, should be plurals (means in future it might have multiple messages)

* Resources Relation: is like establishing the relationship using URI  
  eg: for the messages there can be multiple comments.  
  messages/1/comments/2 : which indicates that message one with comment 2.

**RESTful URI’s belongs to 2 types:**

* **Instant URI’s**
* **Collections URI’s**

****

**Above is Instance URI. Uses instances to identify the data, example message ID, likeId in above pic.**

**Collections URI: This has collections of resources.**

**To get all the messages:**

**/messages**

**All comments for message id 2**

**/messages/2/comments**

**Collection URI’s with Filter options:**

Suppose collection uri results huge data then we can give pagination option like below:

/messages?offset=30&limit=10

Offset: starting from 30 upto 10 records.

If we want records in particular year:

/messages? Year=2013&offset=20&limit=10

**HTTP Methods:**

**Used to perform operations like insert, delete or update.**

Eg: /getMessages.do?id=10 which is struts action to get the message id 10.

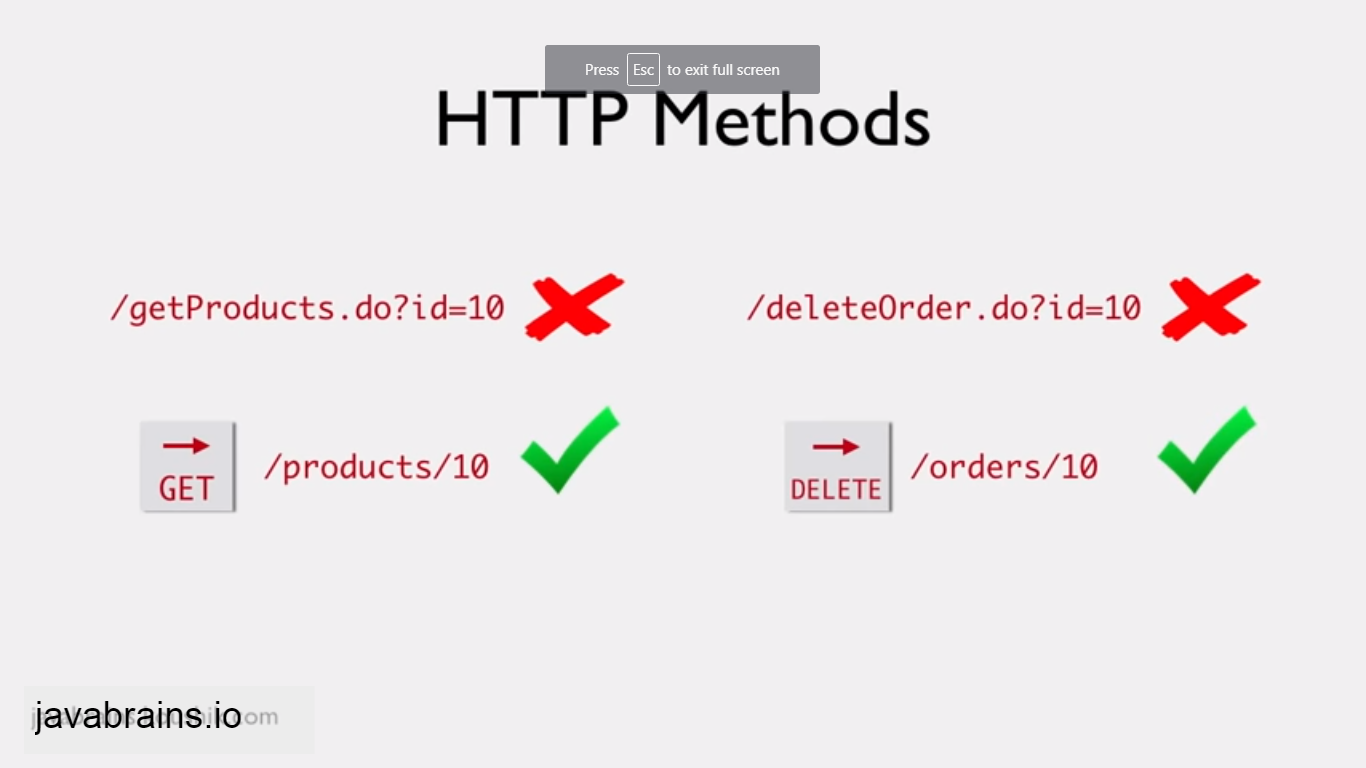
In Rest: /messages/10

/deleteMessages.do?id=10 in Rest: /messages/10

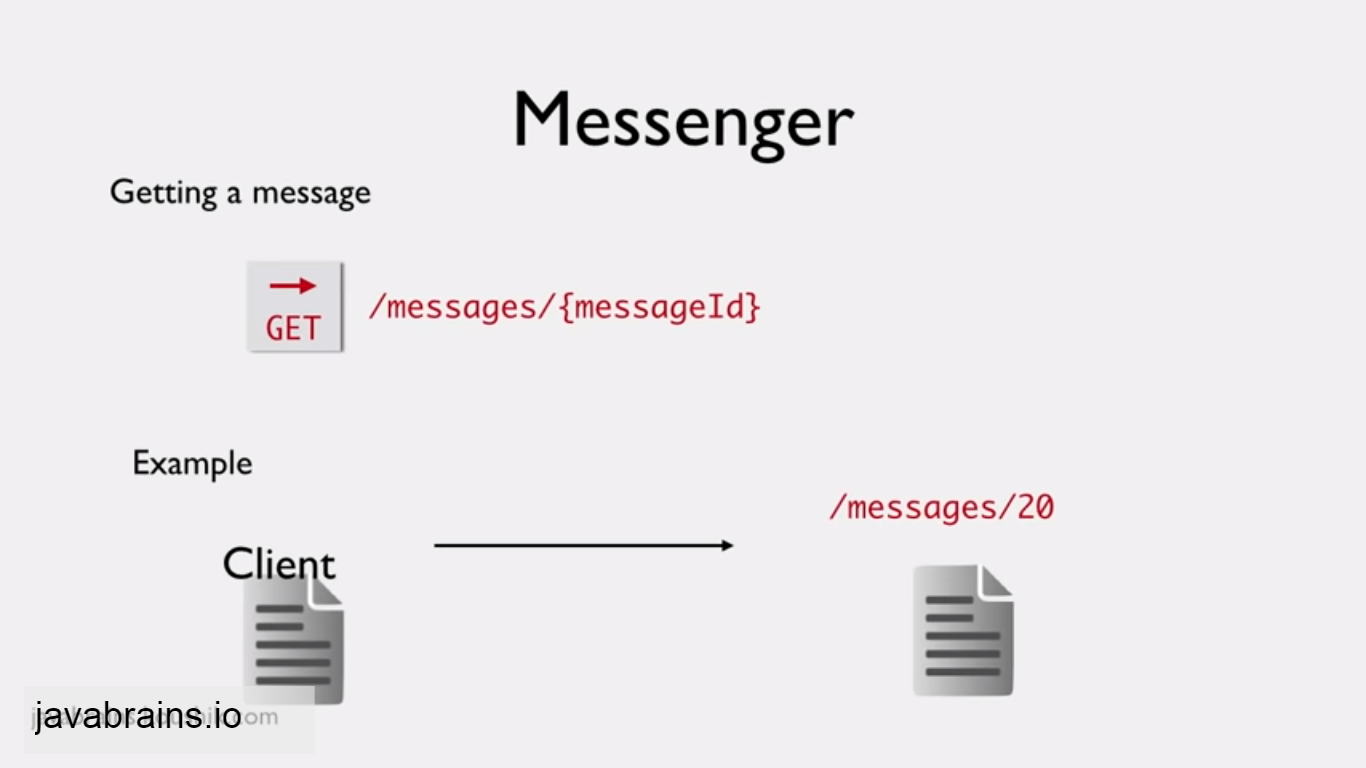
**Common HTTP methods:**

**GET, PUT , POST, DELETE : Commonly used methods**

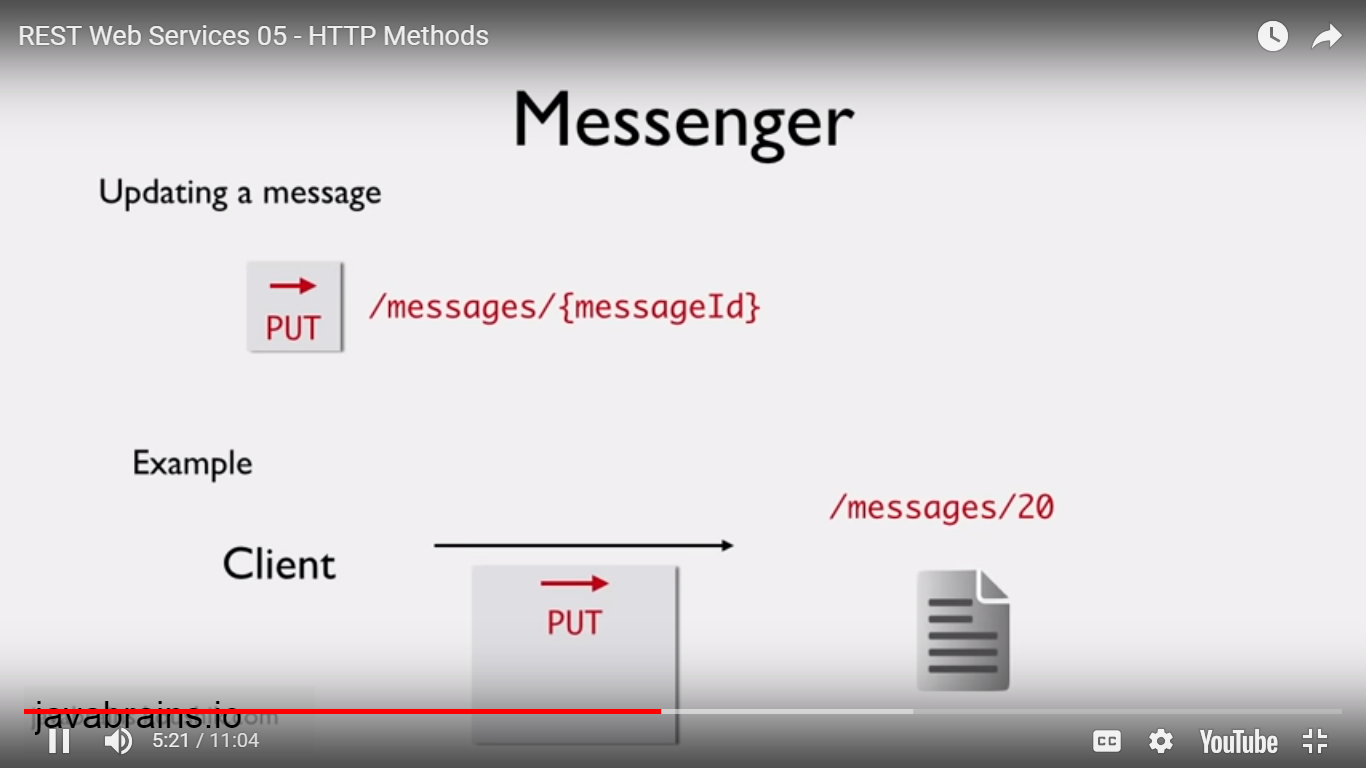
**HEAD, OPTIONS :** can also be used (not regularly)



