Introduction to REST

Agenda

- What is REST?
- Principles of REST
- What is RESTful?
- How to REST
- REST Programming

REST

- What is REST?
  - REST is an acronym for Representational State Transfer
  - REST is an architectural style which is based on web-standards and the HTTP protocol
  - REST based architecture everything is a resource is accessed via a common interface
  - REST allows resources have different representations likes text, XML, JSON

REST

- Principles of REST
  - Client/Server
  - Stateless
  - Cacheable
  - Layered System
  - Code on Demand
  - Uniform Interface
    - Identification of resources
    - Manipulation of resource through representations
    - Self-descriptive messages
    - Hypermedia as the engine of application state (HATEOAS)

REST

- REST Styles
  - It should be stateless
  - It should access all the resources from the server using only URI
  - It does not have inbuilt encryption
  - It does not have session
  - It uses one and only one protocol – HTTP
  - It should use HTTP methods GET, POST, PUT, DELET to perform CRUD operations
  - It should return the result only in the form of JSON, XML, ATOM, OData

REST

- REST Levels
  - Level 0 – any system that has a single endpoint for all its API
  - Level 1 – a resource URI described system
  - Level 2 – a compliant use of standard HTTP methods and multi status code responses
  - Level 3 – hypermedia included in the response which describes additional calls you can makes

REST

- What is RESTful?
  - RESTful is a web services are based on HTTP methods and the concept of REST
  - RESTful is an API interface that two computer systems use to exchange information securely over the internet

REST

- What are API Authentication methods?
  - HTTP authentication
    - Basic authentication
      - user name and password in the request header
    - Bearer authentication
      - token in the request headers to access resources
  - API Keys
    - the server assigns a unique generated value to a client, the client tries to access resources, it uses the unique API key to verify itself
  - OAuth
    - combines passwords and tokens for highly secure login access to any system

REST

- How to REST
  - Jersey 2.39
    - create an application using the Grizzly 2 HTTP server container

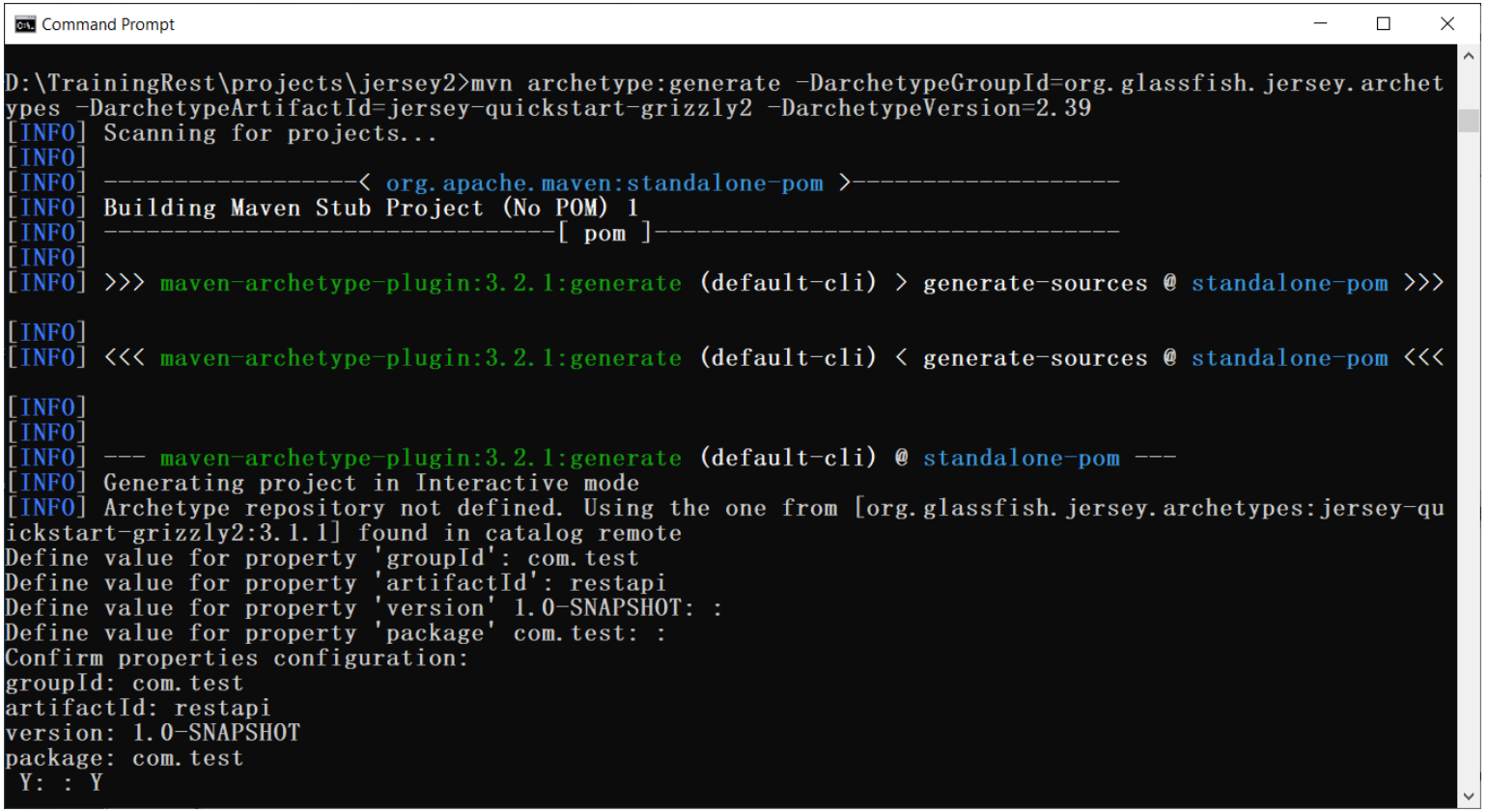
```
mvn archetype:generate -DarchetypeGroupId=org.glassfish.jersey.archetypes \
    -DarchetypeArtifactId=jersey-quickstart-grizzly2 -DarchetypeVersion=2.39
```

    - create a servlet container deployable web application

```
mvn archetype:generate -DarchetypeGroupId=org.glassfish.jersey.archetypes \
    -DarchetypeArtifactId=jersey-quickstart-webapp -DarchetypeVersion=2.39
```

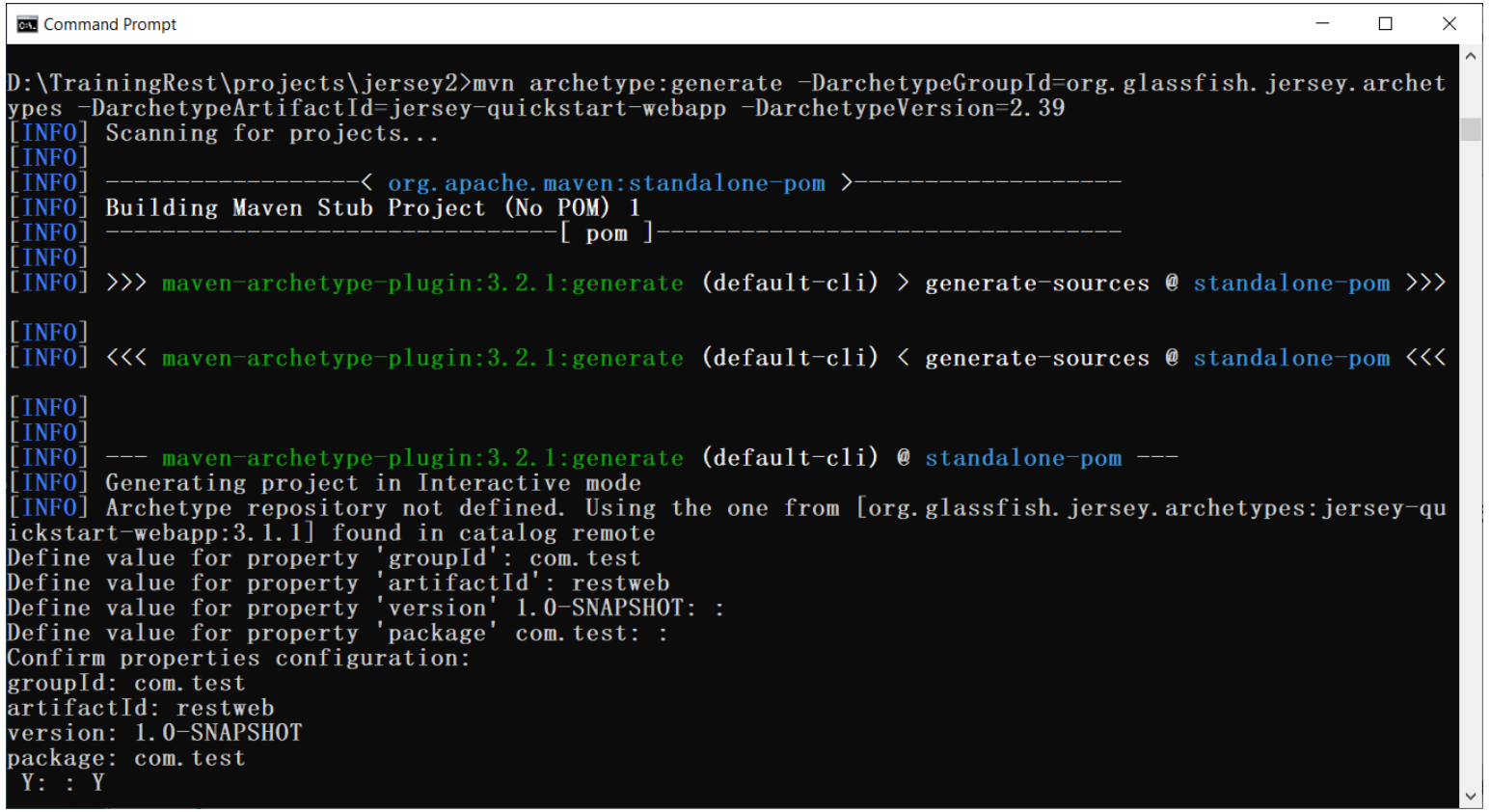
REST

- How to REST
  - Jersey 2.39 - application



REST

- How to REST
  - Jersey 2.39 - web application



REST

- How to REST
  - Jersey 3.1.1 – jakarta project
    - create an application using the Grizzly 3 HTTP server container

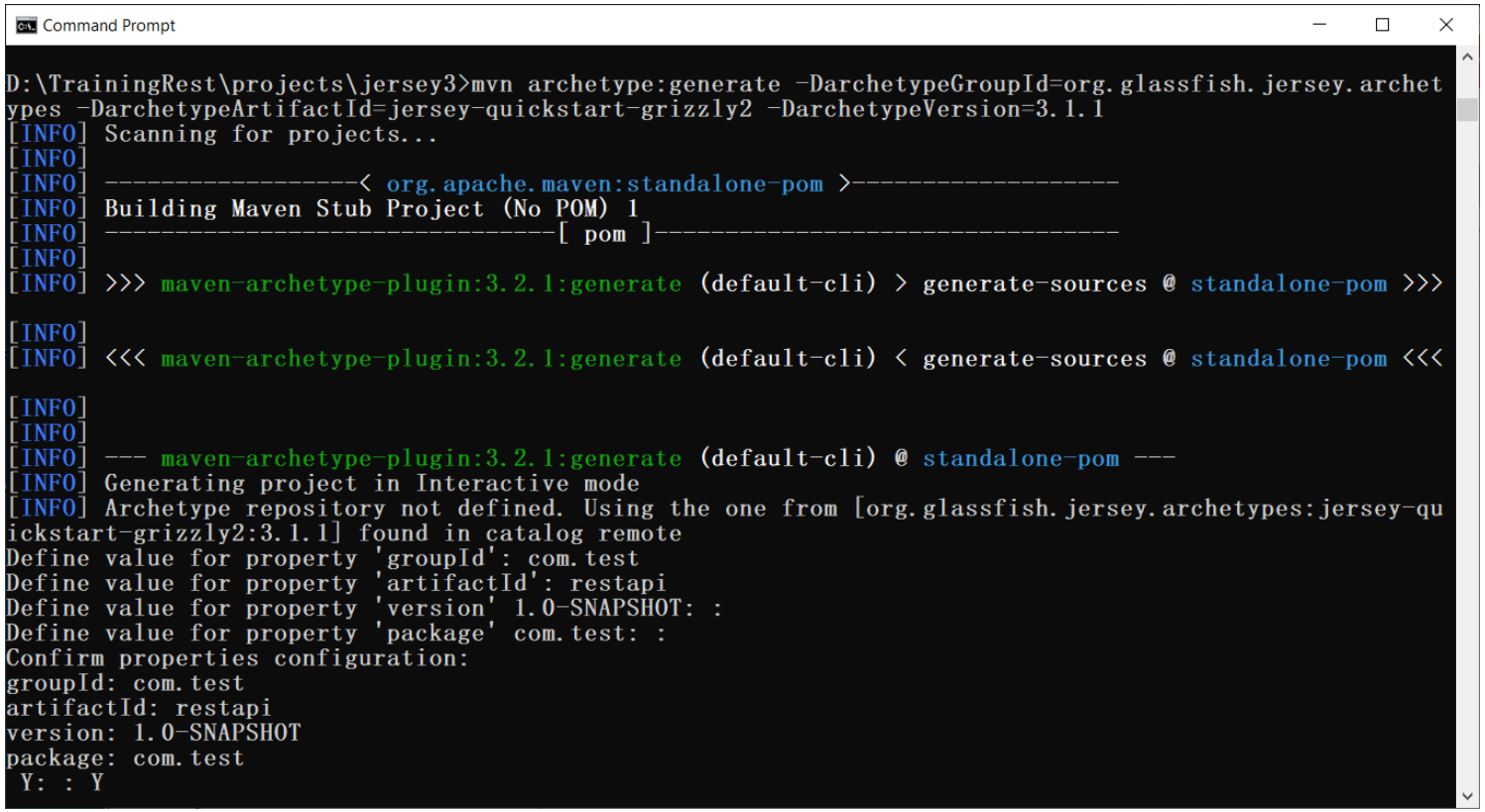
```
mvn archetype:generate -DarchetypeGroupId=org.glassfish.jersey.archetypes \
    -DarchetypeArtifactId=jersey-quickstart-grizzly2 -DarchetypeVersion=3.1.1
```

    - create a servlet container deployable web application

```
mvn archetype:generate -DarchetypeGroupId=org.glassfish.jersey.archetypes \
    -DarchetypeArtifactId=jersey-quickstart-webapp -DarchetypeVersion=3.1.1
```

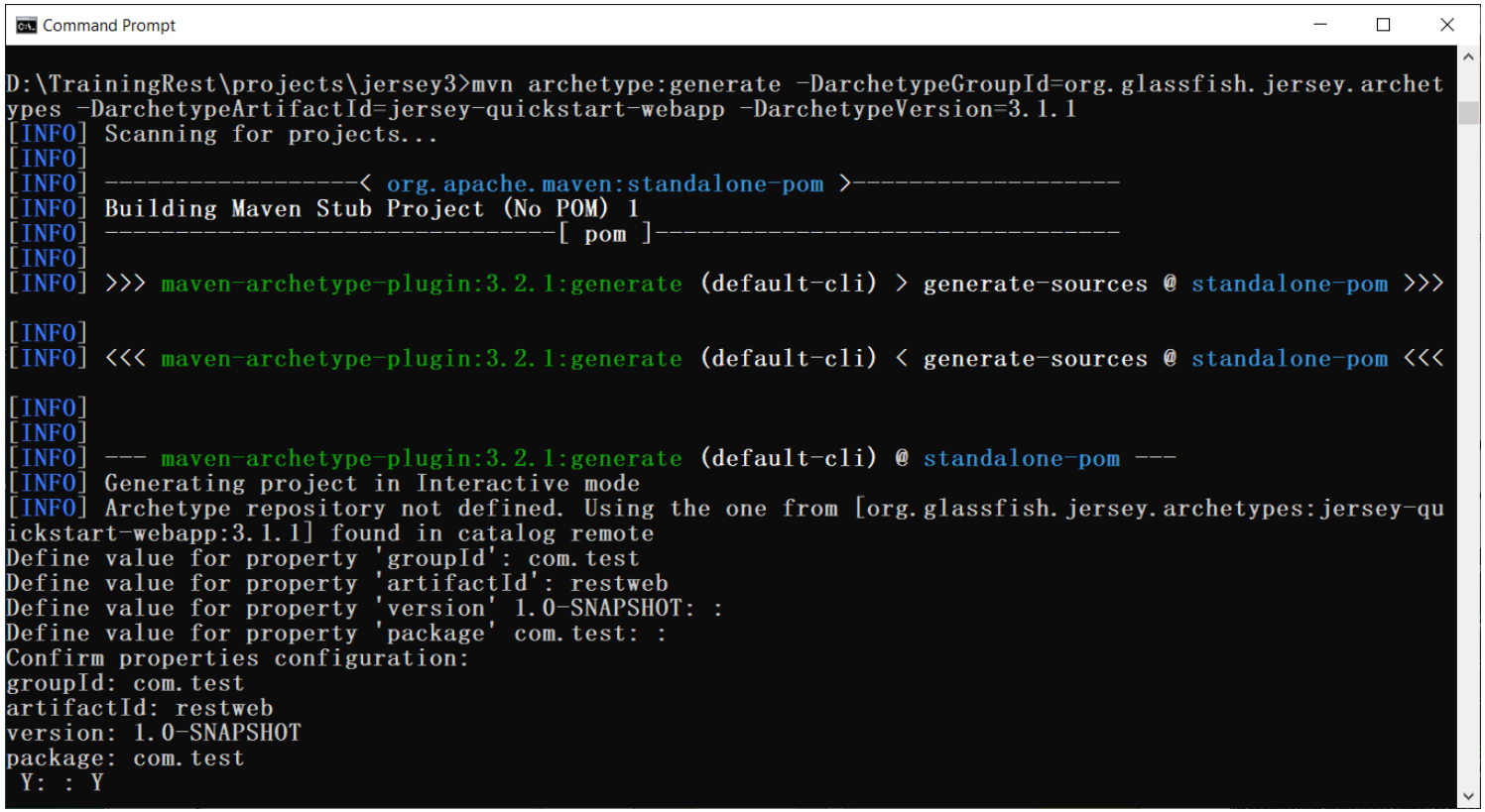
REST

- How to REST
  - Jersey 3.1.1 - application



REST

- How to REST
  - Jersey 3.1.1 - web application



REST

- How to REST
  - MyResource.java

```
package com.test;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("myresource")
public class MyResource {

@GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getIt() {
        return "Got it!";
    }
}
```

REST

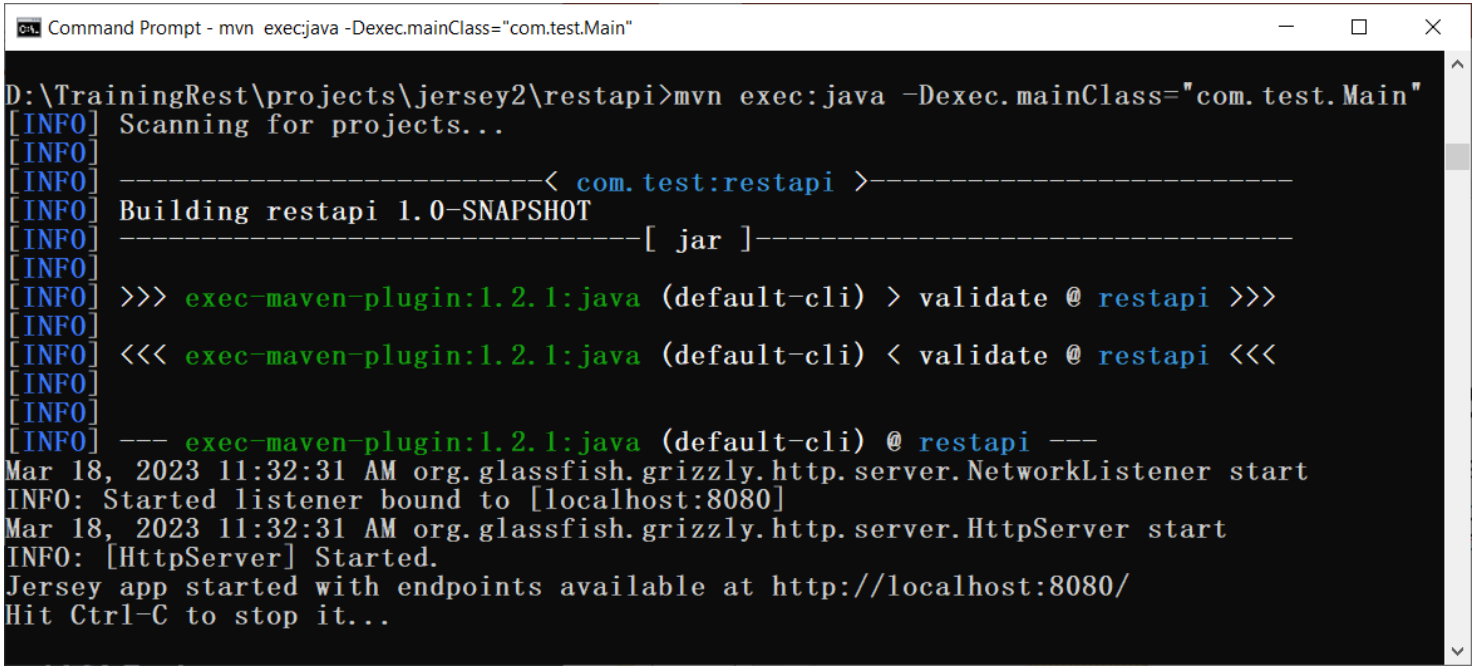
- How to REST
  - MyResource.java - jakarta project

```
package com.test;

import jakarta.ws.rs.GET;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;

@Path("myresource")
public class MyResource {

@GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getIt() {
        return "Got it!";
    }
}
```

REST

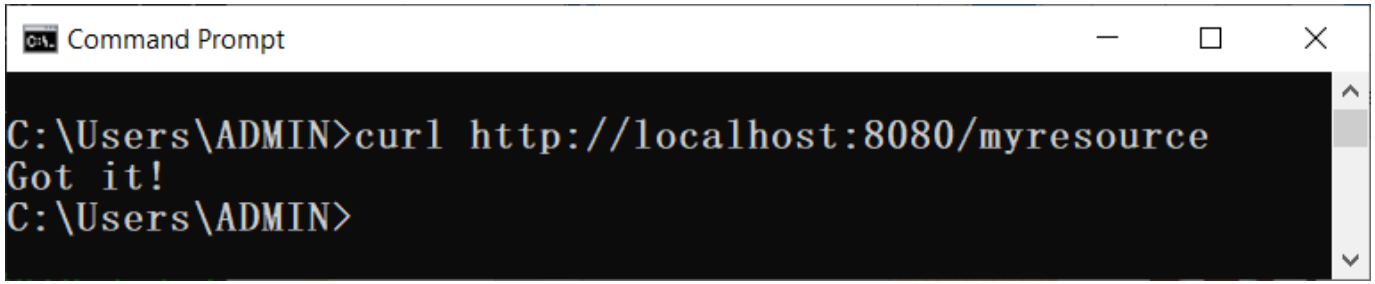
- How to REST
  - Run Application

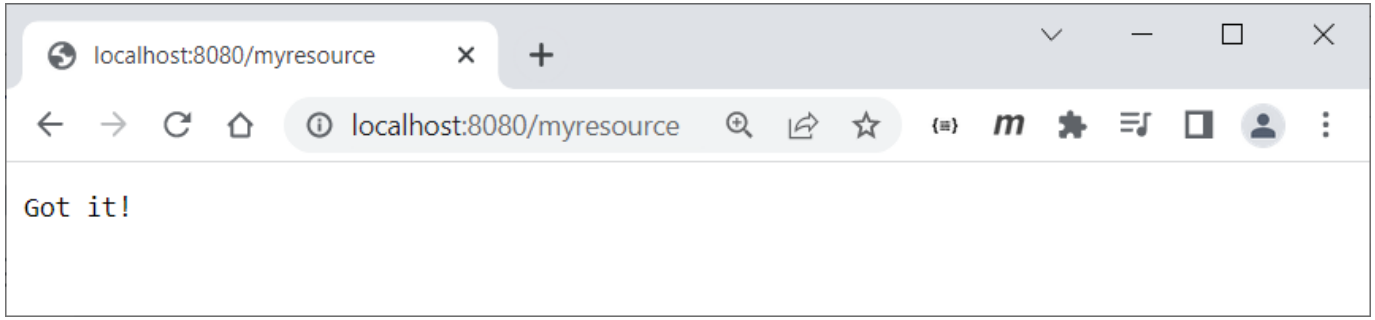
```
mvn exec:java -Dexec.mainClass="com.test.Main"
```



REST

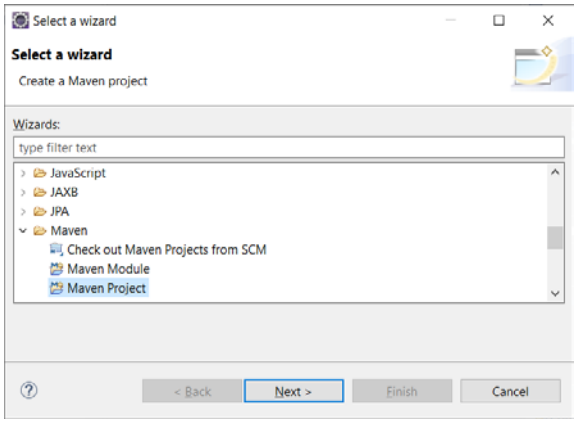
- How to REST
  - Run Application

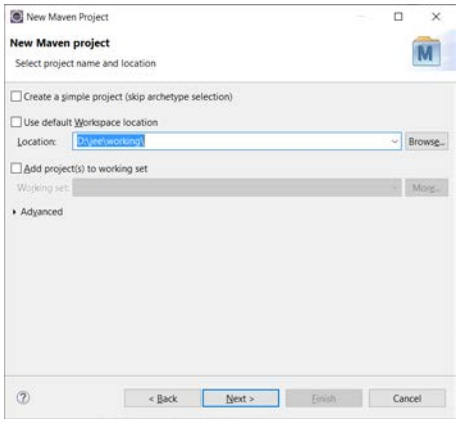


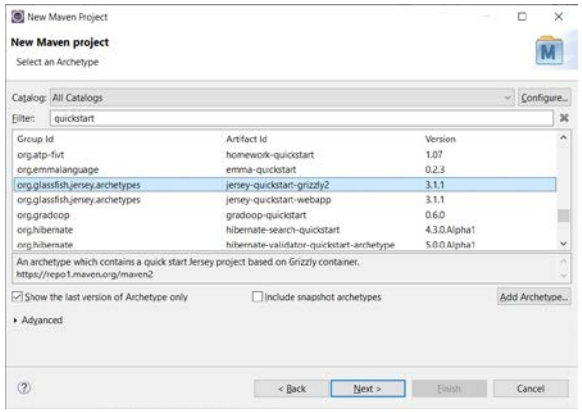


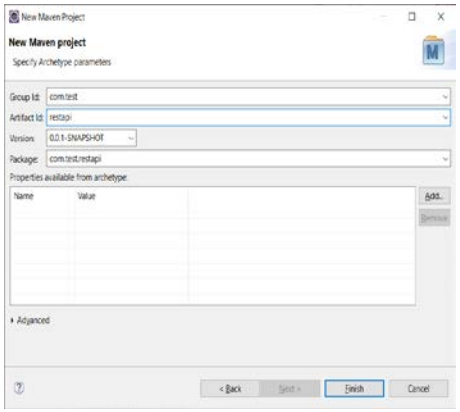
REST

- Eclipse IDE Plugin
  - File -> New -> Other (jar)



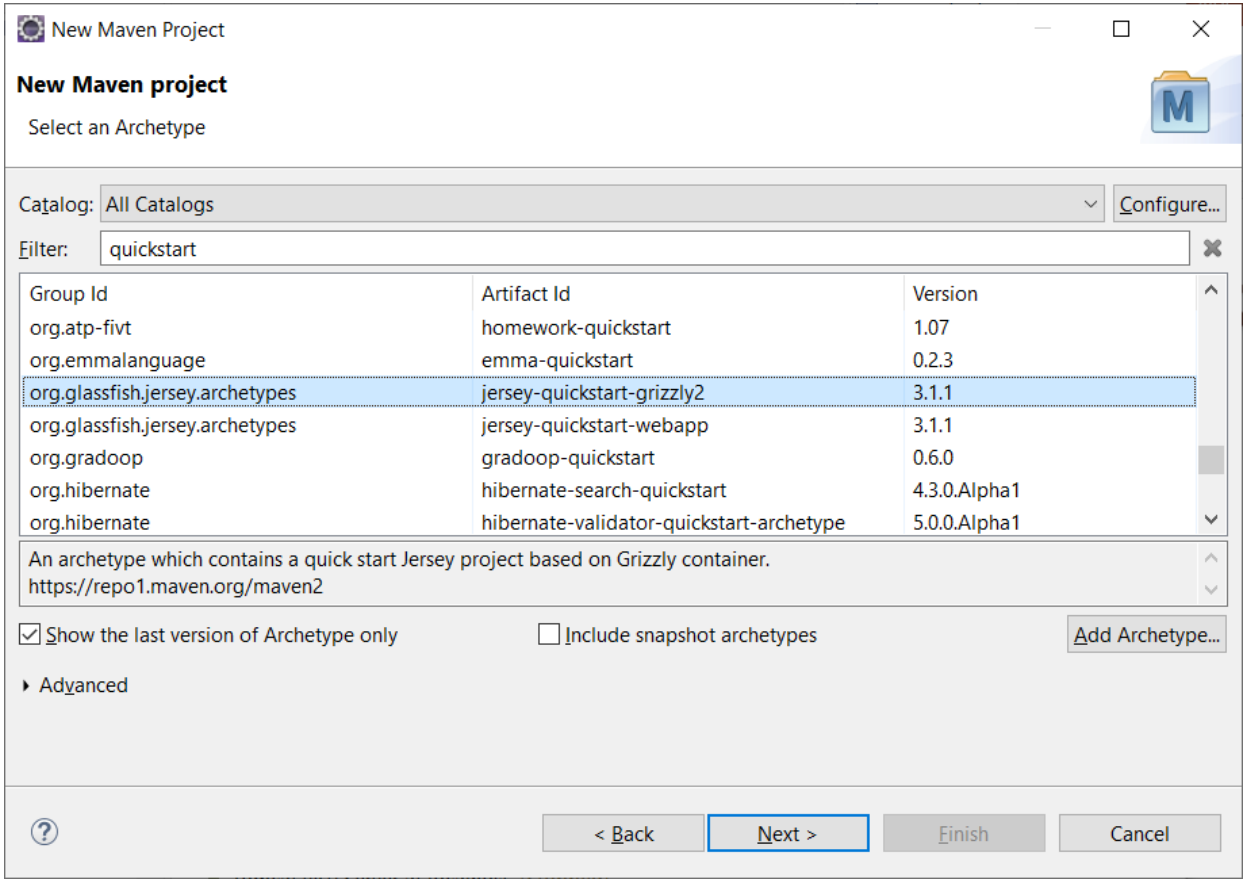






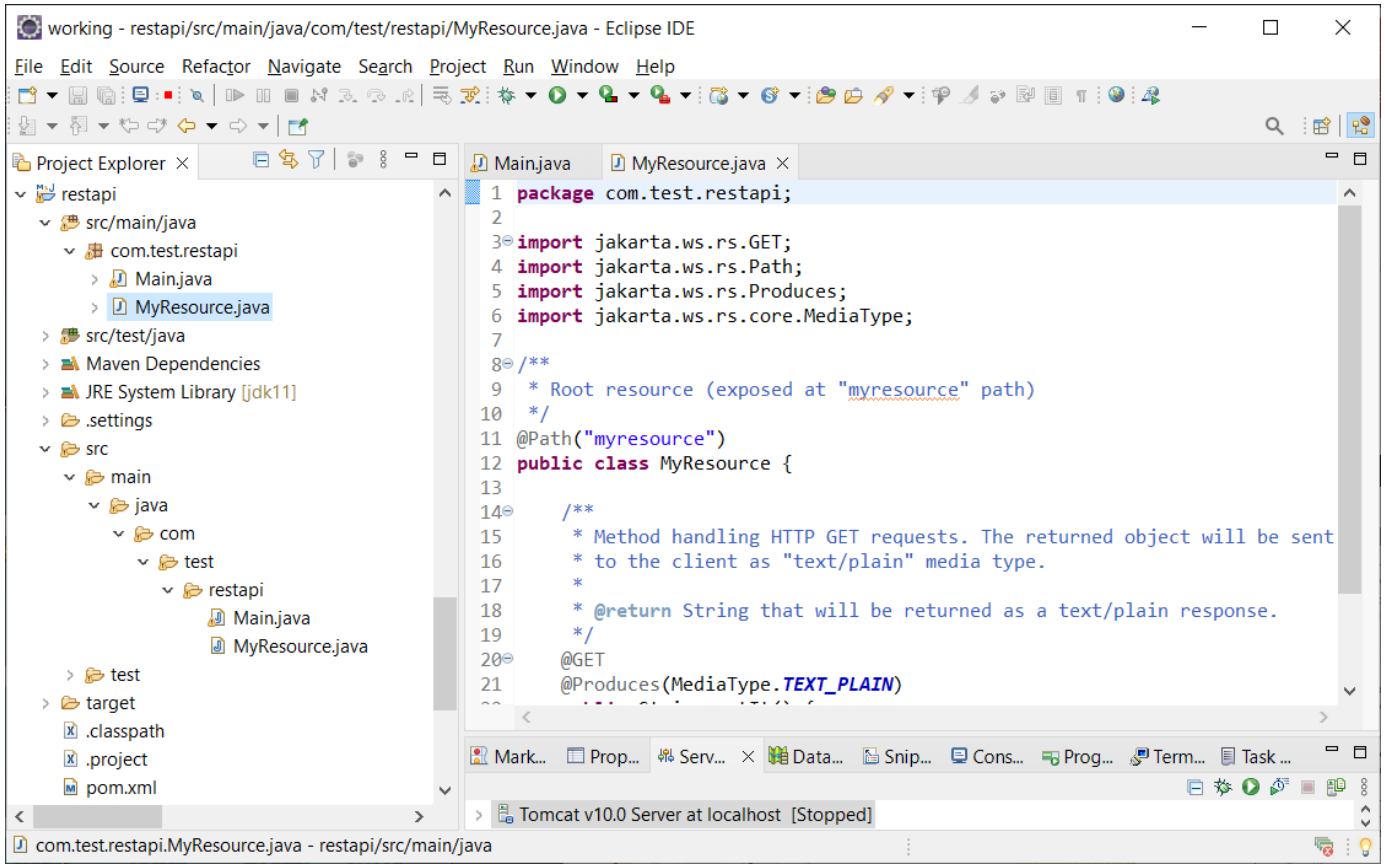
REST

- Eclipse IDE Plugin
  - File -> New -> Other (jar)



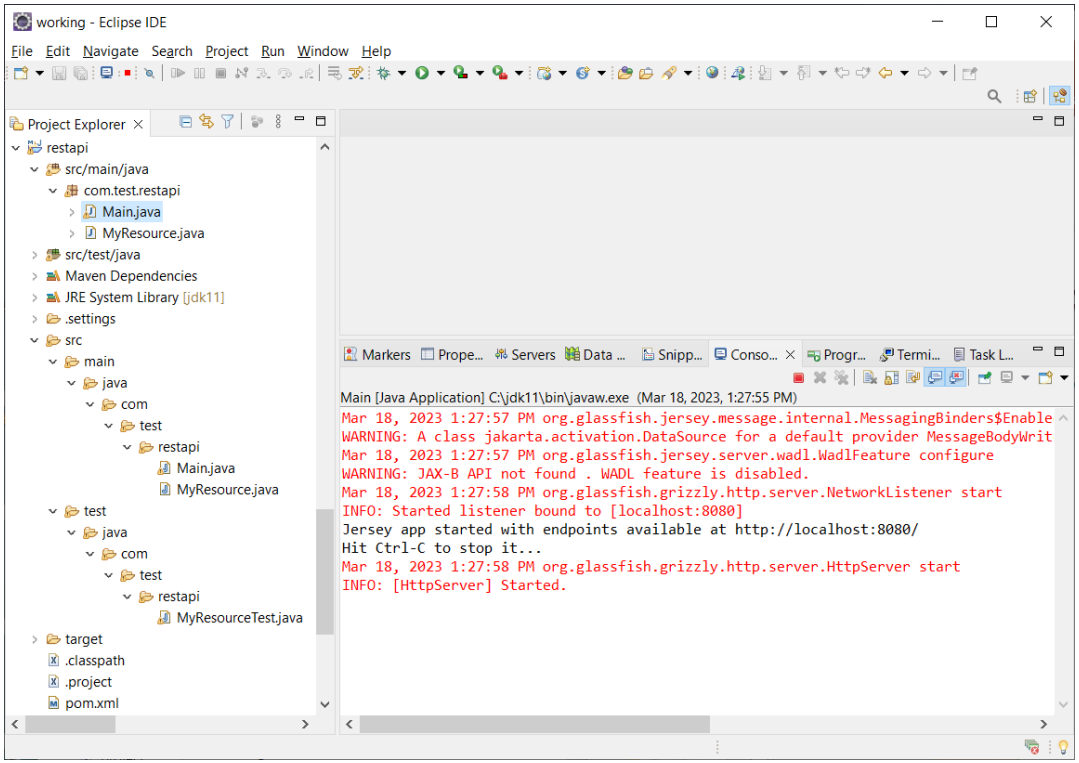
REST

- Eclipse IDE Plugin
  - Java Project



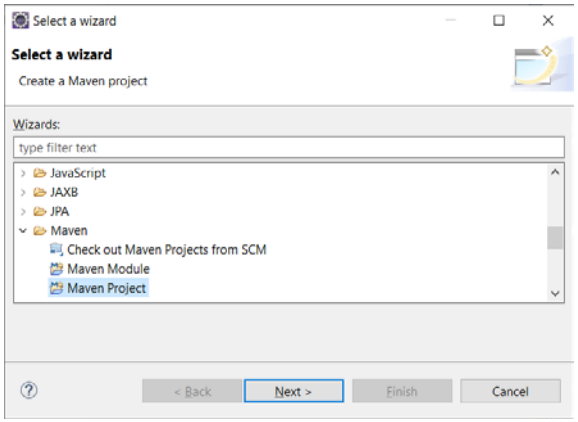
REST

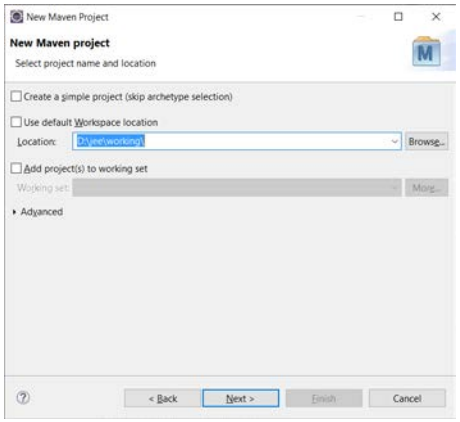
- Eclipse IDE Plugin
  - Run Application
    - Main.java -> right click -> Run As -> Java Application

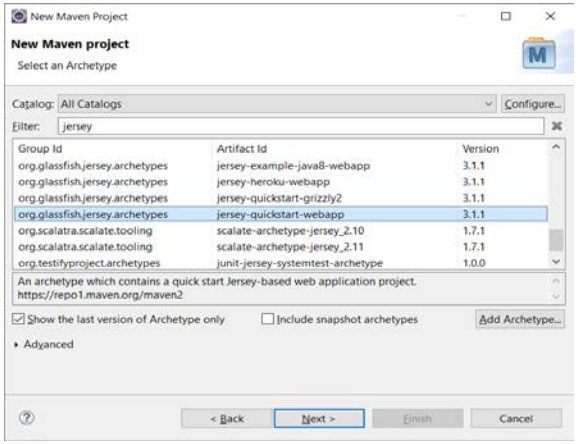


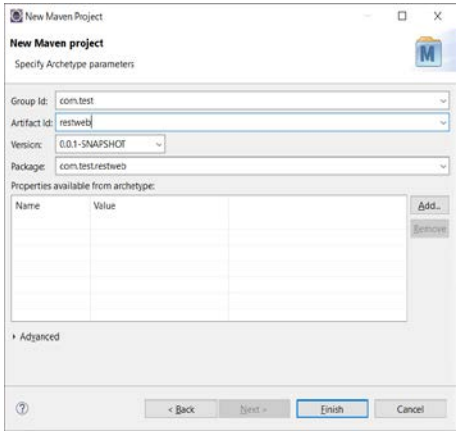
REST

- Eclipse IDE Plugin
  - File -> New -> Other (war)



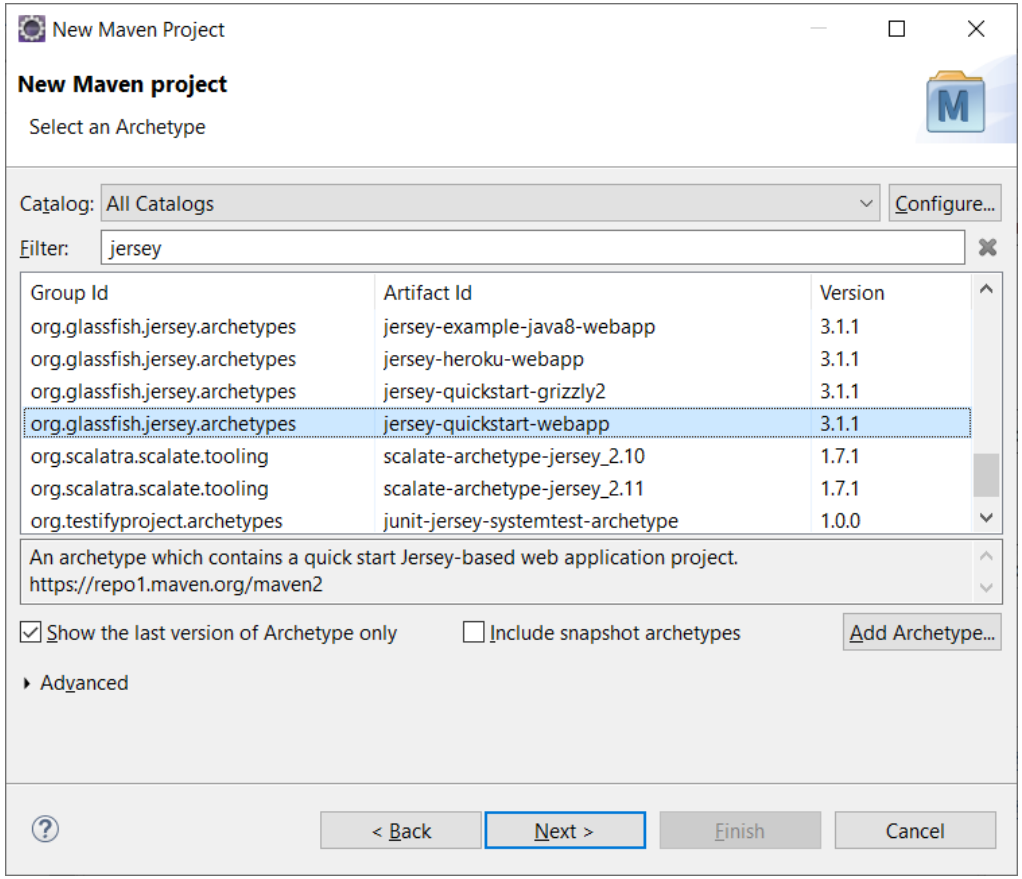






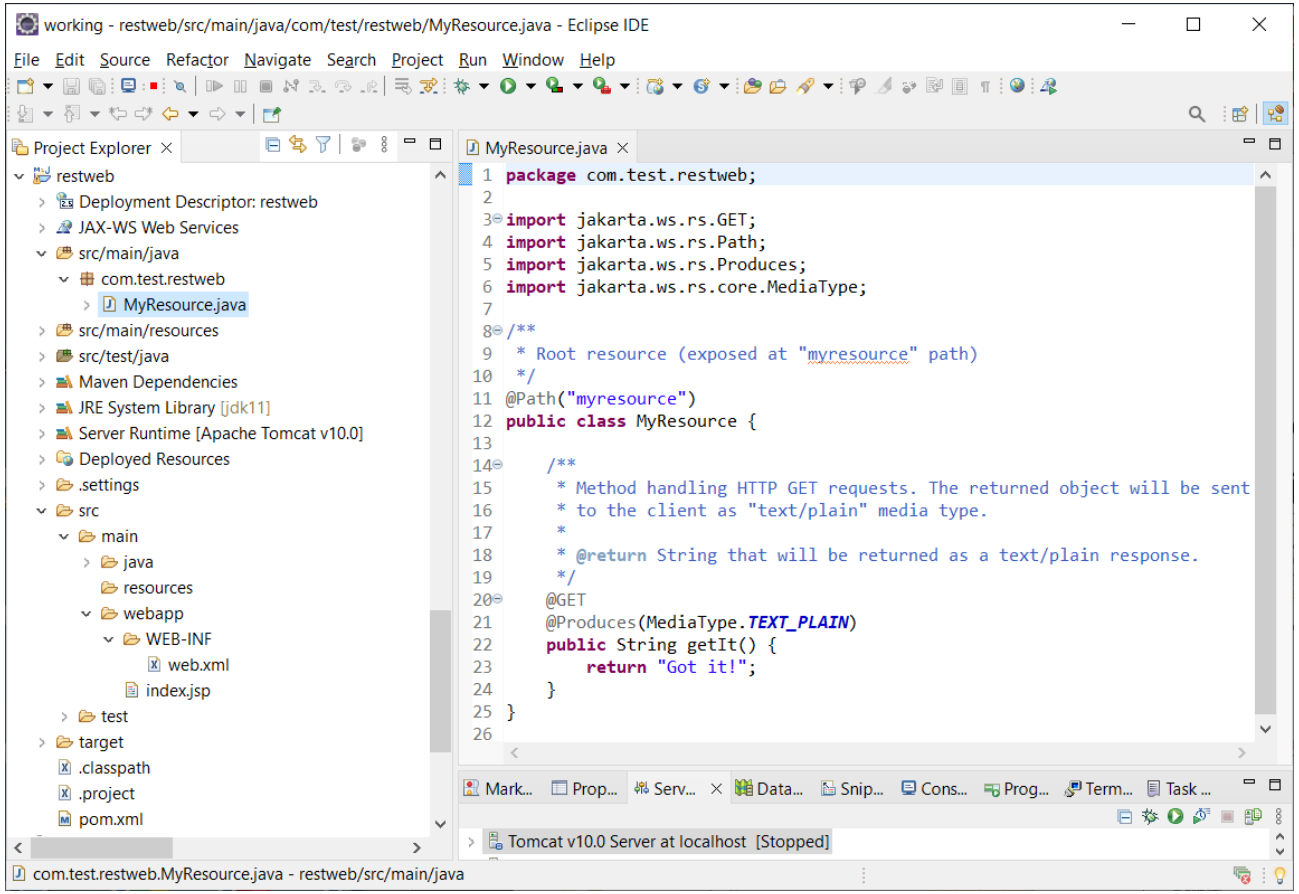
REST

- Eclipse IDE Plugin
  - File -> New -> Other (war)



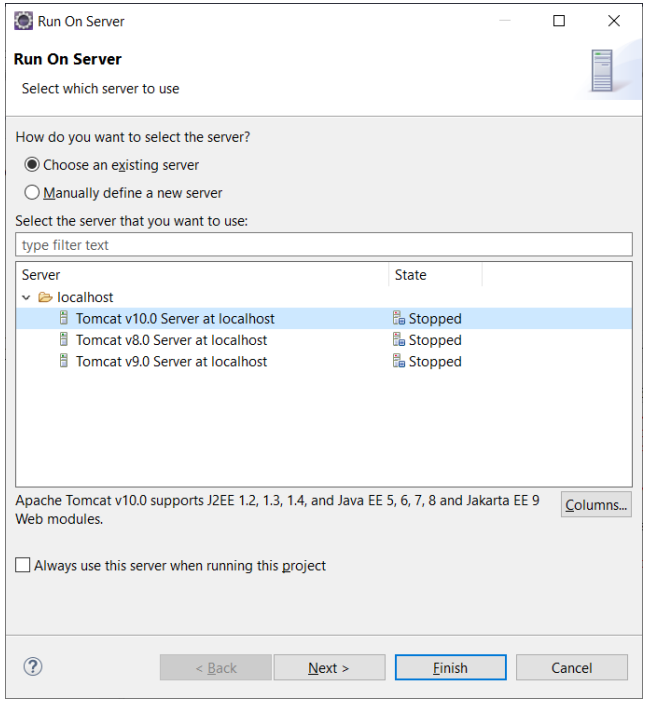
REST

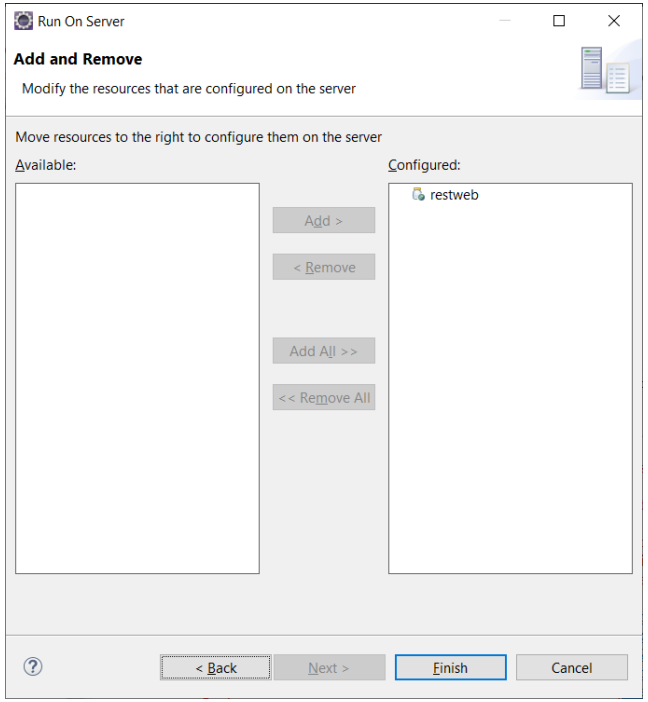
- Eclipse IDE Plugin
  - Web Project



REST

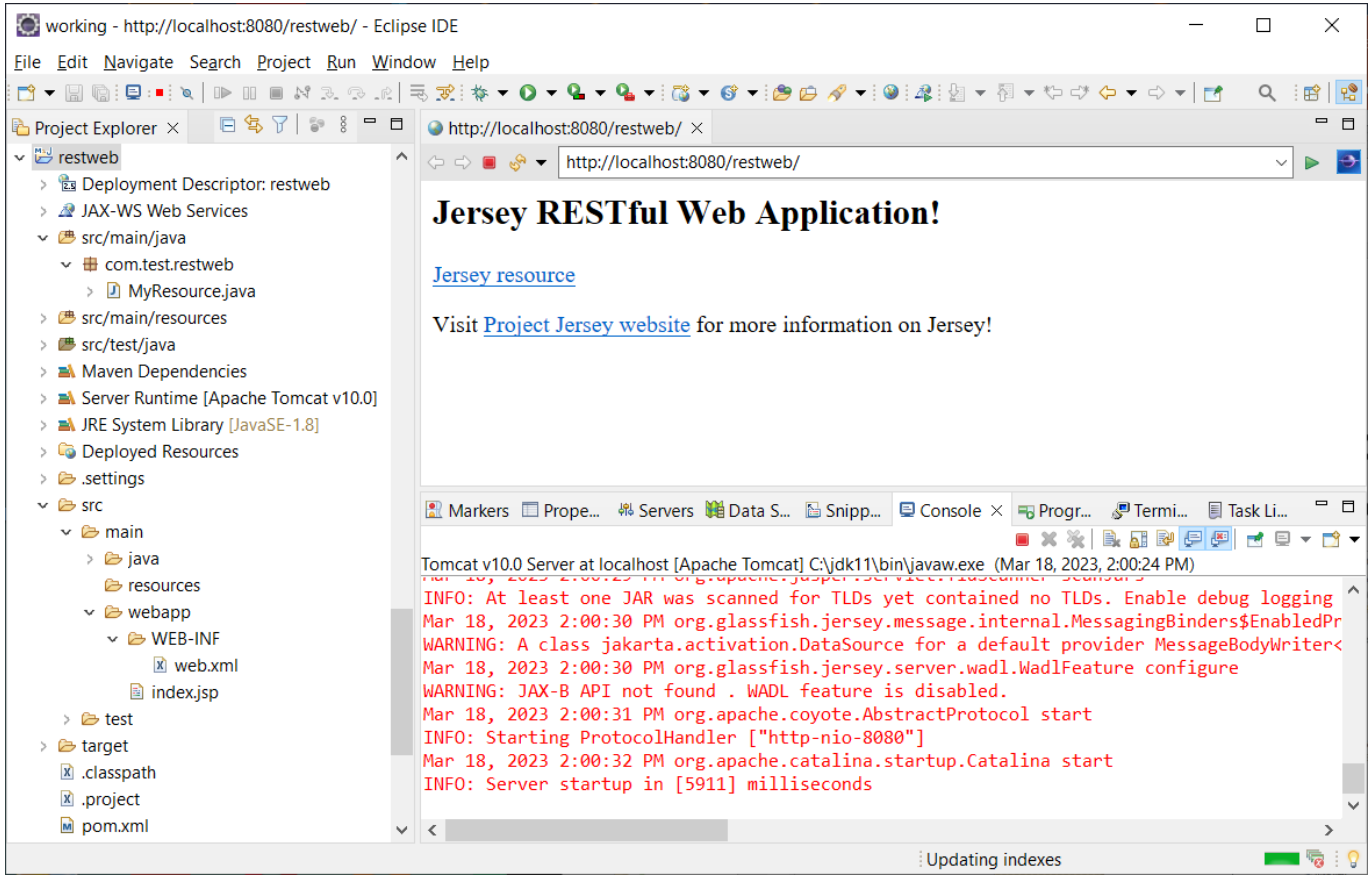
- Eclipse IDE Plugin
  - Run Web Application
    - project -> right click -> Run As -> Run on Server





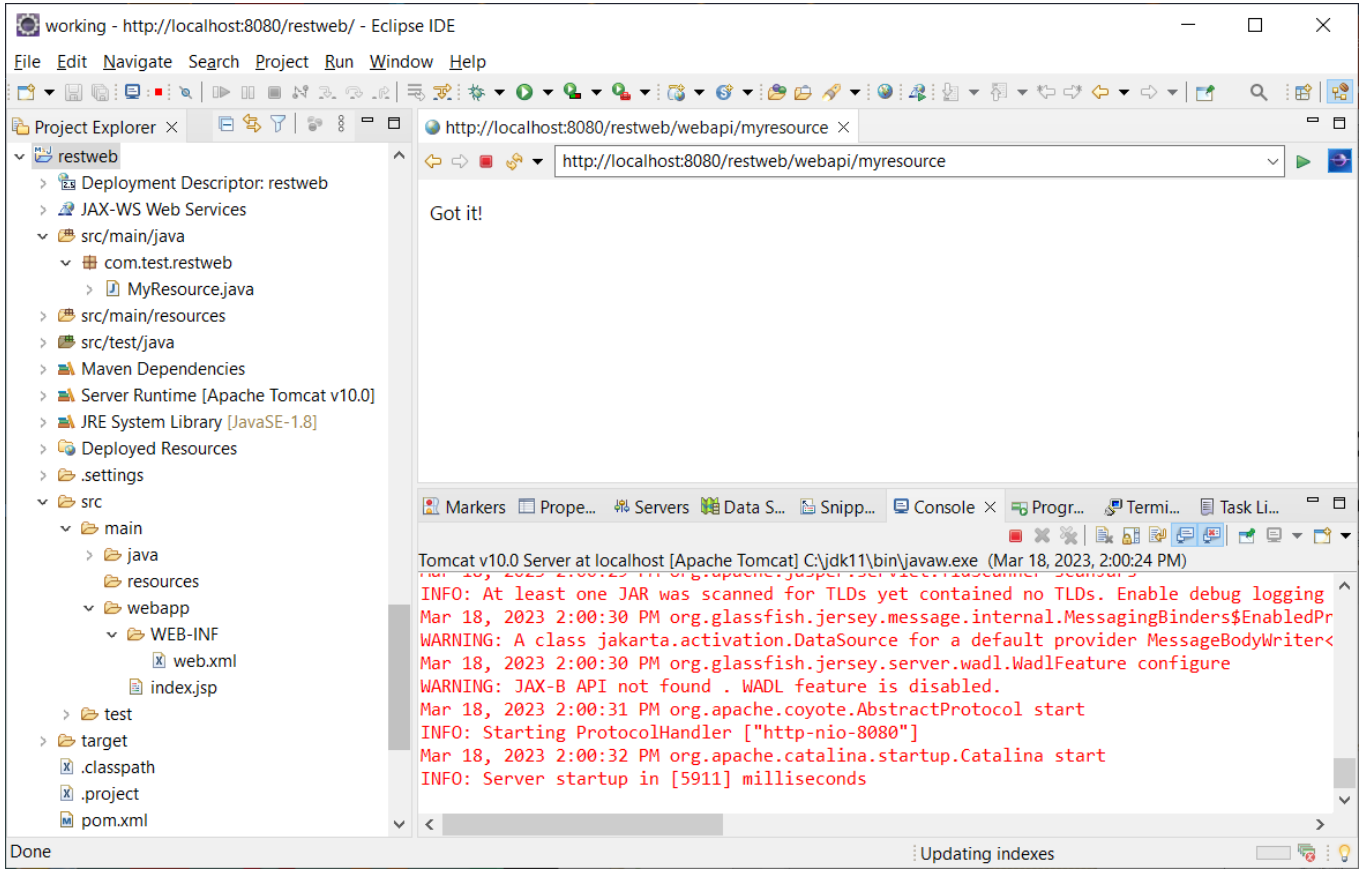
REST

- Eclipse IDE Plugin
  - Run Web Application



REST

- Eclipse IDE Plugin
  - Run Web Application



REST

- Programming
  - Response Plain Text

```
@GET
@Produces(MediaType.TEXT_PLAIN)
public String getIt() {
    return "Got it!";
}
```

- request

```
curl http://localhost:8080/myresource
```

- response

```
Got it!
```

REST

- Programming
  - Response JSON

```
@GET
@Path("/json")
@Produces(MediaType.APPLICATION_JSON)
public String getJson() {
    Map<String,String> map = new HashMap<>();
    map.put("message","Got it!");
    return JSONObject.toJSONString(map);
}
```

```
<dependency>
    <groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1.1</version>
</dependency>
```

REST

- Programming
  - Response JSON
    - request

```
curl http://localhost:8080/myresource/json
```

    - response

```
{"message":"Got it!"}
```

REST

- Programming
  - Query Parameter

```
@GET
@Path("/hi")
@Produces(MediaType.APPLICATION_JSON)
public String hi(@QueryParam("name") String name) {
    Map<String,String> map = new HashMap<>();
    map.put("message","Hi, "+name);
    return JSONObject.toJSONString(map);
}
```

```
curl http://localhost:8080/myresource/hi?name=John
```

```
{"message":"Hi, John"}
```

REST

- Programming
  - Path Parameter

```
@GET
@Path("/hello/{name}")
@Produces(MediaType.APPLICATION_JSON)
public String hello(@PathParam("name") String name) {
    Map<String,String> map = new HashMap<>();
    map.put("message","Hello, "+name);
    return JSONObject.toJSONString(map);
}
```

```
curl http://localhost:8080/myresource/hello/John
```

```
{"message":"Hello, John"}
```

REST

- Programming
  - Form Parameter

```
@POST
@Path("/greet")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
public String greet(@FormParam("name") String
name,@FormParam("surname") String surname) {
    System.out.println("name="+name+",
surname="+surname);
    Map<String,String> map = new HashMap<>();
    map.put("message","Greeting, "+name+" "+surname);
    return JSONObject.toJSONString(map);
}
```

REST

- Programming
  - Form Parameter – Multi Values

```
@POST
@Path("/greeting")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
public String greeting(MultivaluedMap<String, String>
params) {
    System.out.println("params="+params);
    String name = params.getFirst("name");
    String surname = params.getFirst("surname");
    Map<String,String> map = new HashMap<>();
    map.put("message","Greeting, "+name+" "+surname);
    return JSONObject.toJSONString(map);
}
```

REST

- Programming
  - Form Parameter
    - request

```
curl -X POST http://localhost:8080/myresource/greet
-d "name=John&surname=Doe"
```

```
curl -X POST
http://localhost:8080/myresource/greeting -d
"name=John&surname=Doe"
```

    - response

```
{"message":"Greeting, John Doe"}
```

REST

- Programming
  - Raw Data Parameter

```
@POST
@Path("/bonjour")
@Produces(MediaType.APPLICATION_JSON)
public String bonjour(String params) {
    System.out.println("params="+params);
    Map<String,String> map = new HashMap<>();
    map.put("message","Bonjour, "+params);
    return JSONObject.toJSONString(map);
}
```

```
curl -X POST http://localhost:8080/myresource/bonjour
-d "John Doe"
```

```
{"message":"Bonjour, John Doe"}
```

REST

- Programming
  - JSON Parameter

```
@POST
@Path("/xinchao")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public String xinchao(String params) {
    System.out.println("params="+params);
    Map<String,String> map = new HashMap<>();
    map.put("message","Xin chao, ");
    try {
          JSONParser parser = new JSONParser();
          JSONObject json = (JSONObject)parser.parse(params);
          String name = (String)json.get("name");
          String surname = (String)json.get("surname");
          map.put("message","Xin chao, "+name+" "+surname);
    } catch(Exception ex) {
          ex.printStackTrace();
    }
    return JSONObject.toJSONString(map);
}
```

REST

- Programming
  - JSON Parameter
    - request

```
curl -X POST -H "Content-Type: application/json"
http://localhost:8080/myresource/xinchao -d
"{\"name\":\"John\",\"surname\":\"Doe\"}"
```

    - response

```
{"message":"Xin chao, John Doe"}
```

REST

- Programming
  - Response Interface

```
@POST
@Path("/sabaidi")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Response sabaidi(String params) {
    Greet greet = new Greet("Sabaidi, ");
    try {
          JSONParser parser = new JSONParser();
          JSONObject json = (JSONObject)parser.parse(params);
          String name = (String)json.get("name");
          String surname = (String)json.get("surname");
          greet.setMessage("Sabaidi, "+name+" "+surname);
    } catch(Exception ex) {
          ex.printStackTrace();
    }
    return Response.ok(greet).build();
}
```

REST

- Programming
  - Response Interface – Greet.java

```
package com.test.restapi;

public class Greet {
    private String message;
    public Greet() { super(); }
    public Greet(String message) {
        setMessage(message);
    }
    public String getMessage() { return message; }
    public void setMessage(String message) {
        this.message = message;
    }
    public String toString() {
        return super.toString()+"{message="+message+"}";
    }
}
```

REST

- Programming
  - Response Interface
    - dependency

```
<dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-json-jackson</artifactId>
</dependency>
```

    - or

```
<dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-json-binding</artifactId>
</dependency>
```

REST

- Programming
  - Response Interface
    - request

```
curl -X POST -H "Content-Type: application/json"
http://localhost:8080/myresource/sabaidi -d
"{\"name\":\"John\",\"surname\":\"Doe\"}"
```

    - response

```
{"message":"Sabaidi, John Doe"}
```

REST

- Programming
  - Java Class Parameter

```
@POST
@Path("/nihao")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Greet nihao(Account account) {
   System.out.println("account="+account);
   return new Greet("Nihao, "+account.getFullName());
}
```

```
curl -X POST -H "Content-Type: application/json"
http://localhost:8080/myresource/nihao -d
"{\"name\":\"John\",\"surname\":\"Doe\"}"
```

```
{"message":"Nihao, John Doe"}
```

REST

- Programming
  - Java Class Parameter – Account.java

```
package com.test.restapi;

public class Account {
	private String name;
	private String surname;
	public Account() { super(); }
	public String getName() { return name; }
	public void setName(String name) {
		this.name = name;
	}
	public String getSurname() { return surname; }
	public void setSurname(String surname) {
		this.surname = surname;
	}
	public String getFullName() {
		return name+" "+surname;
	}
	public String toString() {
		return super.toString()+"{name="+name+", surname="+surname+"}";
	}
}
```

REST

- Programming
  - Response XML

```
@POST
@Path("/hallo")
@Produces(MediaType.APPLICATION_XML)
@Consumes(MediaType.APPLICATION_JSON)
public Hello hallo(Account account) {
   System.out.println("account="+account);
   return new Hello("Hallo,"+account.getFullName());
}
```

```
<dependency>
    <groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1.1</version>
</dependency>
```

REST

- Programming
  - Response XML
    - dependency

```
<dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-moxy</artifactId>
</dependency>

<dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-jaxb</artifactId>
</dependency>
```

REST

- Programming
  - Response XML
    - request

```
curl -X POST -H "Content-Type: application/json"
http://localhost:8080/myresource/hallo -d
"{\"name\":\"John\",\"surname\":\"Doe\"}"
```

    - response

```
<?xml version="1.0" encoding="UTF-8"?>
<hello><message>Hallo, John Doe</message></hello>
```

REST

- Programming
  - Response XML – Hello.java

```
package com.test.restapi;

import jakarta.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="hello")
public class Hello {
    private String message;
    public Hello() { super(); }
    public Hello(String message) {
        setMessage(message);
    }
    public String getMessage() { return message; }
    public void setMessage(String message) {
        this.message = message;
    }
}
```

REST

- Programming
  - Multiple Produces

```
@POST
@Path("/hola")
@Produces({MediaType.APPLICATION_JSON,
MediaType.APPLICATION_XML})
@Consumes(MediaType.APPLICATION_JSON)
public Hello hola(Account account) {
    System.out.println("account="+account);
    return new Hello("Hola,
"+account.getFullName());
}
```

REST

- Programming
  - Multiple Produces – response json
    - request

```
curl -X POST -H "Accept: application/json" -H
"Content-Type: application/json"
http://localhost:8080/myresource/hola -d
"{\"name\":\"John\",\"surname\":\"Doe\"}"
```

    - response

```
{"message":"Hola, John Doe"}
```

REST

- Programming
  - Multiple Produces – response xml
    - request

```
curl -X POST -H "Accept: application/xml" -H
"Content-Type: application/json"
http://localhost:8080/myresource/hola -d
"{\"name\":\"John\",\"surname\":\"Doe\"}"
```

    - response

```
<?xml version="1.0" encoding="UTF-8"?>
<hello><message>Hola, John Doe</message></hello>
```

REST

- Programming
  - Multiple Consumes

```
@POST
@Path("/konnichiwa")
@Produces(MediaType.APPLICATION_JSON)
@Consumes({MediaType.APPLICATION_JSON,
MediaType.APPLICATION_XML})
public Hello konnichiwa(Account account) {
    System.out.println("account="+account);
    return new Hello("Konnichiwa,
"+account.getFullName());
}
```

REST

- Programming
  - Multiple Consumes – request json
    - request

```
curl -X POST -H "Content-Type: application/json"
http://localhost:8080/myresource/konnichiwa -d
"{\"name\":\"John\",\"surname\":\"Doe\"}"
```

    - response

```
{"message":"Konnichiwa, John Doe"}
```

REST

- Programming
  - Multiple Consumes – request xml
    - request

```
curl -X POST -H "Content-Type: application/xml" http://localhost:8080/myresource/konnichiwa -d "<account><name>John</name><surname>Doe</surname></account>"
```

    - response

```
{"message":"Konnichiwa, John Doe"}
```

REST

- Programming
  - Multiple Consumes – Account.java (I)

```
package com.test;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="account")
public class Account {
        private String name;
        private String surname;
        public Account() { super(); }
        public String getName() {
                return name;
        }
```

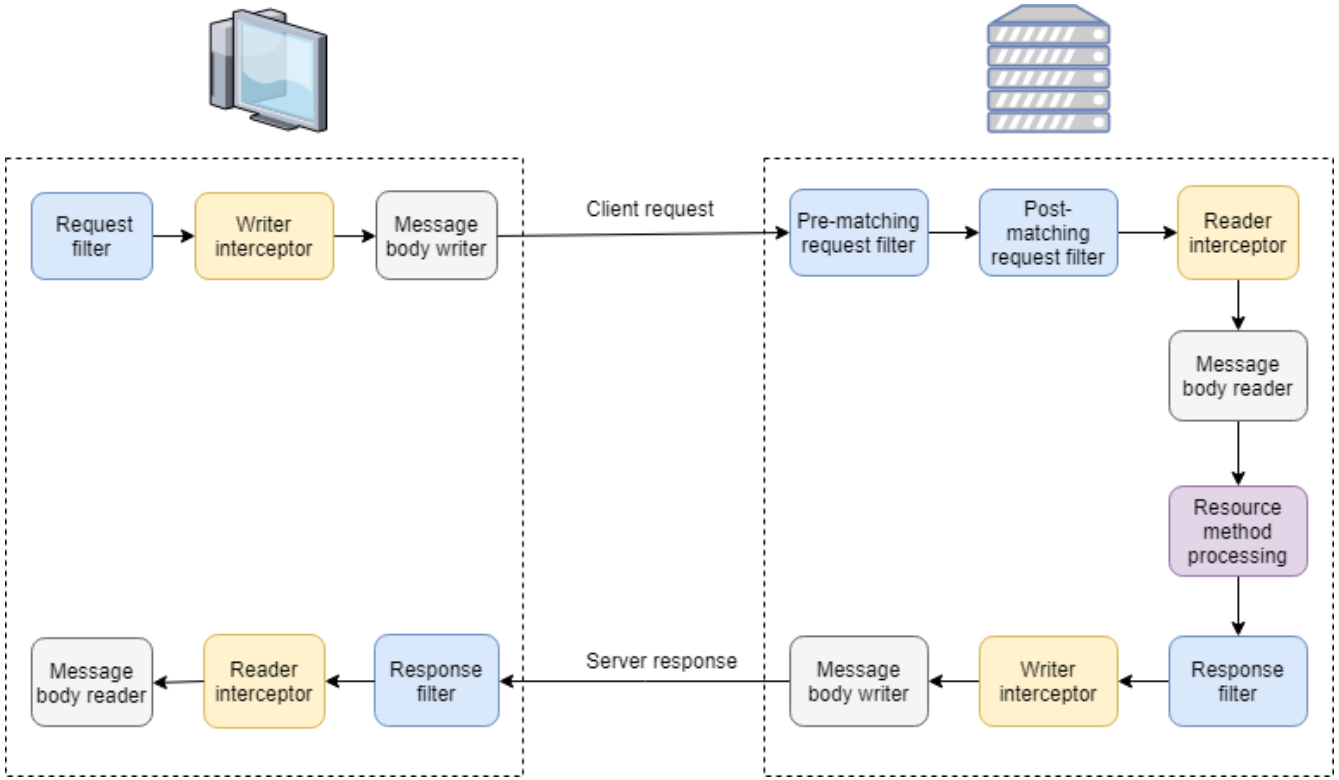
REST

- Programming
  - Multiple Consumes – Account.java (II)

```
        @XmlElement
        public void setName(String name) {
                this.name = name;
        }
        public String getSurname() { return surname;}
        @XmlElement
        public void setSurname(String surname) {
        this.surname = surname;
        }
        public String getFullName() {
                return name+" "+surname;
        }
        public String toString() {
                return super.toString()+"{name="+name+",
                surname="+surname+"}";
        }
}
```

REST

- Programming
  - Execution Order



Reference

- https://www.vogella.com/tutorials/REST/article.html
- https://restfulapi.net/
- https://www.redhat.com/en/topics/api/what-is-a-rest-api
- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://stackoverflow.com/questions/1568834/whats-the-difference-between-rest-restful
- https://eclipse-ee4j.github.io/jersey/
- https://eclipse-ee4j.github.io/jersey/download.html

Q & A