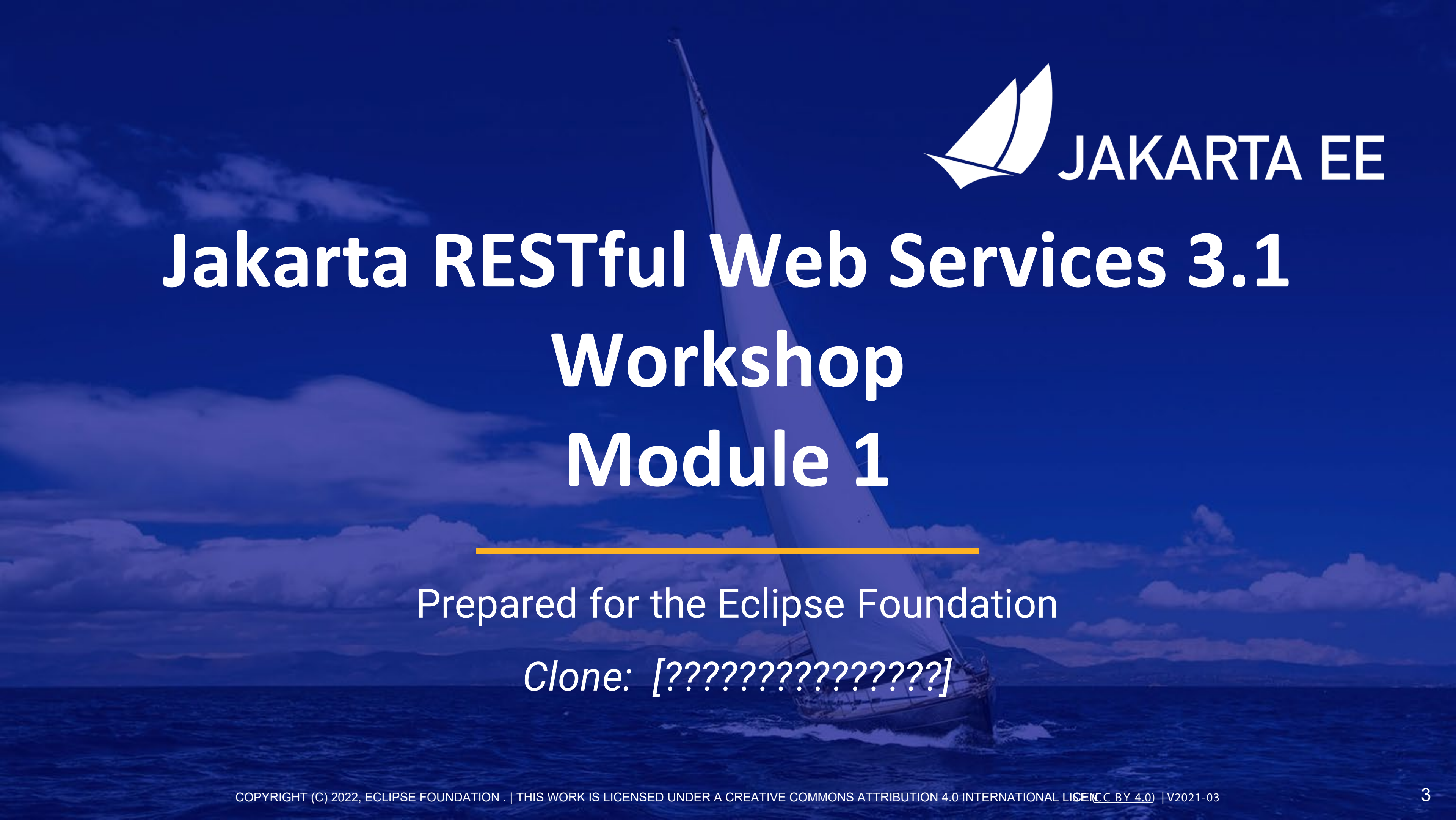
JAKARTA® EE

Jakarta RESTful Web Services 3.1 Workshop

Workshop Modules

- **Module 1**
  - Set up the environment
- **Module 2**
  - Coding the task that will become a service.
- **Module 3**
  - SeBootstrap web service implementation
  - GET
- **Module 4**
  - Server based web service implementation
  - GET
- **Module 5**
  - Java SE desktop client
  - GET
- **Module 6**
  - Servlet and Jakarta Faces application server client
  - POST
- **Module 7**
  - MultiPart File Upload
- **Module 8**
  - JPA service