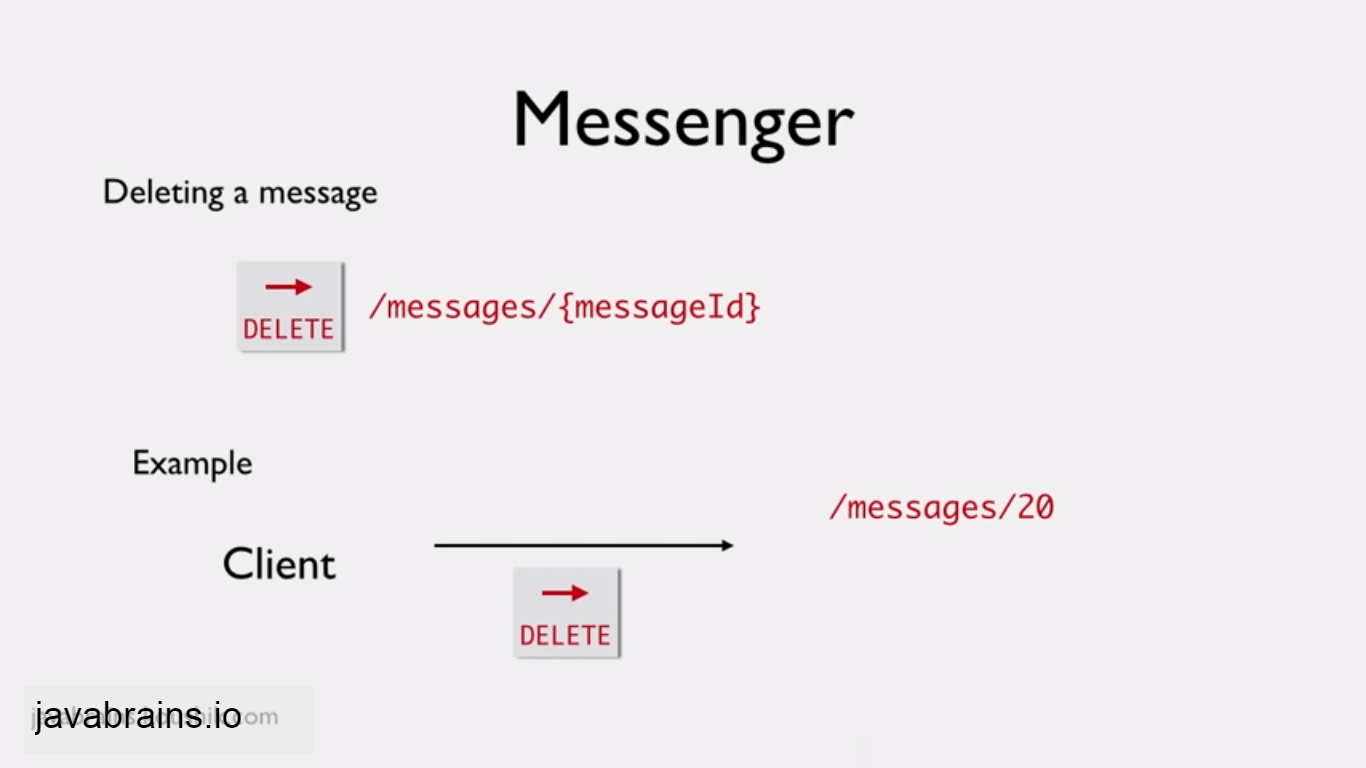
To get the message 20



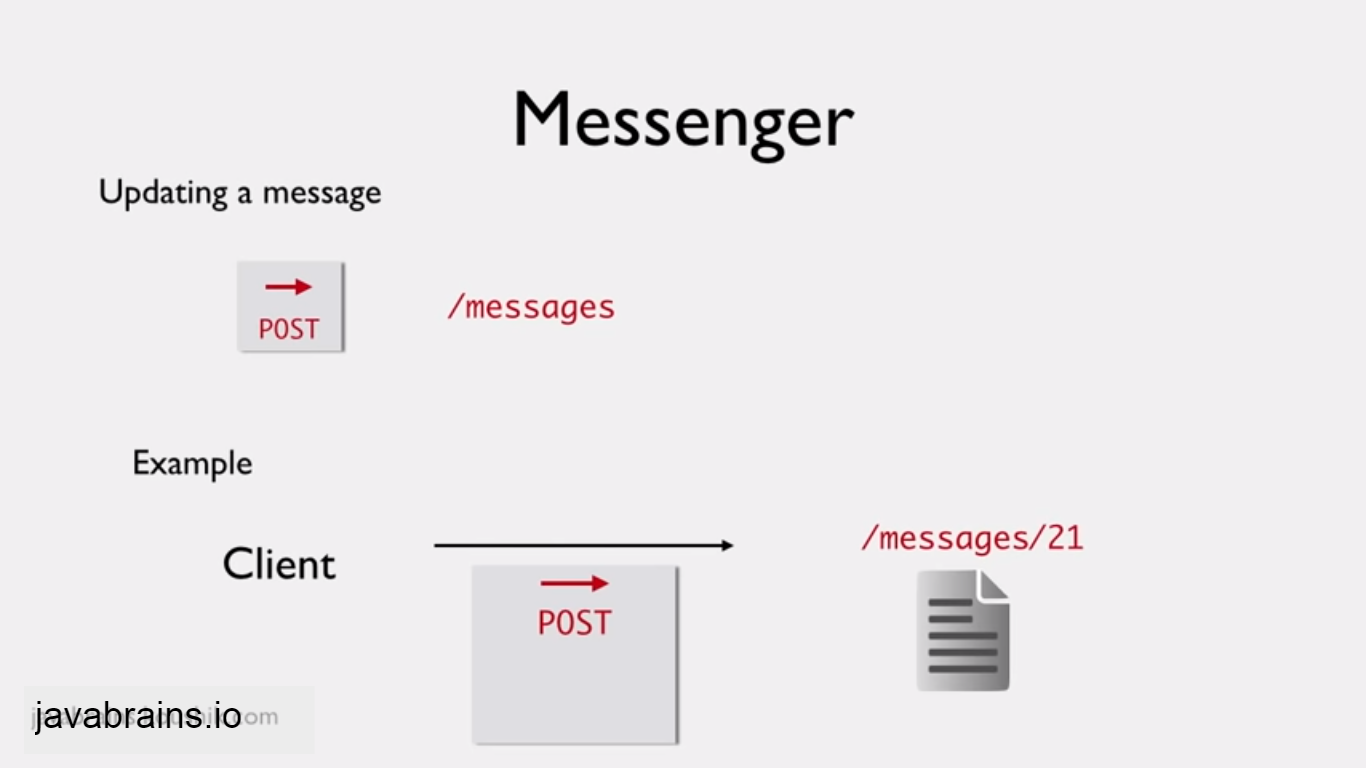
To update: here request is send through request body.