JAKARTA EE

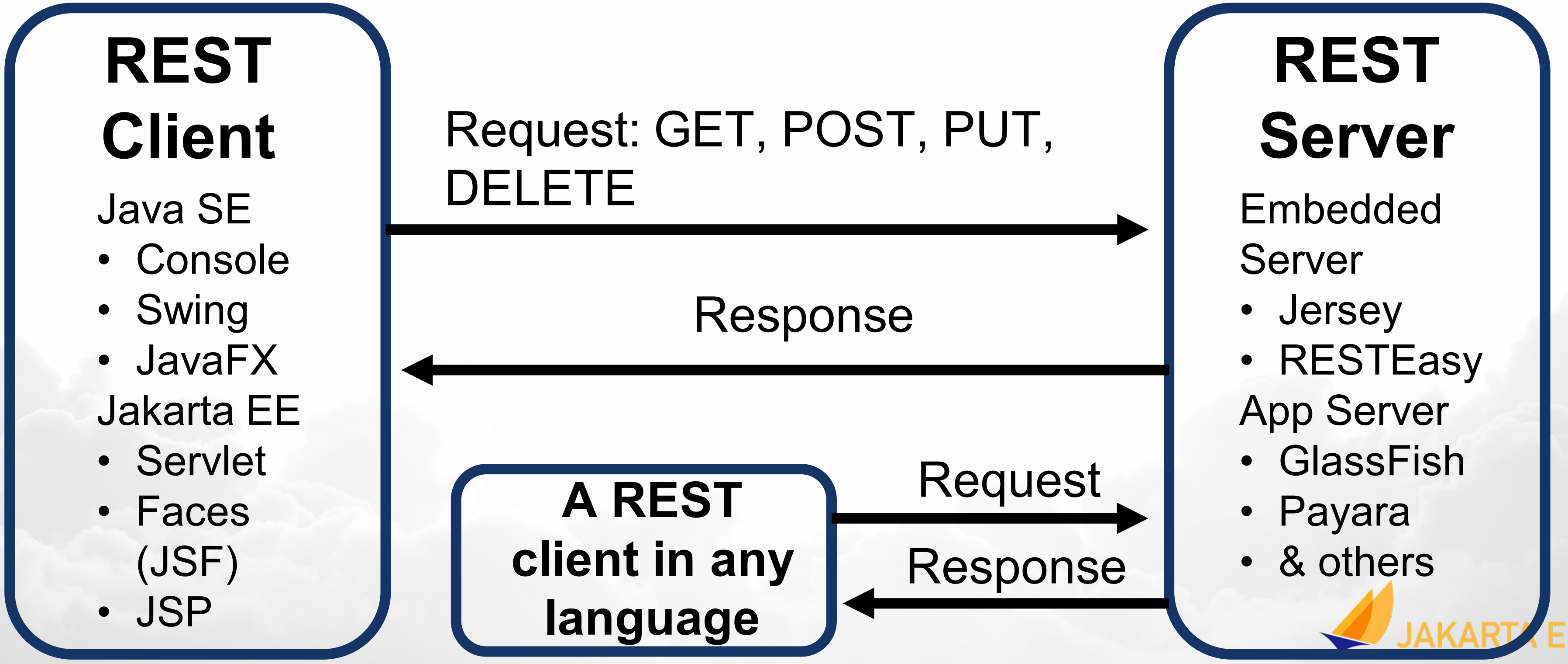


Jakarta RESTful Web Services 3.1 Workshop Module 1

Prepared for the Eclipse Foundation

Clone: [????????????????]

How does a Jakarta.ws.rs work?



The Tool Box

- Java SE 17
  - Jakarta versions are tied to a Java LTS version.
  - Jakarta 10 supports Java 17
- Build tool
  - Maven
  - Use the most recent version
- IDE
  - Not required but convenient
  - Must support Jakarta EE coding

- Application Server
  - Any Jakarta 10 compliant server can work
  - Workshop uses GlassFish 7
- Database
  - Any DB with a JDBC driver can be used
  - Workshop uses Derby that is included with GlassFish
- Basic Service Testing Tool
  - cURL for CLI testing of services

JAKARTA EE

Maven

- All the projects in this workshop use Maven
- Ensure that the Maven command line tool, `mvn`, is on your path
- The pom files include a `<defaultGoal>`
- This means that to build and sometimes run a project all you need do is open a terminal or console in a project's root folder and enter `mvn`
  - No Maven switches are required
- In some modules you will need to deploy the code to GlassFish

Important

- Read the participant documents, these will tell you what is expected from you
- Read the source code, pom files, and other XML files
    - All are commented and contain additional information

Jakarta RESTful Web Services 3.1 Workshop Module 2

Coding the task that will become a service.

Your Task

- The workshop requires a task that can be used as a RESTful web service
- You are free to create any task
- Avoid complexity and create a task in a POJO and, if needed, one DTO, typically coded as a JavaBean
- There is no need for your task to have a UI as the methods will be called by the RESTful web services