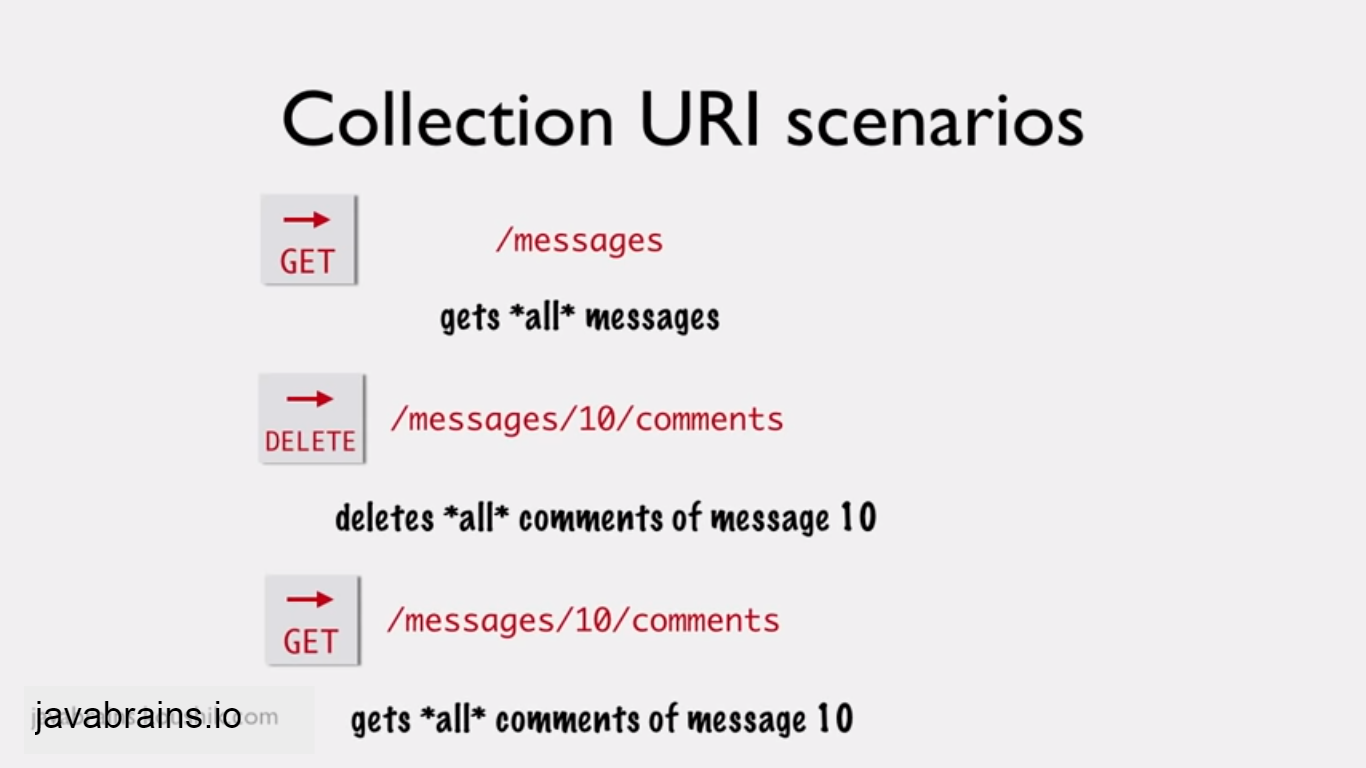


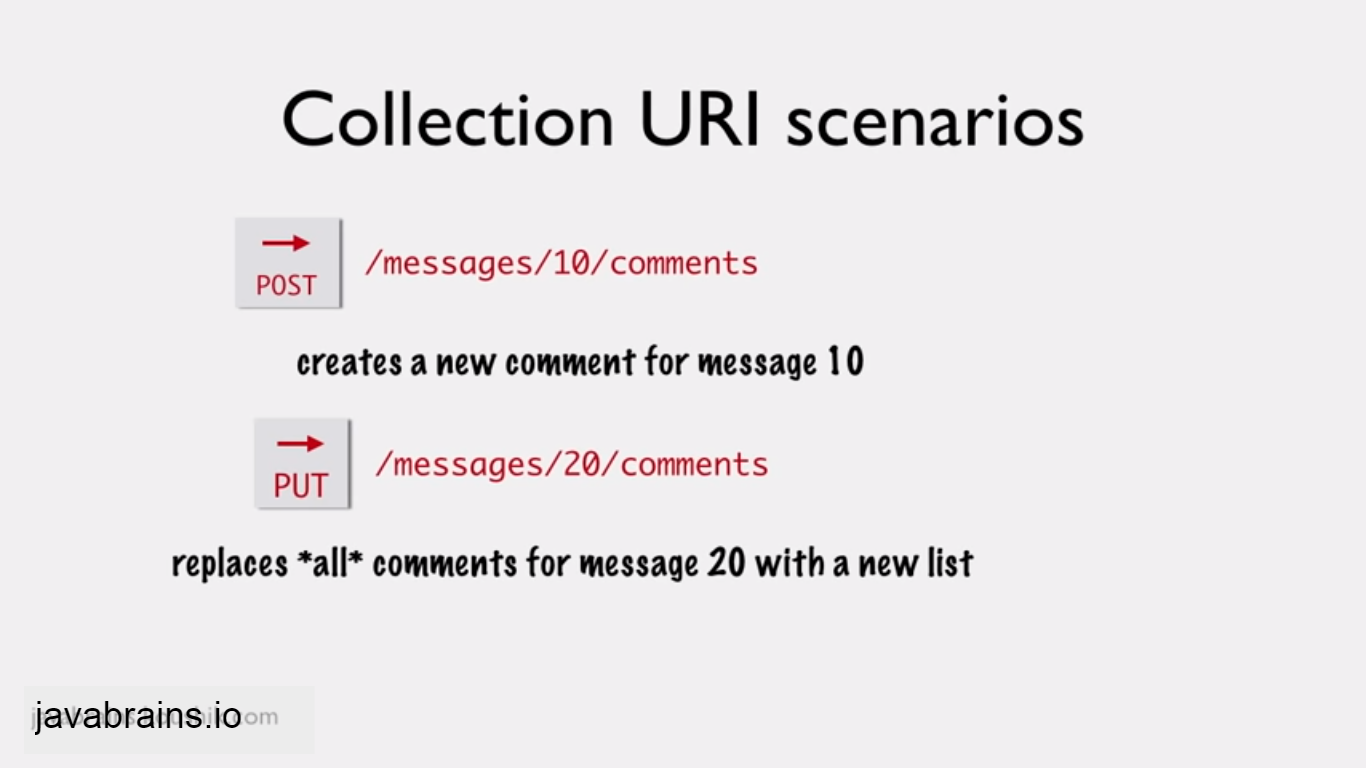
To delete:



To create new message: Post method has request body contain new message.





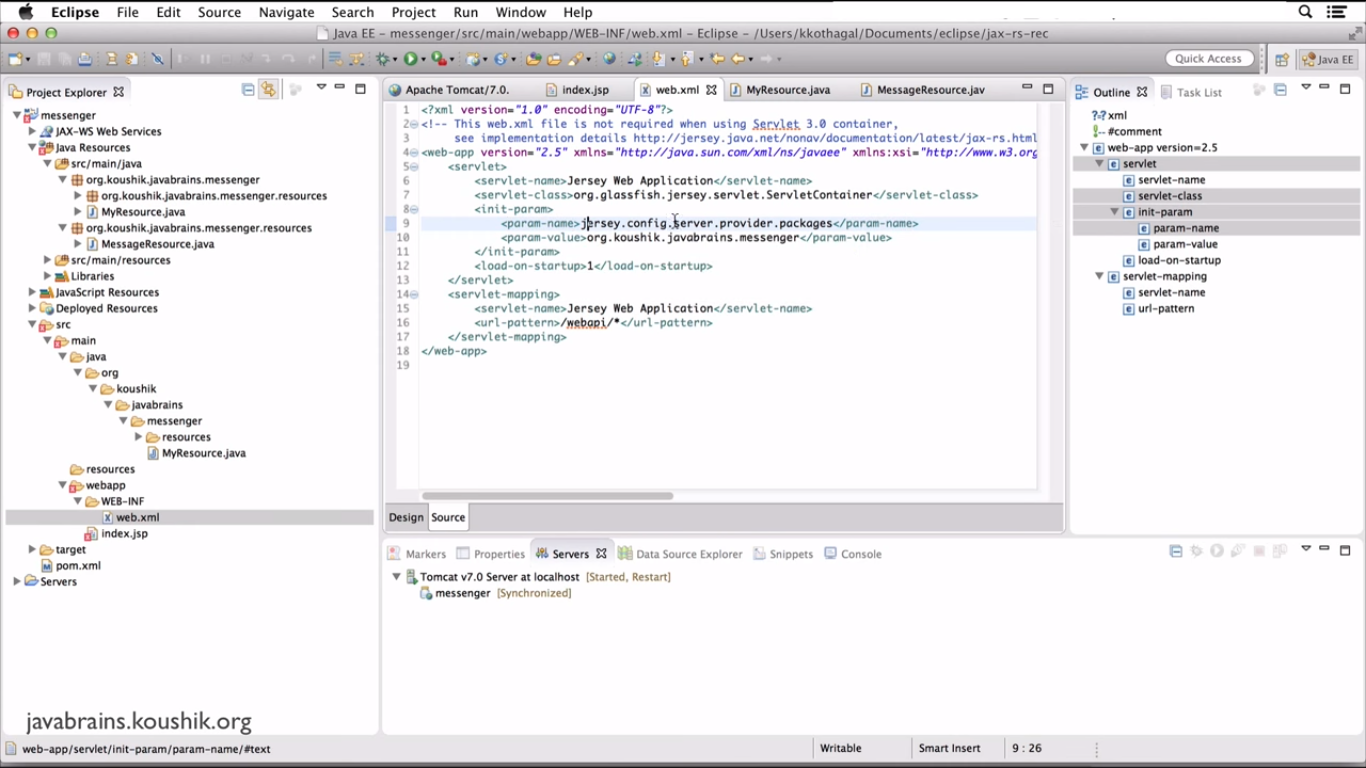


Developing Helloworld using jersey:

To create simple hello world program:

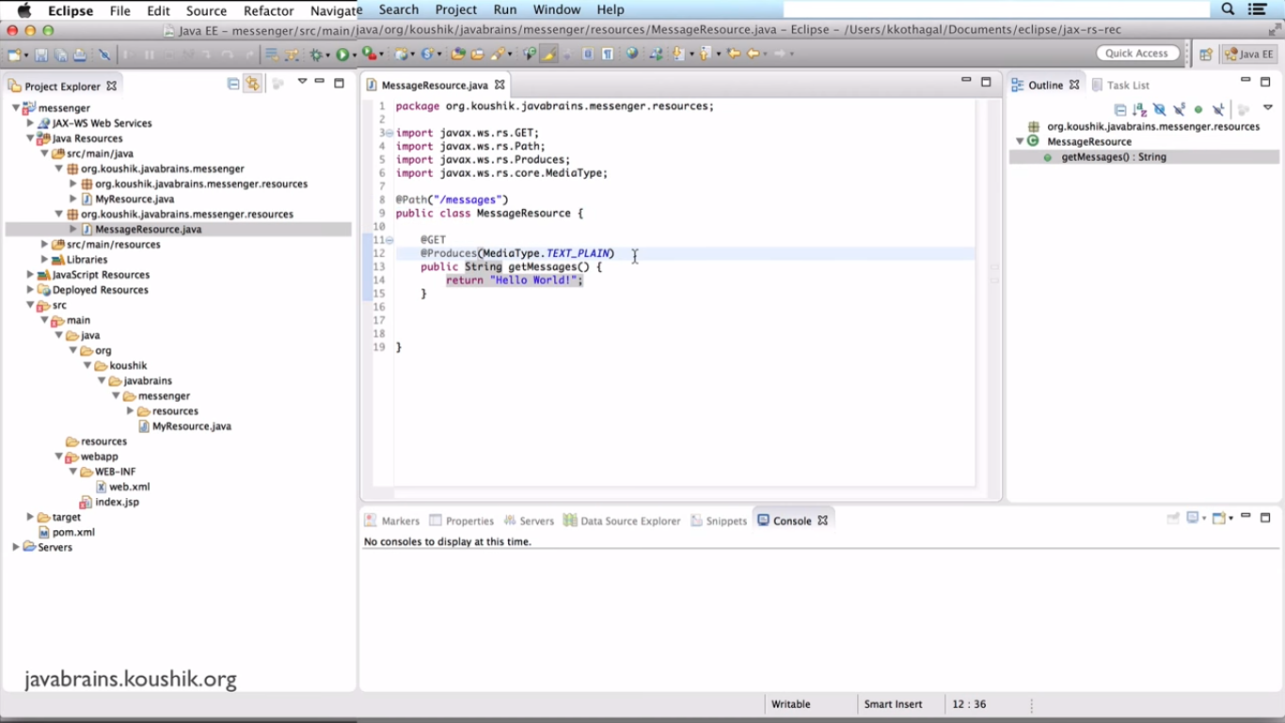
In the package “messenger” we need to mention the main package name in web.xml.

Application will look into the package and then look into the class path name to execute the resources.



Here, the @path tells us the class which is mapped to resources “messenger”.

@GET is used to get the data

@produce is used to mention the format.

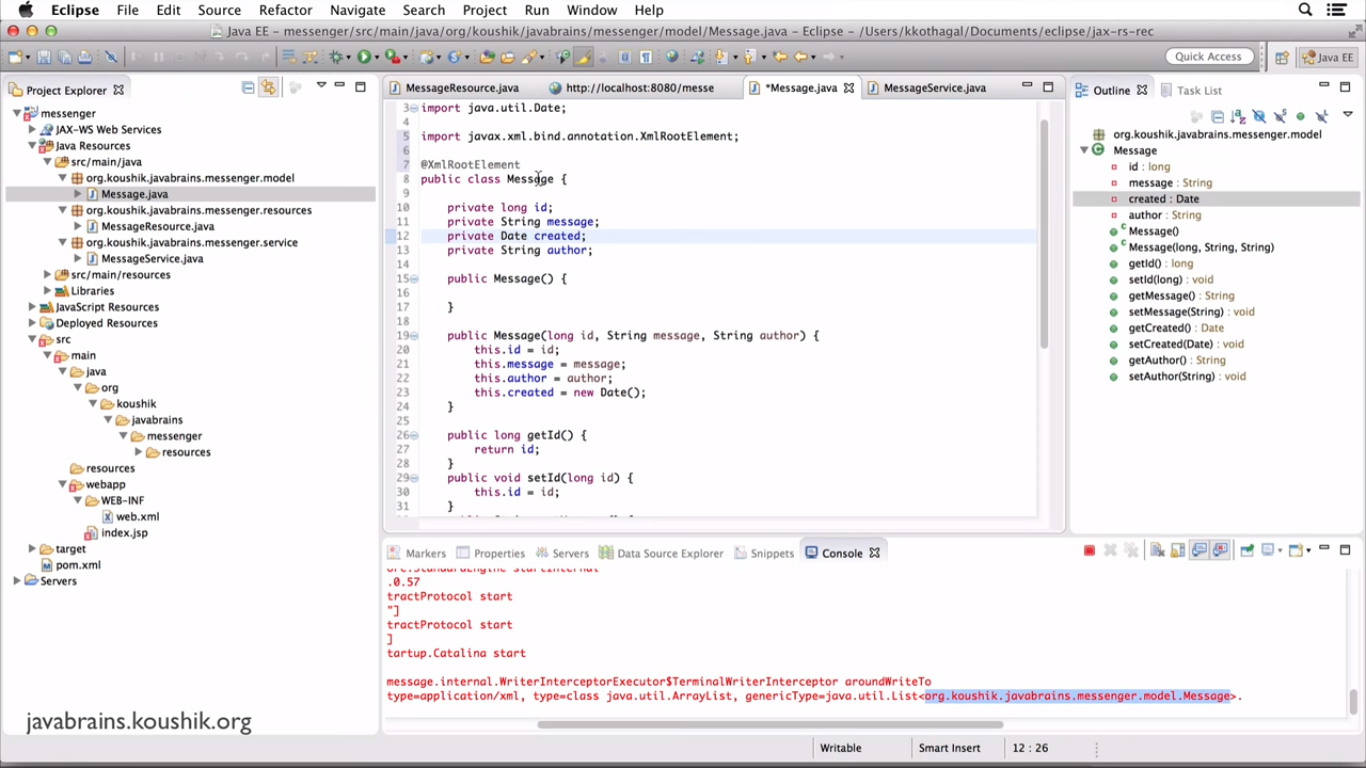
Returning XML Response:

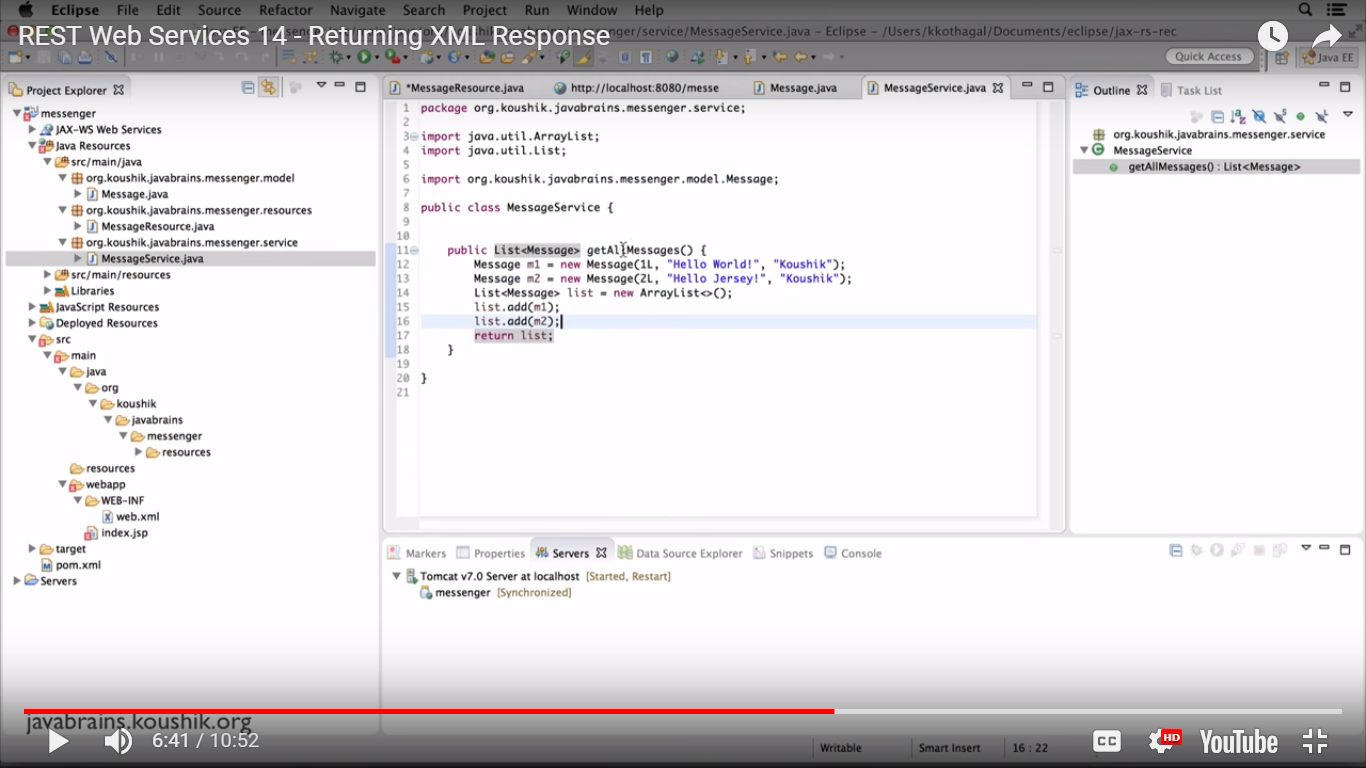
Lets have mock the sample data instead of hitting into the database.

Create sample bean class, service class which can return the list of messages/data.