The Task - Compound Interest Calculation

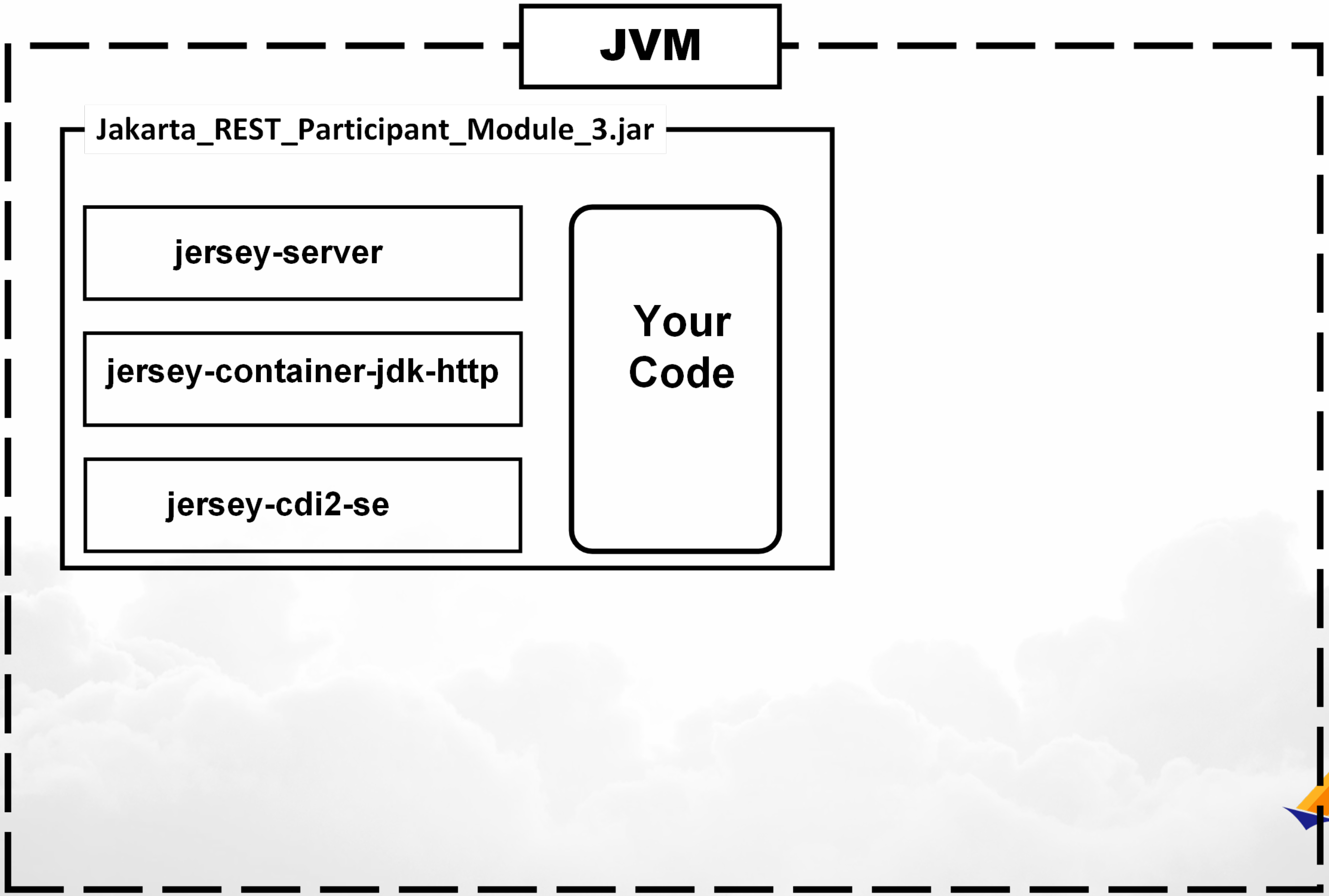
- If you would like to use our task look at `mod_02_compoundinterest_participant`
- This project contains the class CompoundInterest.java with two methods to complete
    - `public void calculateCompoundInterest(CompoundBean compoundBean)`
    - `private boolean validateBean(CompoundBean compoundBean)`
- There is also a JUnit5 parameterized test class that is complete:
    - `ParameterizedTests.java`
- If you use our task, then ensure that it can pass the unit tests

JAKARTA EE

Jakarta RESTful Web Services 3.1 Workshop Module 3

Java SE standalone RESTful web service using the Jakarta EE 10 SeBootstrap class

Module 3: SeBootstrap web service implementation.


JVM
Jakarta_REST_Participant_Module_3.jar
jersey-server
jersey-container-jdk-http
jersey-cdi2-se
Your Code

The task

- Open the project `mod_03_restsebootsrap_participant`
- This project implements a simple Greeting service
- Review the files and run the project
- Implement your own RESTful task or your completed CompoundInterest RESTful task

Se Bootstrap web service implementation

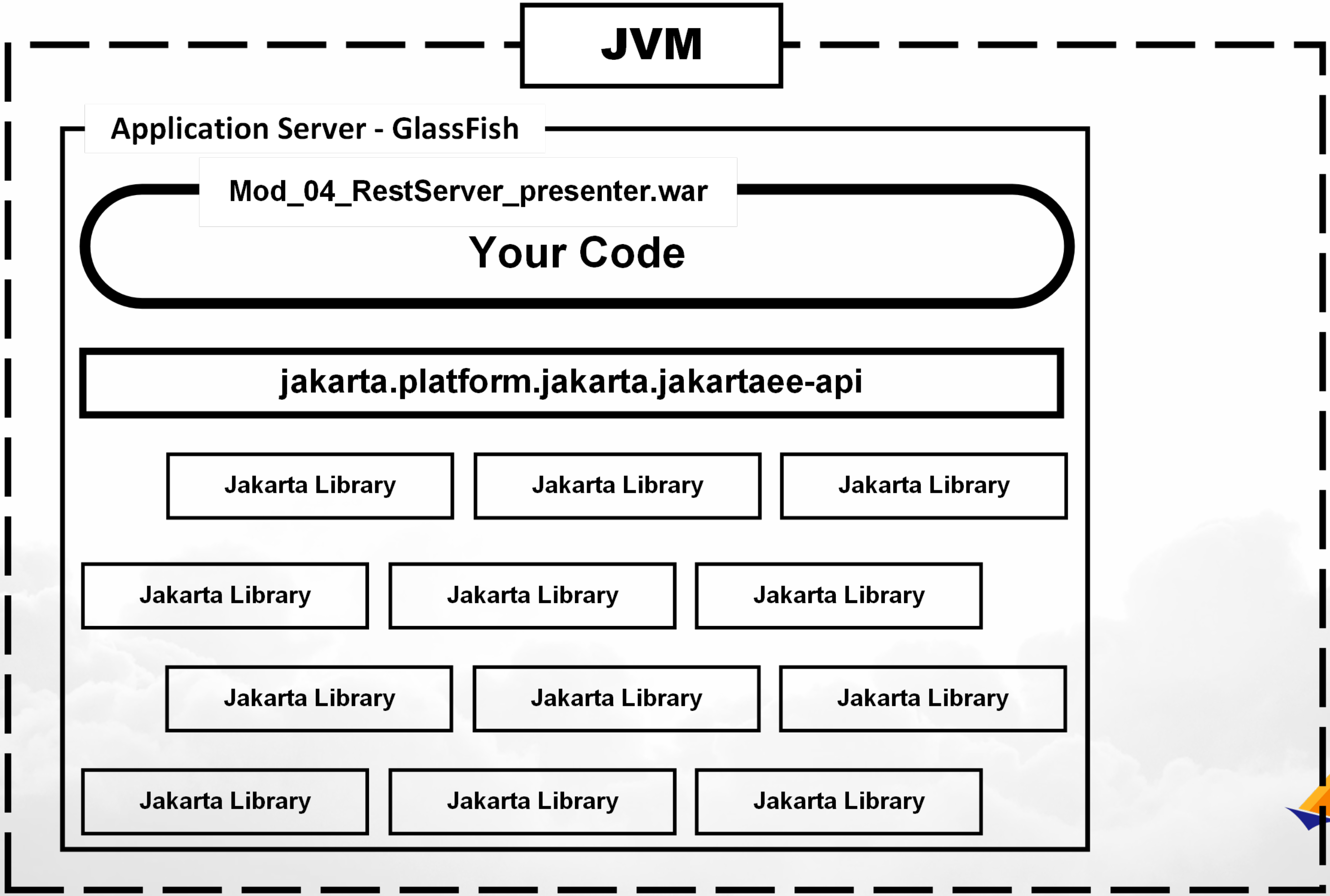
- Web services can be standalone applications
- An embedded server is required, such as Jersey or RESTEasy
- There are three classes required:
    - The task class with annotations that define the code as a service
    - A class that extends Application and overrides the method getClasses that will return all service classes in the project
    - A class that configures the server to listen to a port for requests to the service

JAKARTA EE

Jakarta RESTful Web Services 3.1 Workshop Module 4

Server based web service implementation

Module 4: Server based web service implementation.



Server based web service implementation

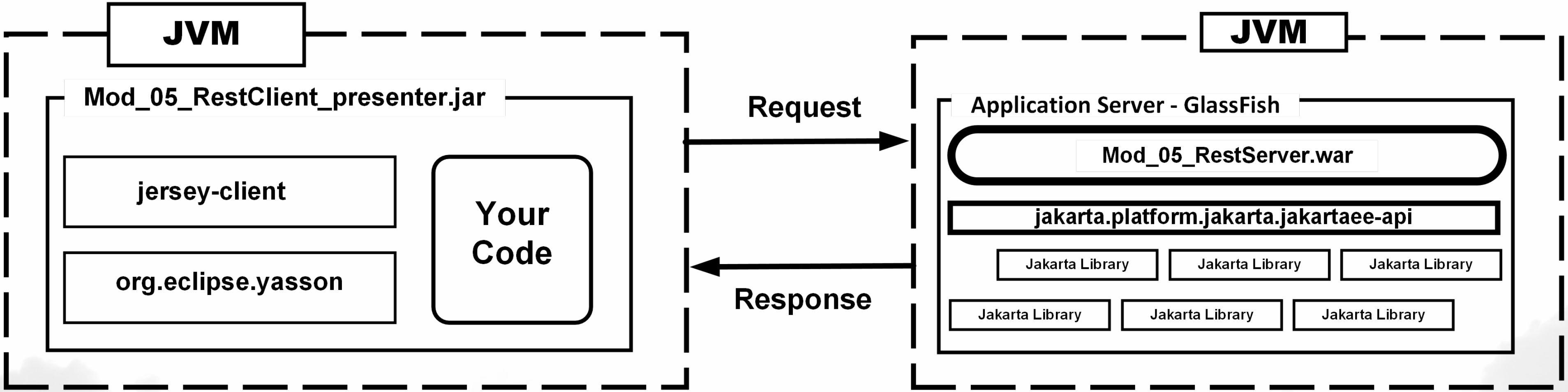
- Nothing could be simpler than when using an Application Server to host a service
- Every possible library/framework that you might want to use is provided by the server
- The result is that such a projects needs only one dependency

```
<dependencies>
    <dependency>
        <groupId>jakarta.platform</groupId>
        <artifactId>jakarta.jakartaee-api</artifactId>
        <version>${jakartaee-api.version}</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Jakarta RESTful Web Services 3.1 Workshop Module 5

Java SE/Desktop web services client

Module 5: Java SE web services client



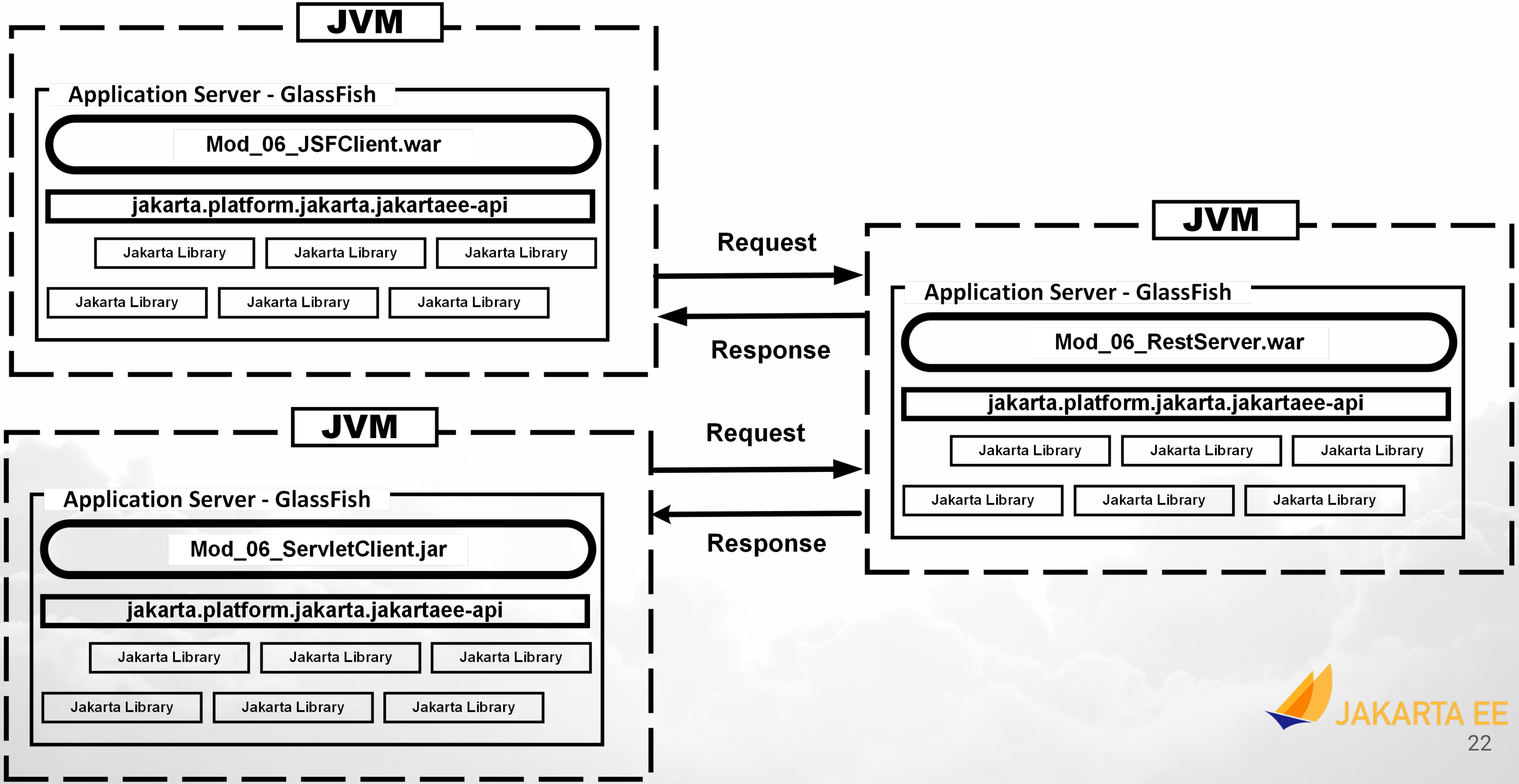
Java SE/Desktop web services client

- Time to look at CLI web service clients
- Fewer Maven dependencies
- De-serialize a JSON string into an Object
- We will see that client code is near identical in both a desktop and server-based projects

Jakarta RESTful Web Services 3.1 Workshop Module 6

Servlet and Jakarta Faces application server client

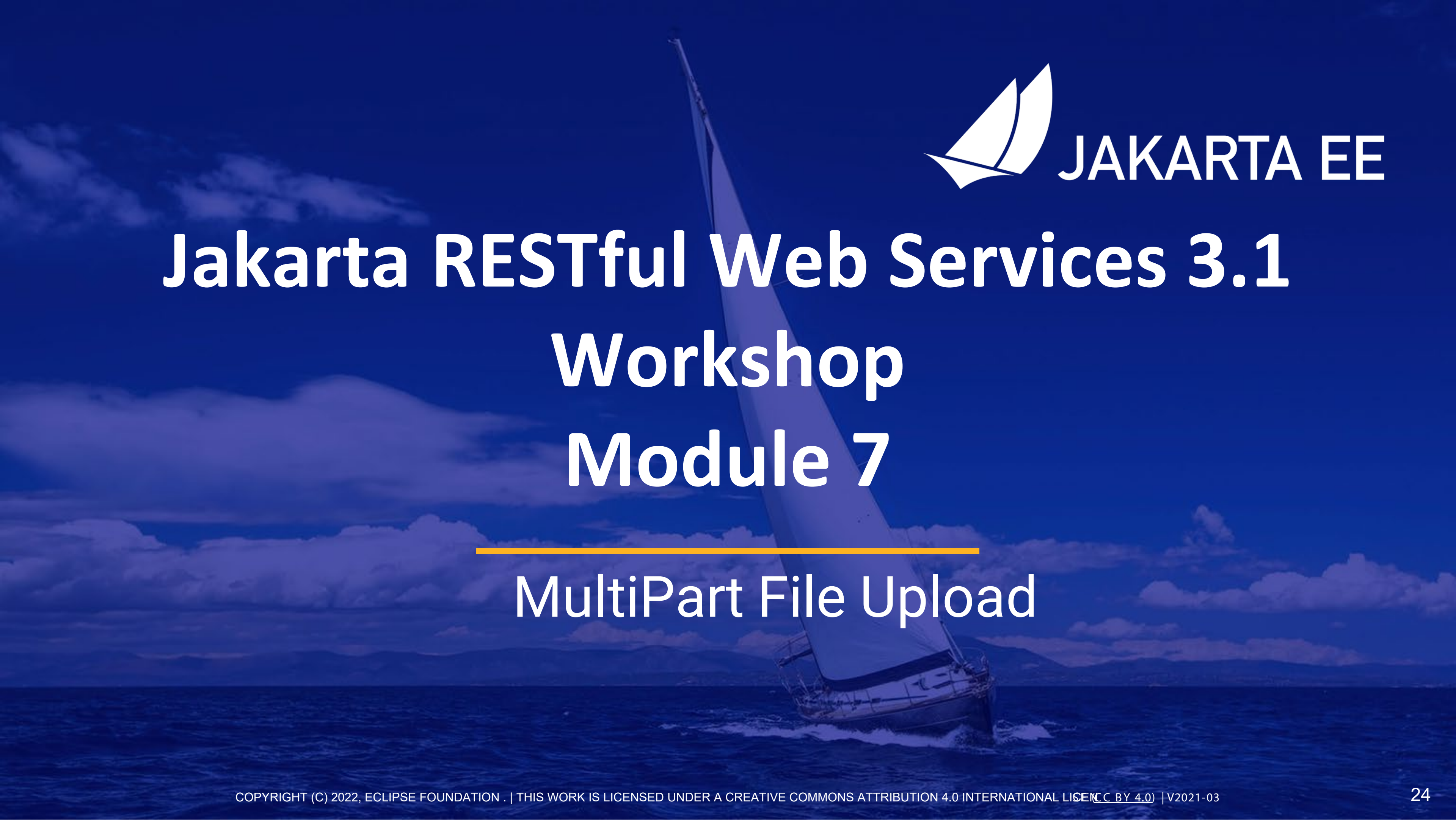
Module 6: Servlet and Jakarta Faces RESTful Clients



JAKARTA EE

GlassFish server web services client

- There are two clients in this module
  - Jakarta Faces
  - Servlet
- The client code to access the service are near identical
- The Jakarta Faces uses a POJO managed by CDI with the client code
- The Servlet embeds the client code method in the Servlet

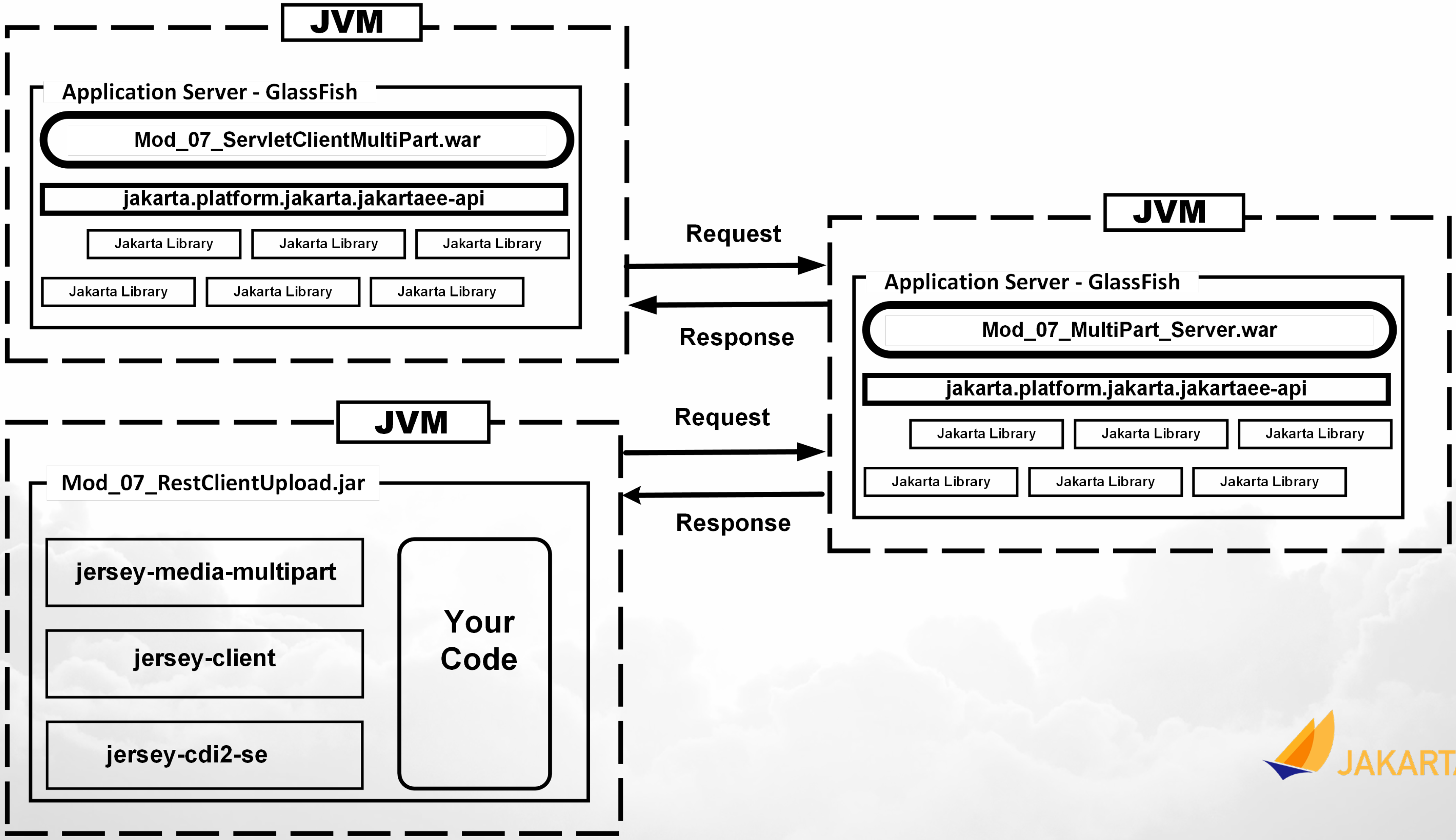


JAKARTA EE

Jakarta RESTful Web Services 3.1 Workshop Module 7

MultiPart File Upload

Module 7: File transfer with Multipart



MultiPart File Upload

- MultiPart allows you to upload or download binary files
- In this module you will see a
  - `mod_07_multipart_server_participant`
  - `mod_07_restclientupload_participant`
  - `mod_07_servletclientmultipart_participant`
- The multipart_server will receive an uploaded file and store it on your disk
- The restclientupload is a desktop client that uploads a file
- The servletclientmultipart is a Servlet client that uploads a file

JAKARTA EE

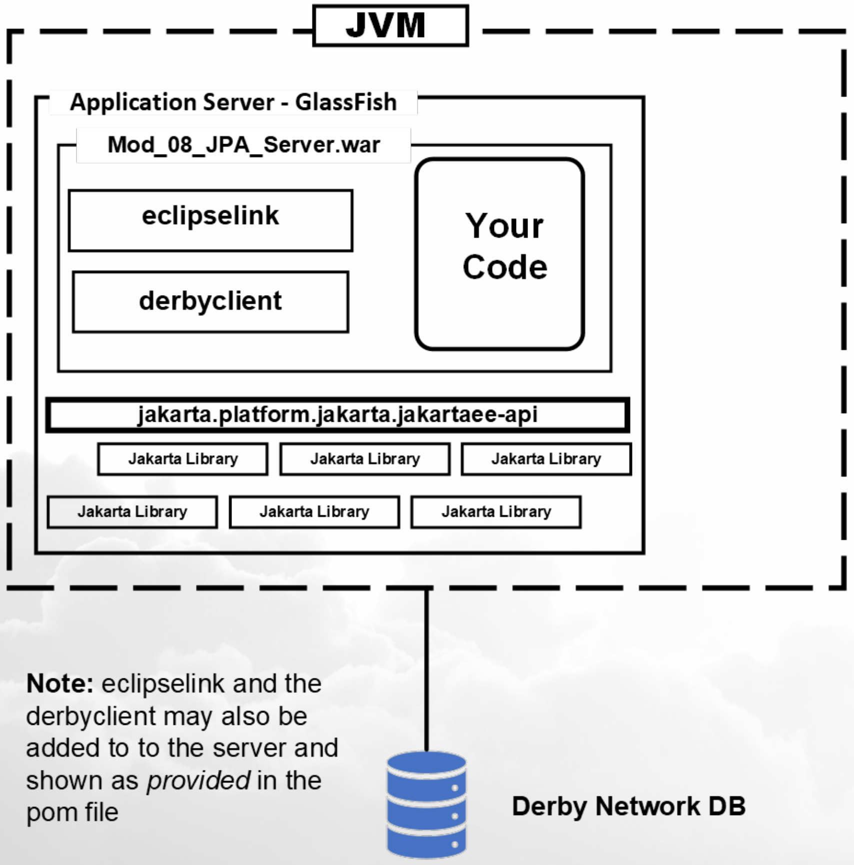
JAKARTA EE

Jakarta RESTful Web Services 3.1 Workshop Module 8

Jakarta Persistence 3.1 & RESTful Web Services 3.1

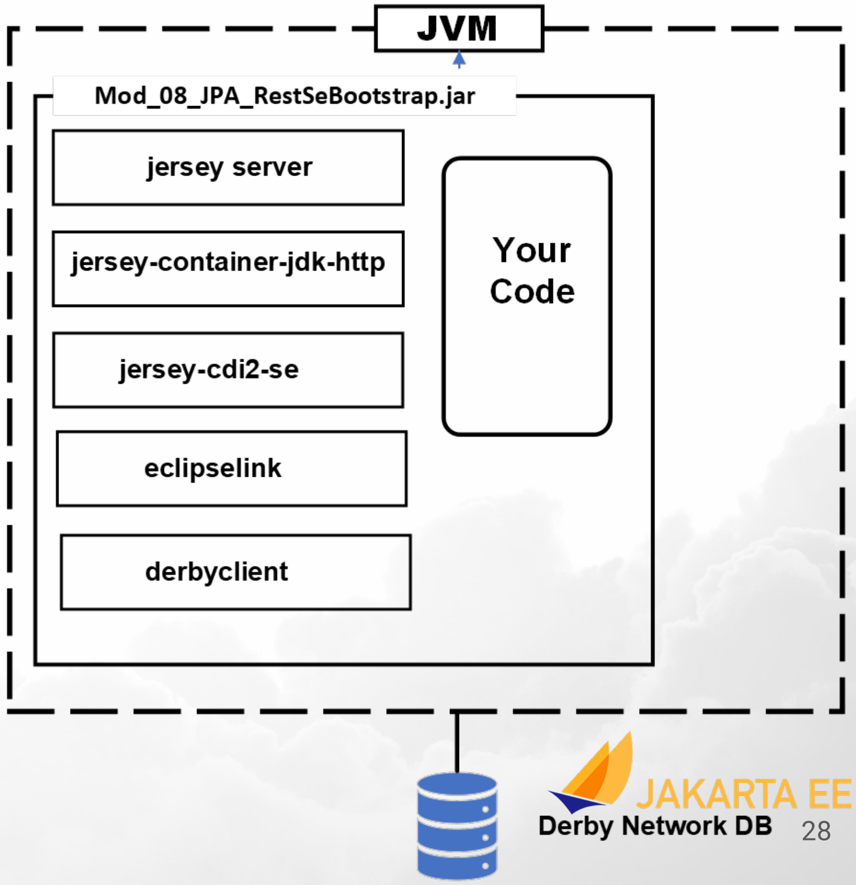
Module 8: Jakarta Persistence 3.1 & RESTful Web Services 3.1

Application Server



Note: eclipselink and the derbyclient may also be added to to the server and shown as *provided* in the pom file

Java SE



Jakarta Persistence 3.1 & RESTful Web Services 3.1

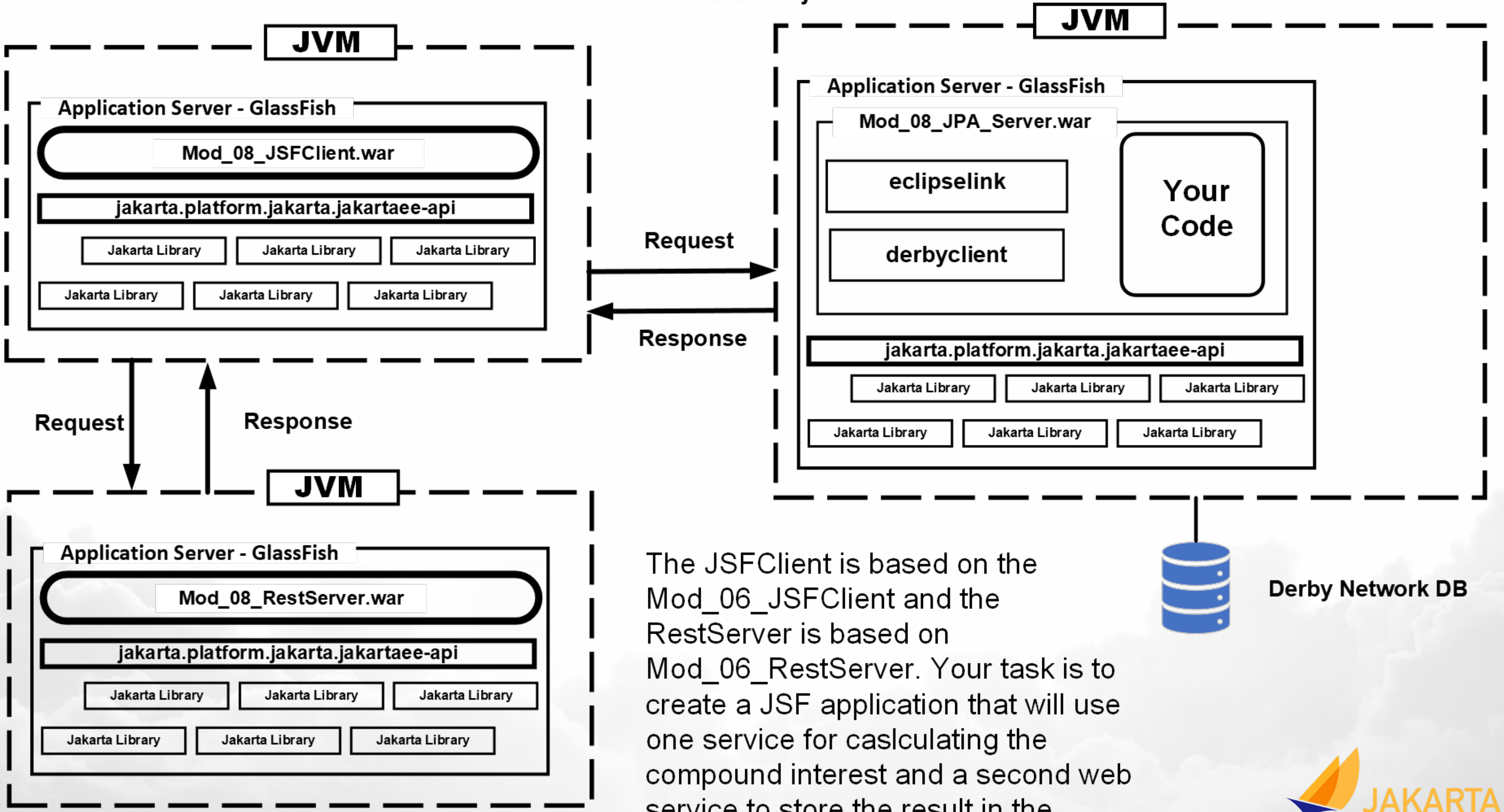
- This module uses Jakarta Persistence to store a record to a database
- The record consists of the compound interest data with a primary key
- The database will be Derby
- There are two servers:
  - Java SE desktop web service
  - `mod_08_jpa_restsebootstrap_participant`
  - GlassFish hosted web service
  - `mod_08_jpa_server_participant`

JAKARTA EE

Jakarta RESTful Web Services 3.1 Workshop Module 8 Project

Jakarta Persistence 3.1 & RESTful Web Services 3.1

Module 8: Project



The JSFClient is based on the Mod_06_JSFClient and the RestServer is based on Mod_06_RestServer. Your task is to create a JSF application that will use one service for caslculating the compound interest and a second web service to store the result in the database.

JAKARTA EE

THANK YOU!

JAKARTA EE