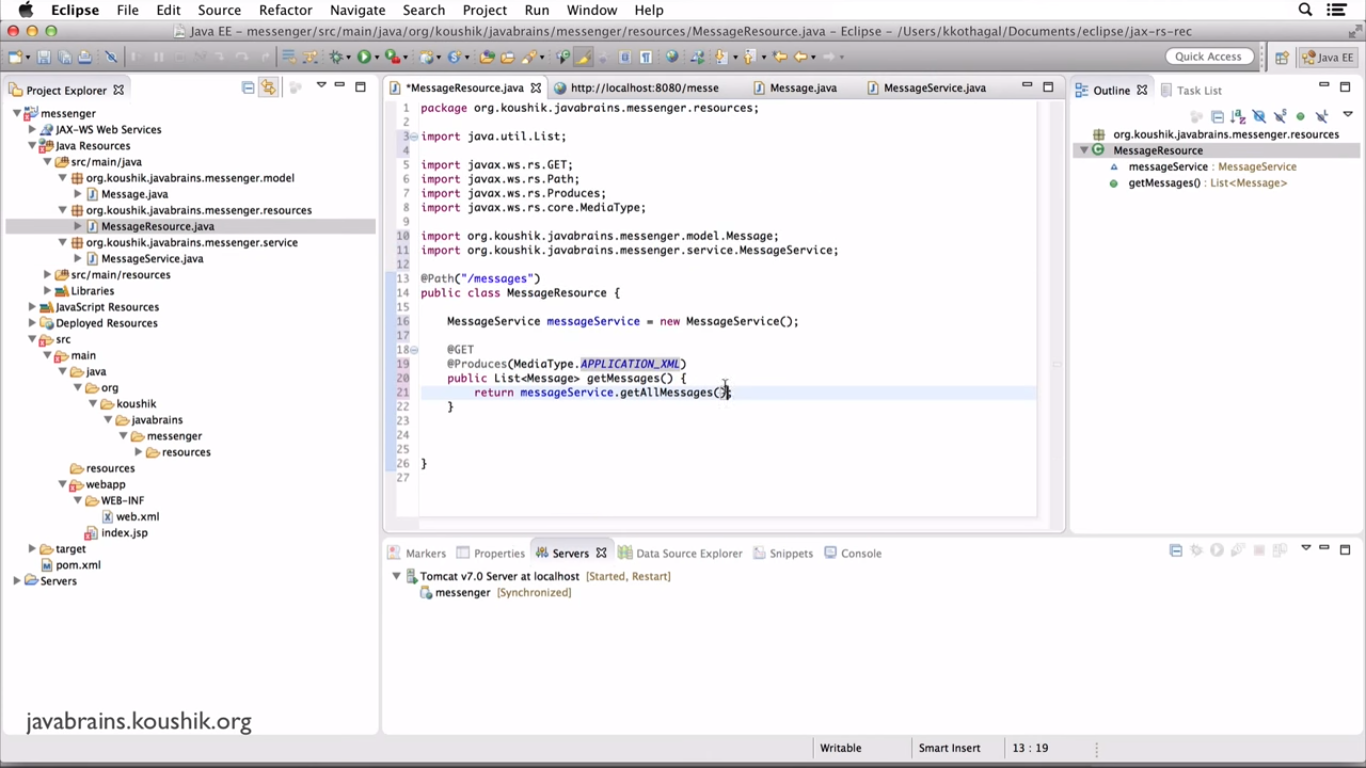
@xmlRootElement is the annotation which tells the **jaxp to convert the member variable into xml element.**

Note: whenever we are sending/receiving xml/json data we need to have empty constructor.

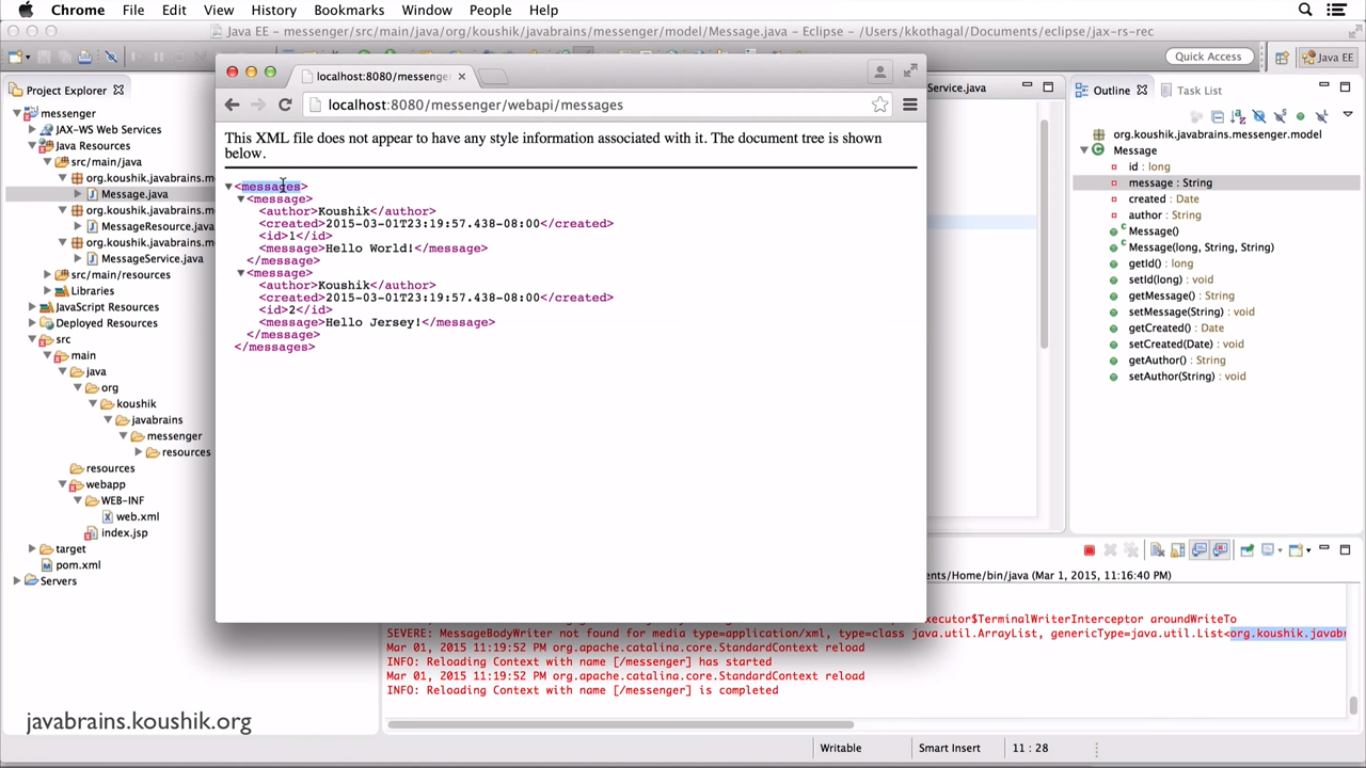
Dummy services class which mocks the data and returns list of messages.



This class tells that getting the list of messages from service in xml format.



Output:



**REST API CLIENT:**

**To send the request and to get the response from the browser we need some tool to test our api.**

**One of them is POSTMAN.** Which is available in chrome webstore.

Building stub services:

