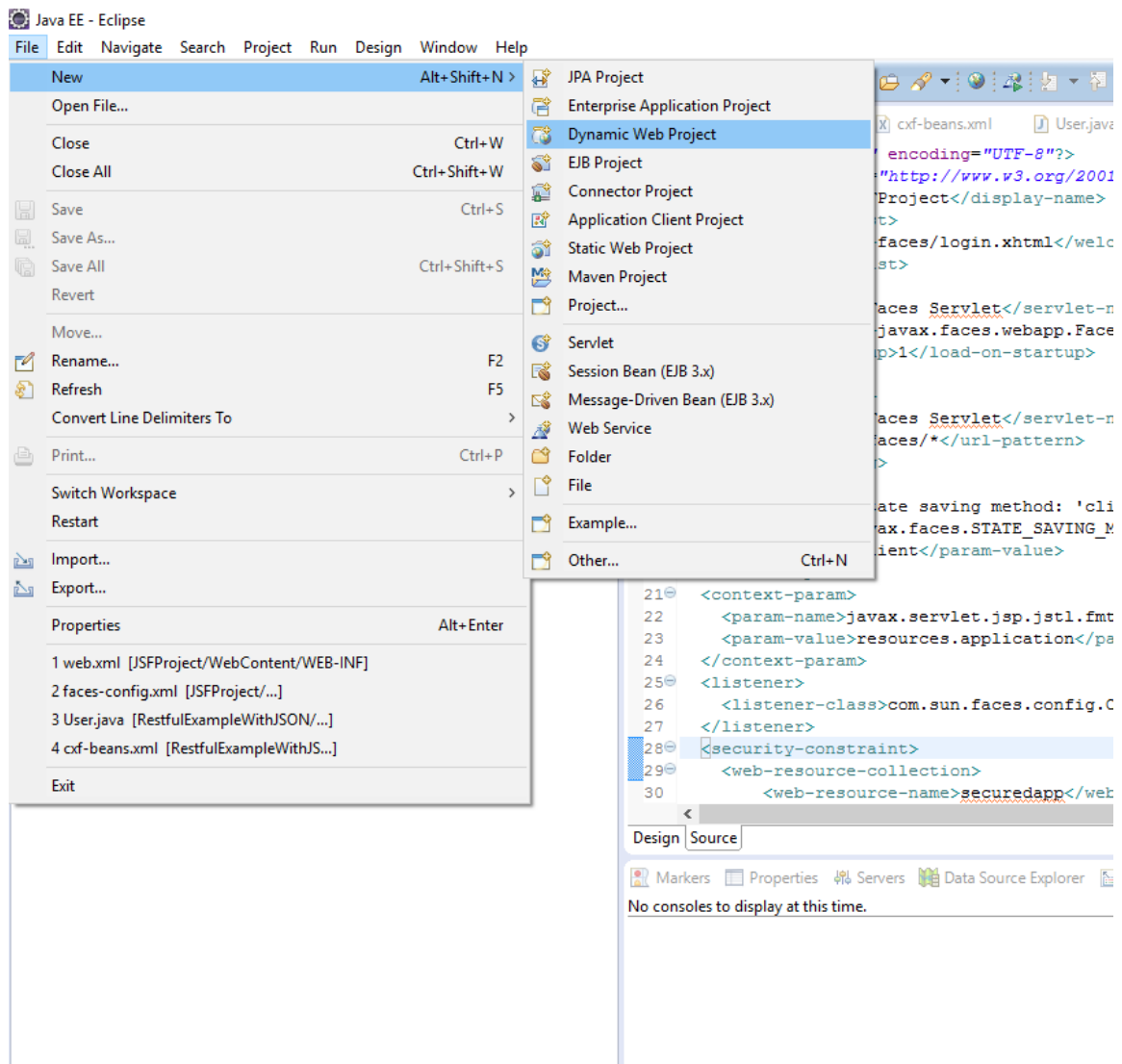
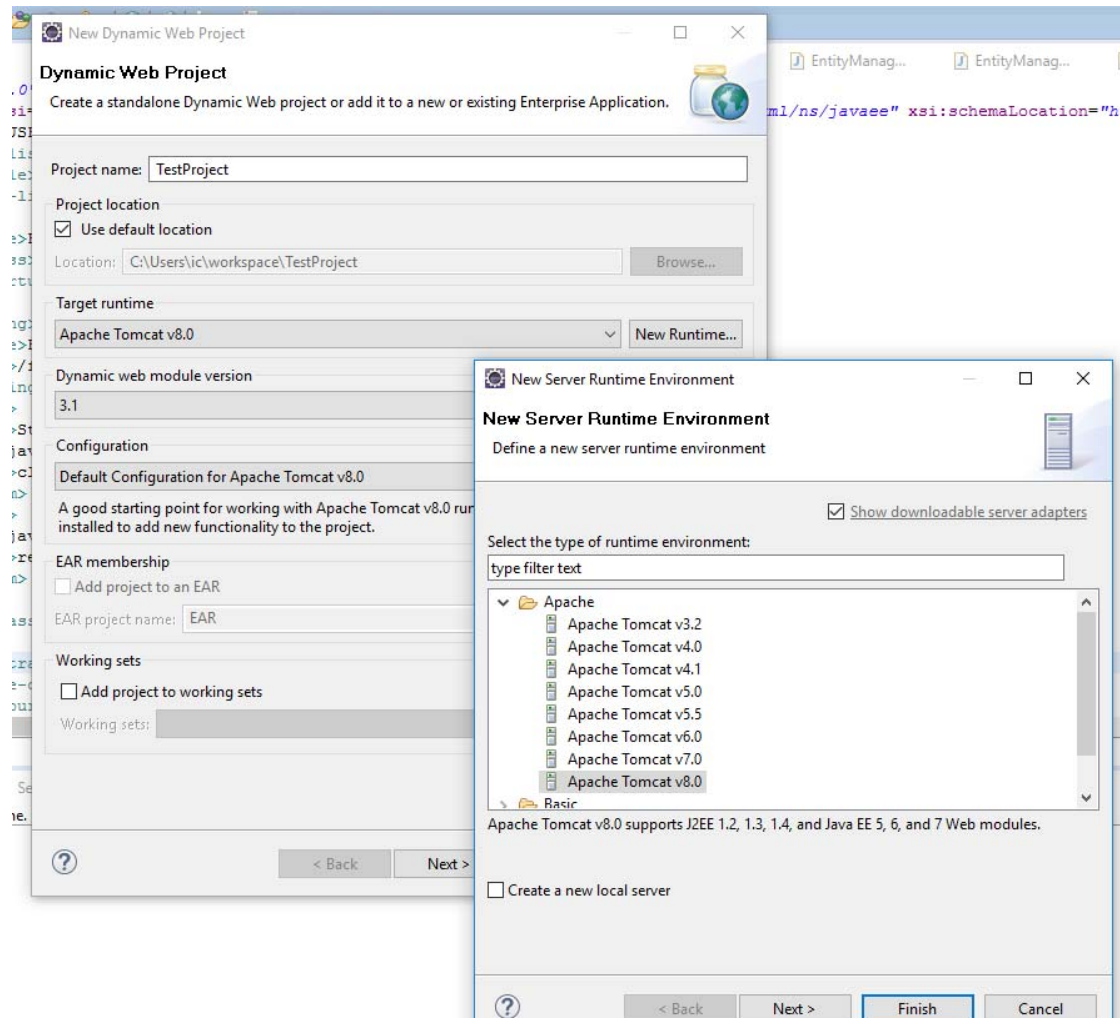


Οδηγίες ρύθμισης στο περιβάλλον Eclipse

1. Δημιουργία νέου Project:



2. Ρύθμιση Server Runtime:



New Server Runtime Environment

Tomcat Server

Specify the installation directory

Name:
Apache Tomcat v8.0 (3)

Tomcat installation directory:
C:\Users\ic\tomcat

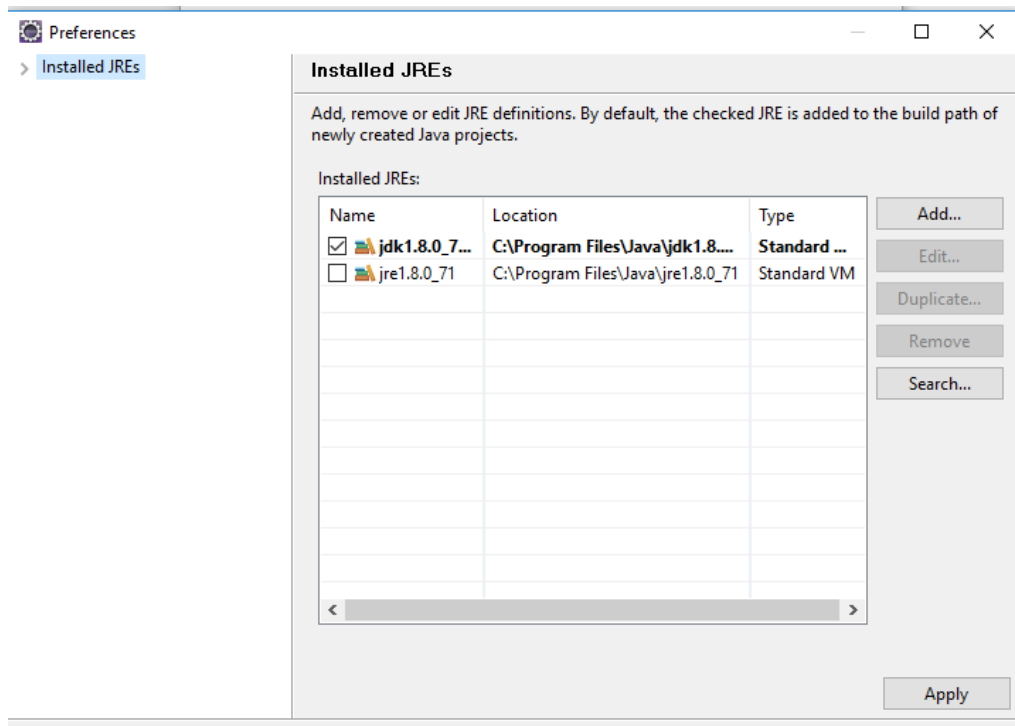
JRE:
Workbench default JRE

Buttons: Browse... Download and Install... Installed JREs...

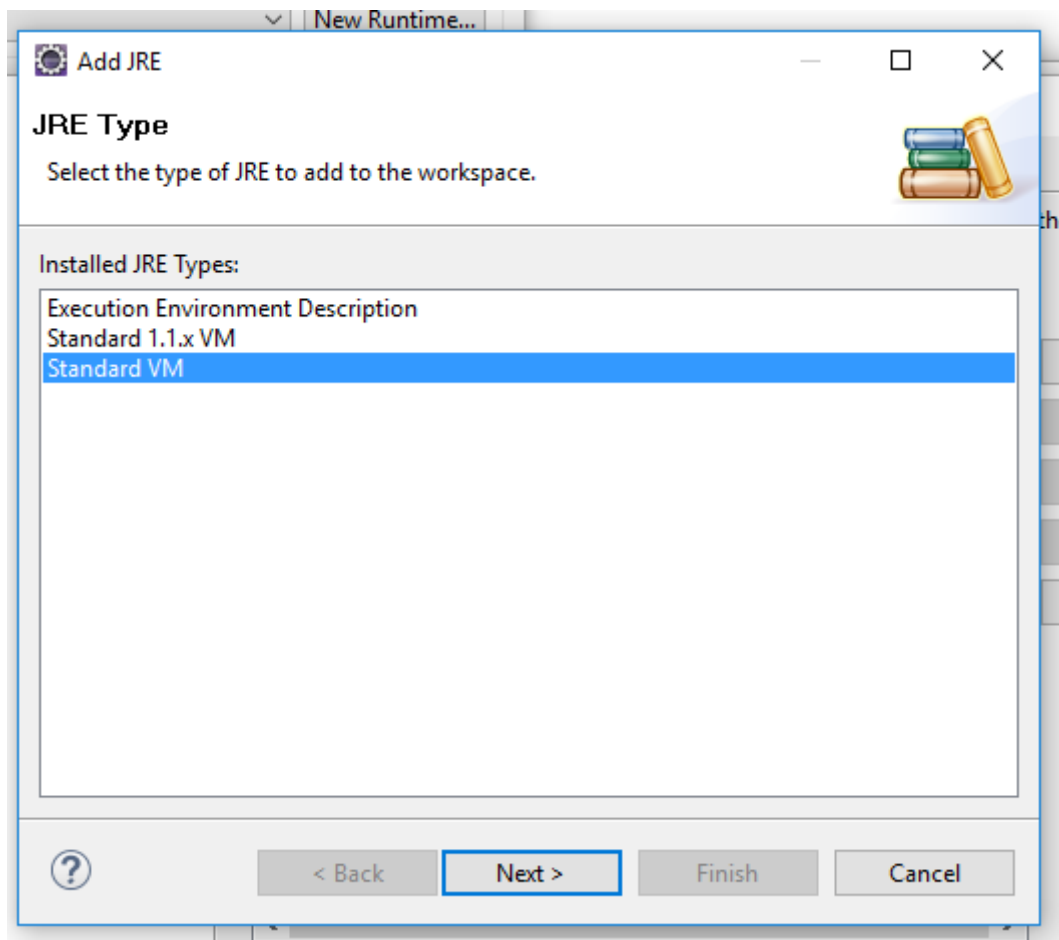
Navigation: ? < Back Next > Finish Cancel

Θα πρέπει να εισαχθεί ο κατάλογος όπου έχει αποσυμπεσθεί ο Tomcat (εν προκειμένω C:\Users\ic\tomcat)


3. Προαιρετικά ρυθμίζεται το Java Runtime Environment που θα χρησιμοποιηθεί:



Σε περίπτωση που το επιθυμητό JRE δεν εμφανίζεται στη λίστα, με το κουμπί Add μπορούμε να το προσθέσουμε.




Επιλέγεται Standard VM και το κουμπί Next. Στο επόμενο παράθυρο που εμφανίζεται εισάγουμε στο JRE home τον κατάλογο όπου είναι εγκατεστημένο το επιθυμητό JDK/JRE. Ο κατάλογος των βιβλιοθηκών συστήματος «γεμίζει» αυτόματα.

 Add JRE

JRE Definition

Specify attributes for a JRE



JRE home:

C:\Program Files\Java\jdk1.8.0_71

Directory...

JRE name:


jdk1.8.0_71

Default VM arguments:

Variables...


JRE system libraries:

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\resources.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\rt.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\jsse.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\jce.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\charsets.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\jfr.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\ext\access-bri


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\ext\clldrdata.ja


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\ext\dnsns.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\ext\jaccess.jar


>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\ext\jfxrt.jar

>

>

 C:\Program Files\Java\jdk1.8.0_71\jre\lib\ext\localedata

>

<

>

Add External JARs...

Javadoc Location...

Source Attachment...


External annotations...

Remove

Up

Down

Restore Default



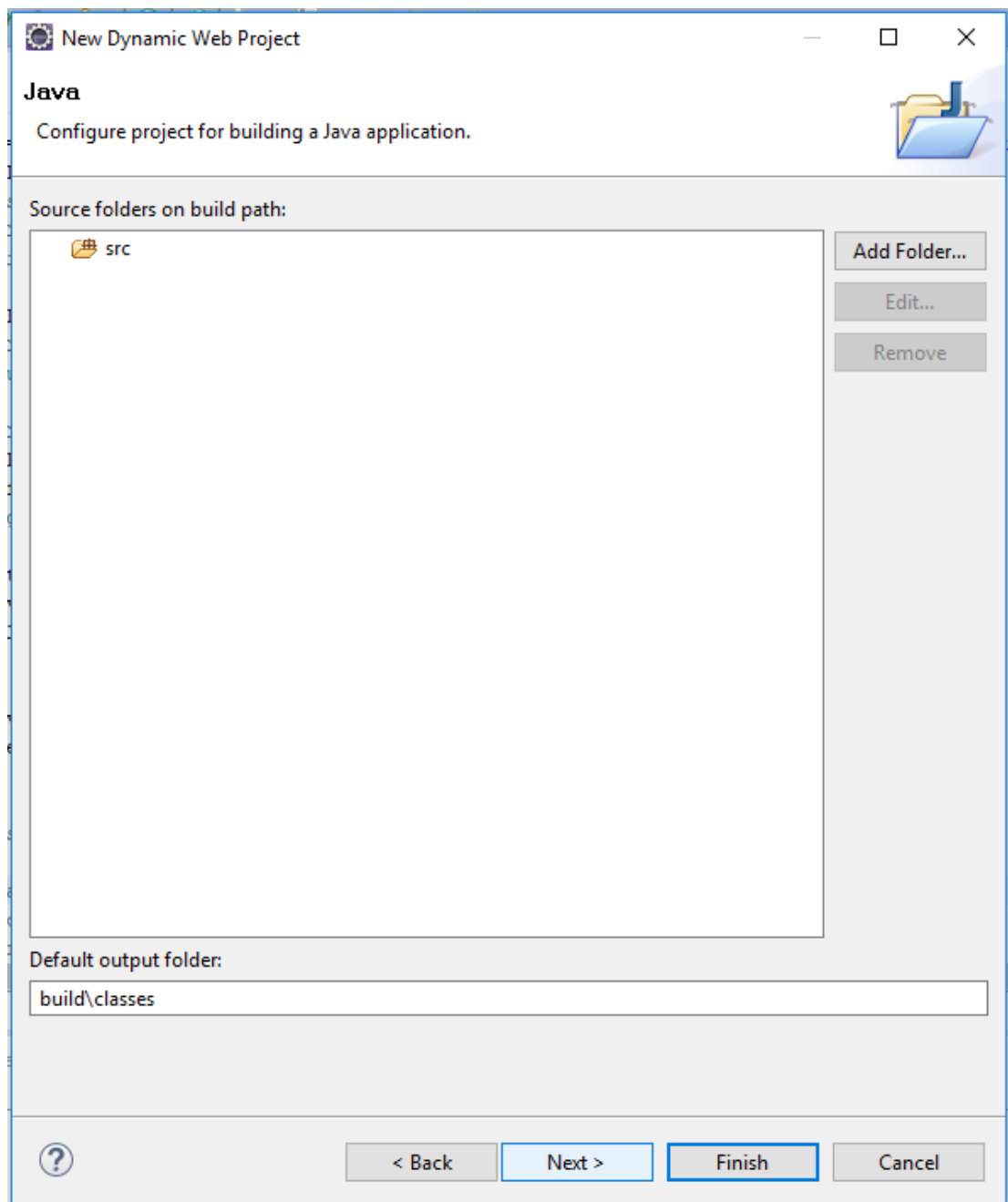
< Back

Next >

Finish

Cancel

4. Αφού επιλεγθεί το Target Runtime και το όνομα του Project (βλ. Βήμα 2) πατάμε το κουμπί Next.



Δεν είναι απαραίτητο να αλλάξουμε κάτι σε αυτό το σημείο, οπότε επιλέγουμε πάλι Next

New Dynamic Web Project

Web Module

Configure web module settings.

Context root: TestProject

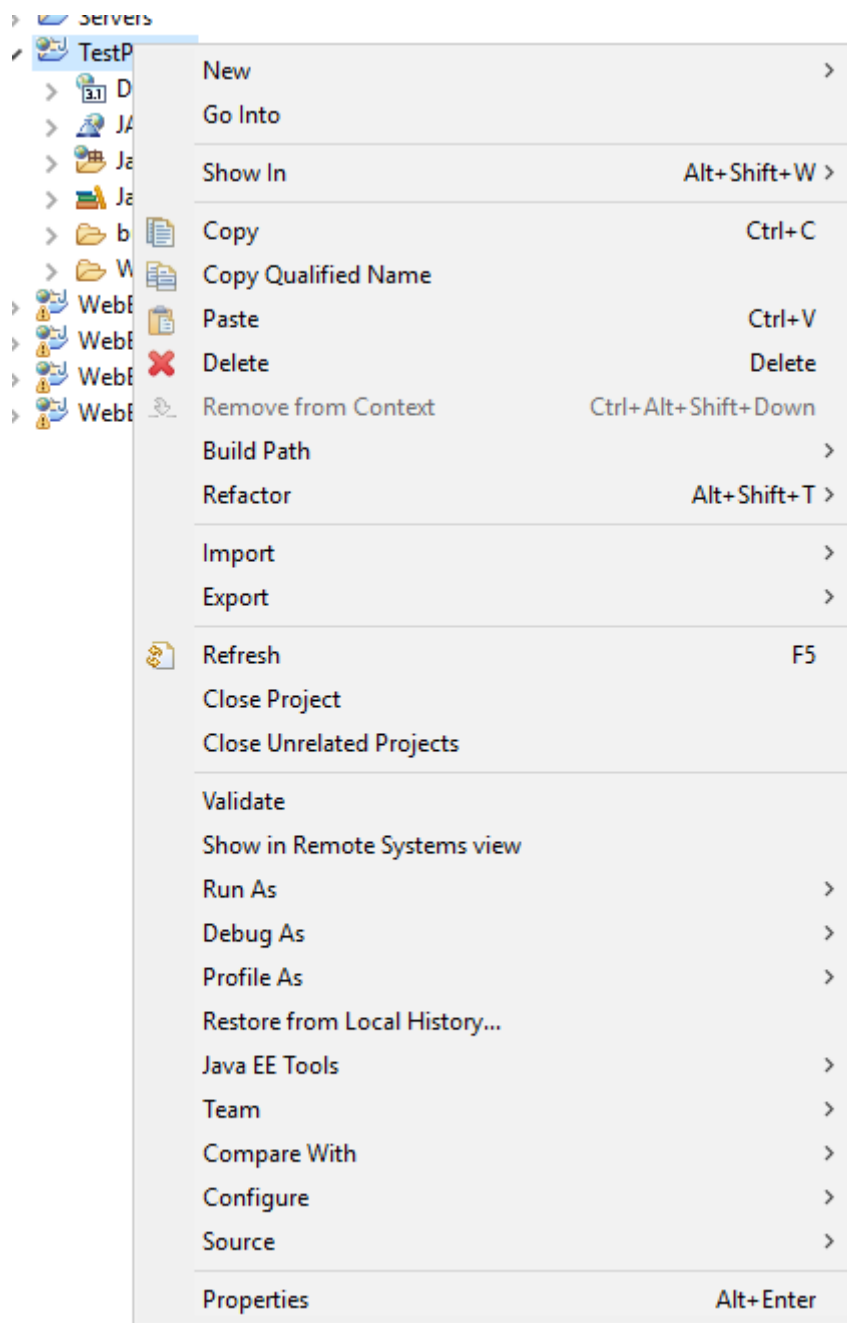
Content directory: WebContent

☒ Generate web.xml deployment descriptor

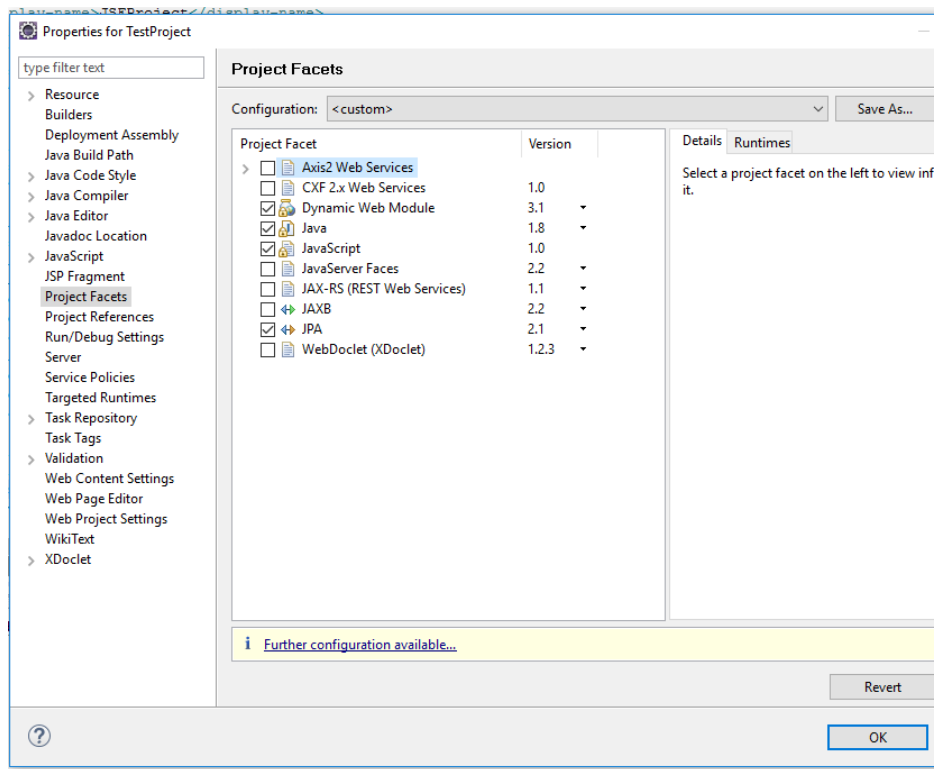
? < Back Next > Finish Cancel

Για να έχουμε έλεγχο στις ρυθμίσεις της εφαρμογής Web τσεκάρουμε την επιλογή: Generate web.xml. Η ρύθμιση του Context root αντιστοιχεί στο μονοπάτι-ρίζα εντός του Web Server όπου θα δέχεται αιτήσεις η εφαρμογή μας. Αν υποθέσουμε ότι ο Tomcat τρέχει στη διεύθυνση <http://localhost:8080>, το URL θα είναι <http://localhost:8080/TestProject>

5. Ρύθμιση Project για χρήση βιβλιοθήκης JPA



Πατάμε δεξί κουμπί πάνω στο TestProject που εμφανίζεται στον Project Explorer και επιλέγουμε με το ποντίκι την επιλογή Properties.

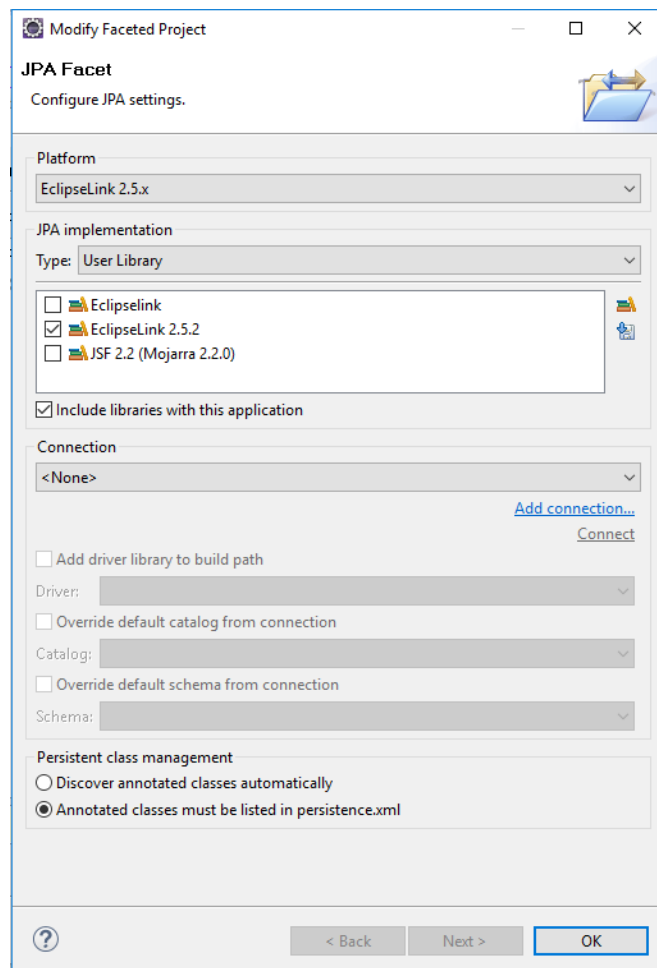


Επιλέγουμε Project Facets και στη συνέχεια τσεκάρουμε την επιλογή JPA.

Εμφανίζεται ο σύνδεσμος Further Configuration Available τον οποίο και επιλέγουμε.

Σε μια νέα εγκατάσταση του Eclipse δεν θα εμφανίζεται κάτι στη λίστα με τις υλοποιήσεις του JPA (JPA Implementation).

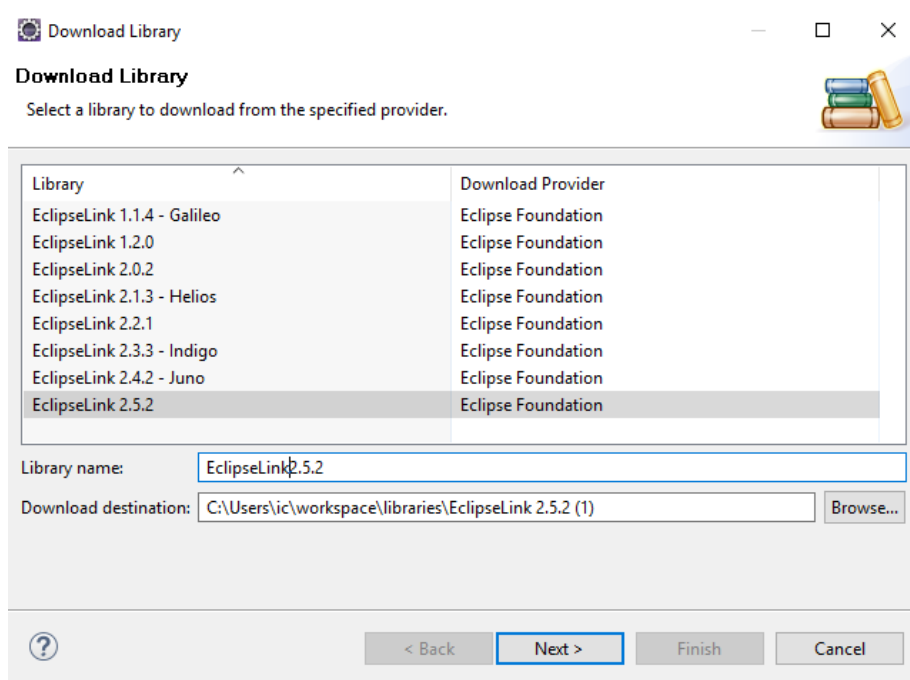
Κάνουμε κλικ με το mouse στο εικονίδιο που εμφανίζει τη δισκέτα και το βέλος (δείτε το επόμενο screenshot).



Επιλέγουμε κάποια από τις βιβλιοθήκες που εμφανίζονται (κατά προτίμηση το EclipseLink 2.5.2) το οποίο κατεβαίνει αυτομάτως από το Eclipse (δείτε το 2^ο screenshot στην επόμενη σελίδα) και πατάμε το κουμπί Next.

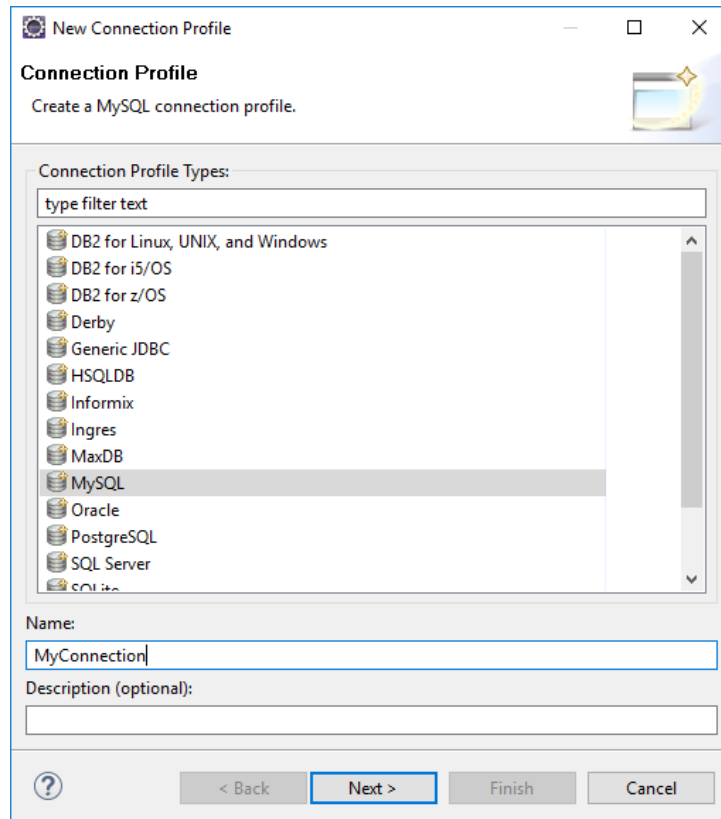
Αποδεχόμαστε την άδεια που εμφανίζεται σε επόμενο παράθυρο, και πατάμε το κουμπί Finish.

Στη συνέχεια επιλέγουμε τη βιβλιοθήκη που εγκαταστήσαμε, τσεκάροντας την αντίστοιχη επιλογή στη λίστα (EclipseLink 2.5.2).



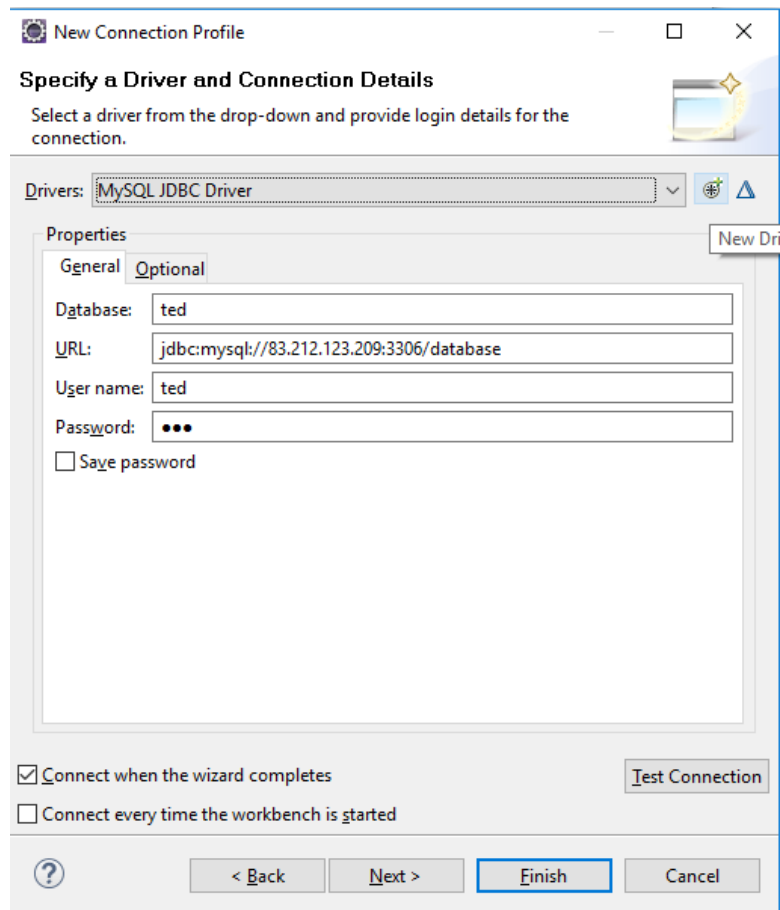
Το επόμενο βήμα αφορά τη δημιουργία της σύνδεσης (Connection) στη βάση. Επιλέγουμε με το mouse τον σύνδεσμο Add connection στο παράθυρο JPA Facet που είχε ανοίξει προηγουμένως.

Εμφανίζεται το παράθυρο Connection Profile όπου επιλέγουμε το είδος της βάσης στην οποία θα συνδεθούμε (έστω MySQL) και δίνουμε ένα όνομα στη σύνδεση (εν προκειμένω MyConnection) .



Εμφανίζεται ένα παράθυρο όπου πρέπει να ρυθμίσουμε τον Driver που θα χρησιμοποιήσουμε για τη σύνδεση στη βάση (δηλαδή το κατάλληλο jar (Java Archive)) καθώς τα στοιχεία σύνδεσης στην βάση (username, password, url).

Αν δεν υπάρχει εγκατεστημένος driver θα πρέπει να πατήσουμε το εικονίδιο που βρίσκεται ακριβώς δίπλα από το πλαίσιο εισόδου κειμένου (textbox) για τον driver (New Driver Definition).



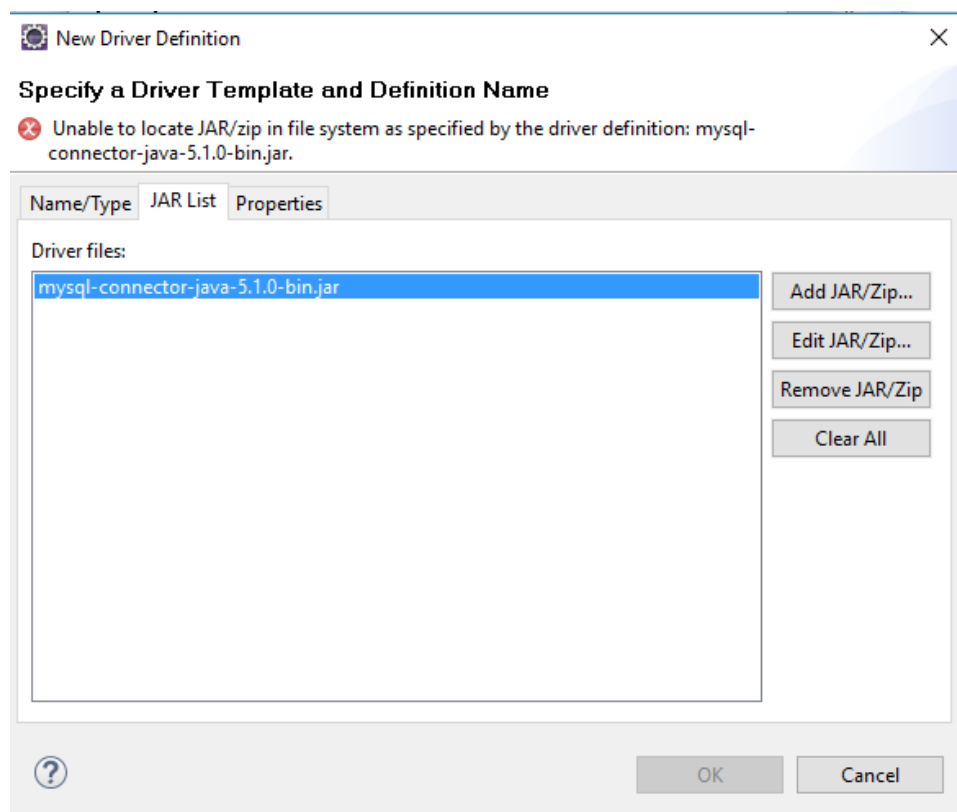
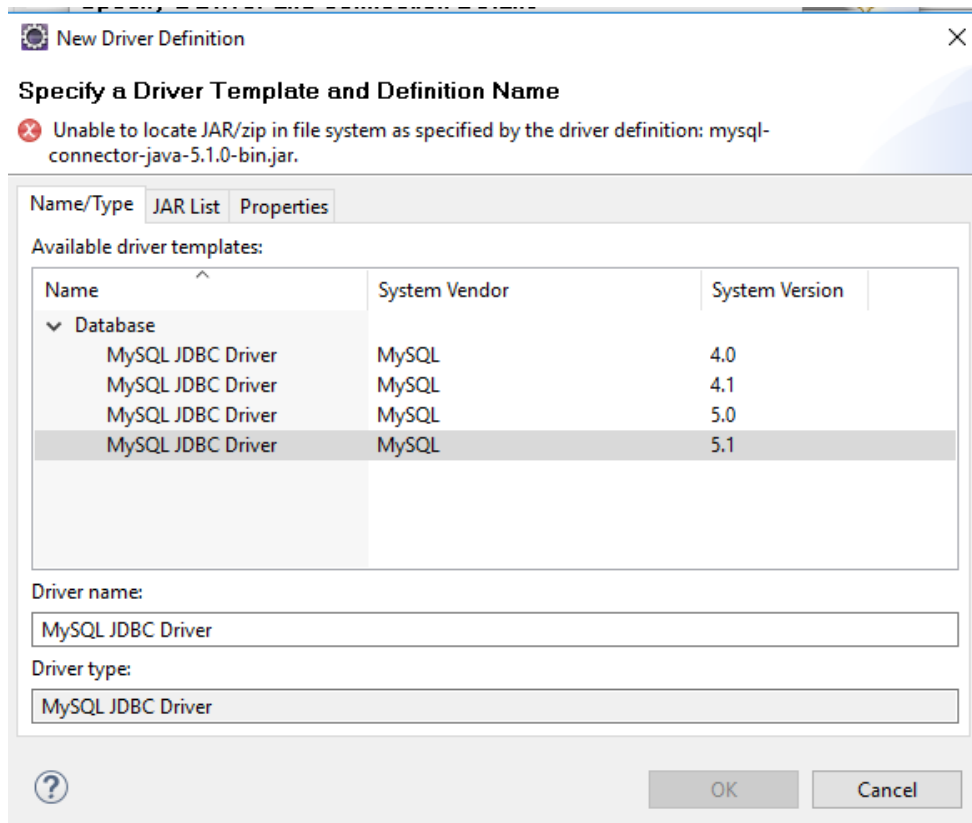
Εμφανίζεται παράθυρο όπου θα πρέπει να ρυθμίσουμε τον driver και να ορίσουμε που βρίσκεται το αντίστοιχο Java Archive. Θα πρέπει να έχουμε κατεβάσει τον driver από το Web και να τον έχουμε αποσυμπίσει σε κατάλογο που μπορεί το Eclipse να προσπελάσει.

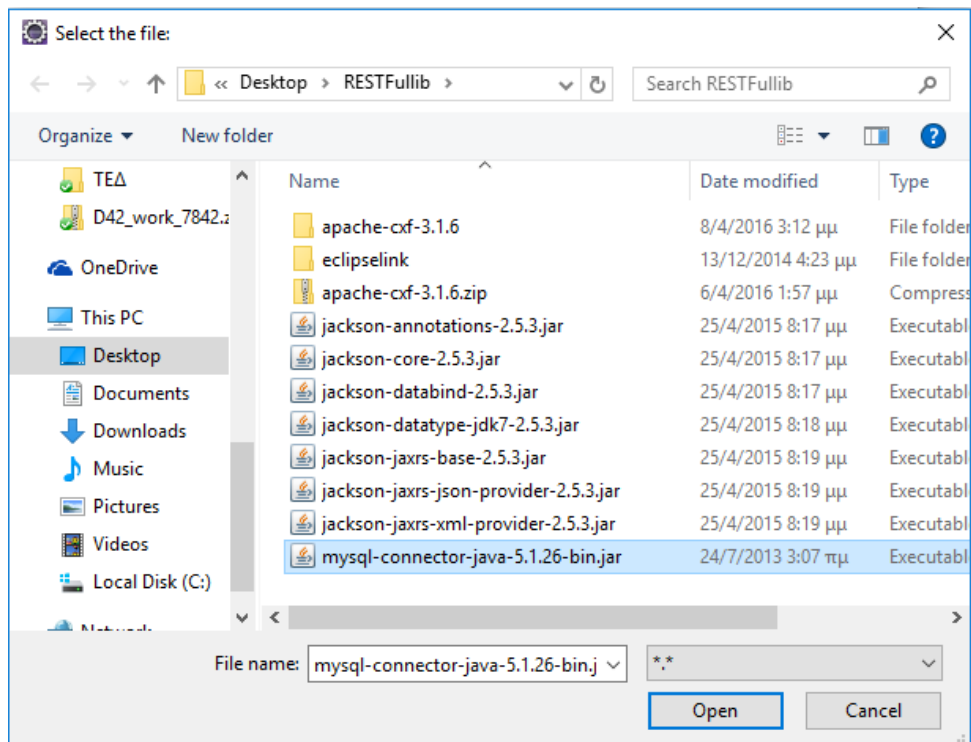
<https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.39.zip>

Επιλέγουμε το tab JAR List και στη συνέχεια πατάμε το κουμπί Edit JAR/Zip όπου ορίζουμε τη θέση του αρχείου jar στον δίσκο (βλ. επόμενα screenshots).

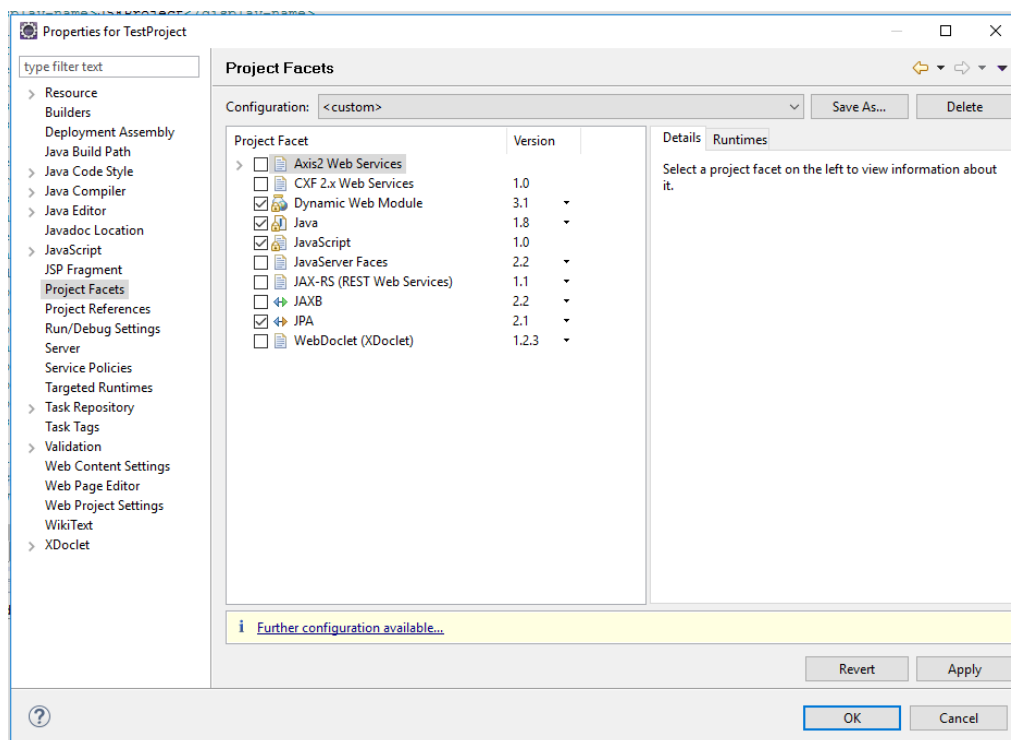
Επιλέγουμε τον Driver που δημιουργήσαμε και συμπληρώνουμε τα στοιχεία της σύνδεσης. Μπορούμε να δοκιμάσουμε αν η σύνδεση είναι επιτυχής με το κουμπί Test Connection.

Ολοκληρώνουμε τη ρύθμιση του JPA πατώντας το κουμπί Finish στο παραπάνω screenshot.

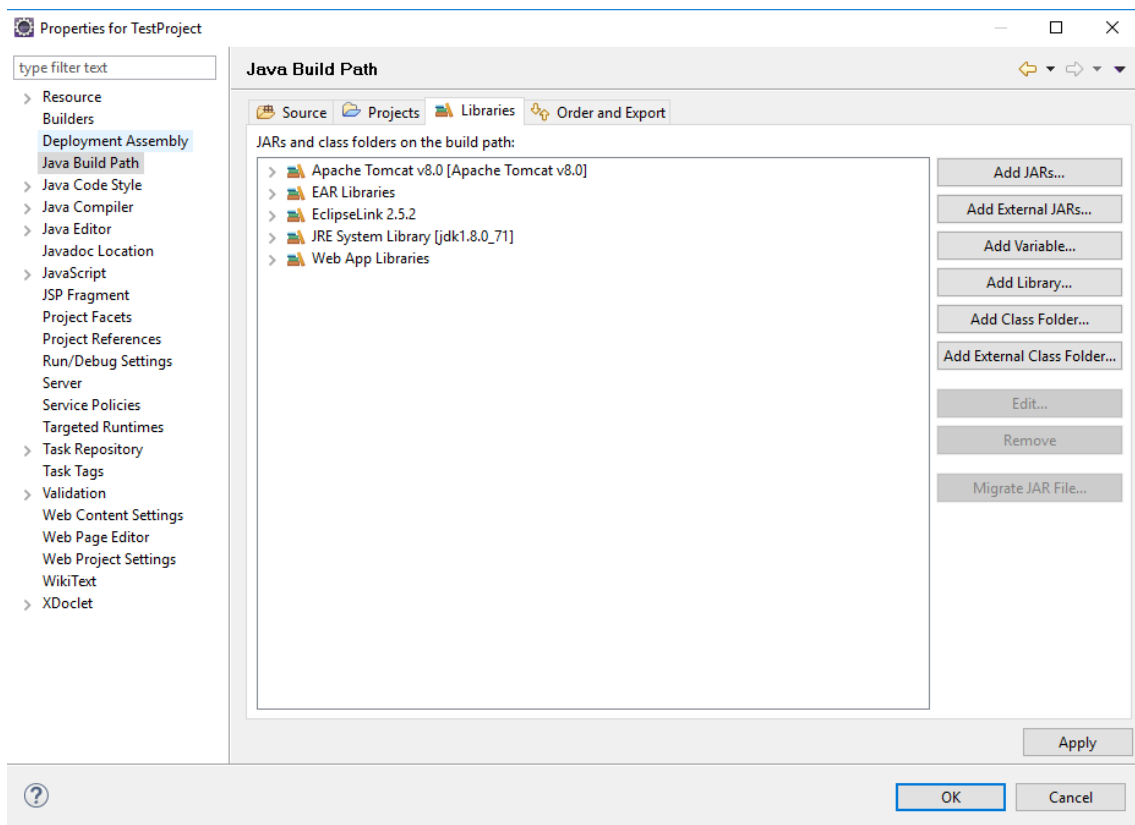




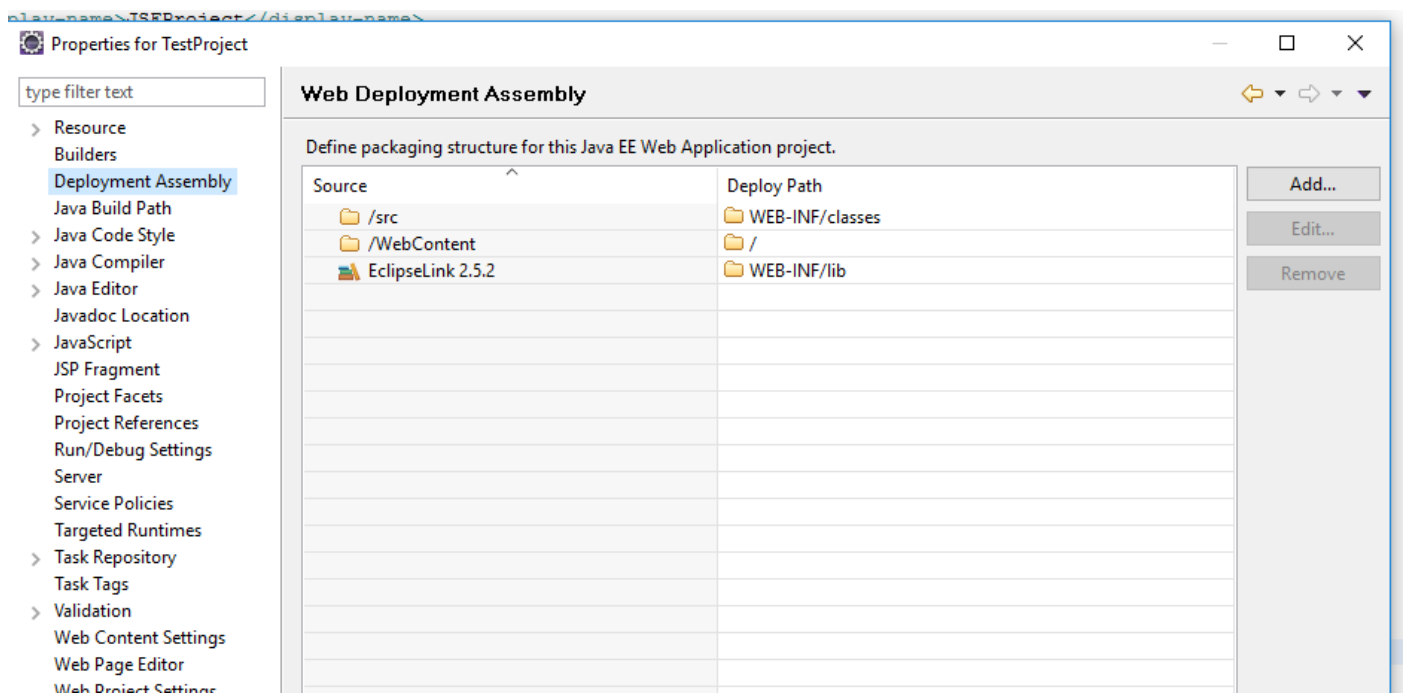
Ολοκληρώνουμε την εγκατάσταση του JPA πατώντας το κουμπί Apply:



Αν επιλέξουμε από τη λίστα αριστερά την επιλογή Java Build Path βλέπουμε ότι έχει προστεθεί η βιβλιοθήκη EclipseLink 2.5.2 όπως επίσης και στο Deployment Assembly.



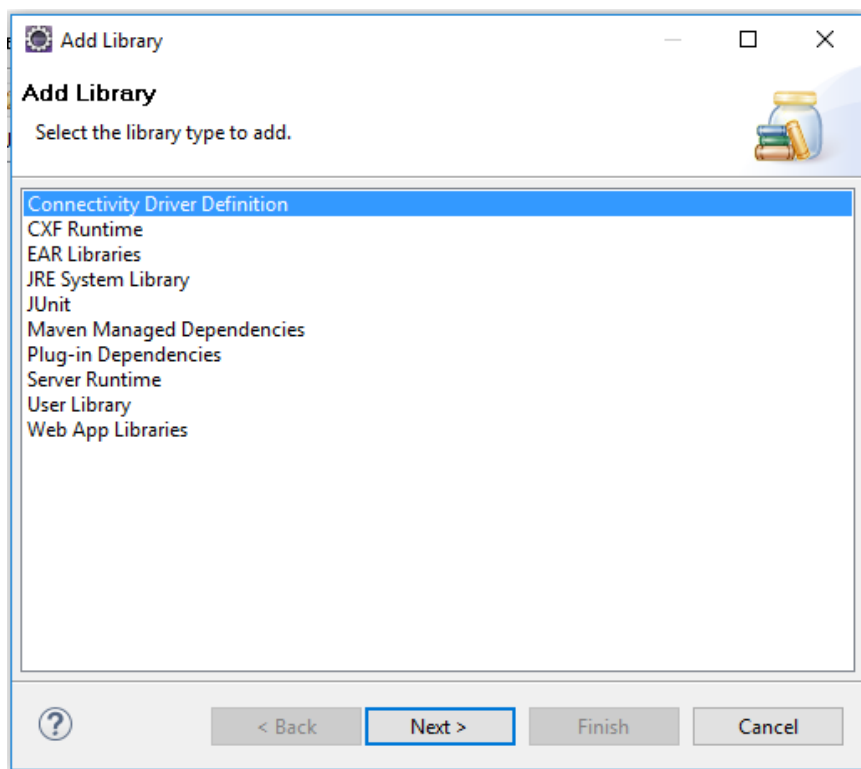
Το build path είναι απαραίτητο για τη μεταγλώττιση του κώδικά μας. Το Deployment Assembly προσδιορίζει ακριβώς ποιες βιβλιοθήκες, ποια αντικειμενικά αρχεία bytecode (.class) και ποιοι κατάλογοι με περιεχόμενο Web θα συμπεριλαμβάνονται στο αρχείο war (Web Archive) το οποίο παράγεται για την ανάπτυξη της εφαρμογής μέσα στον Application Server (Tomcat).



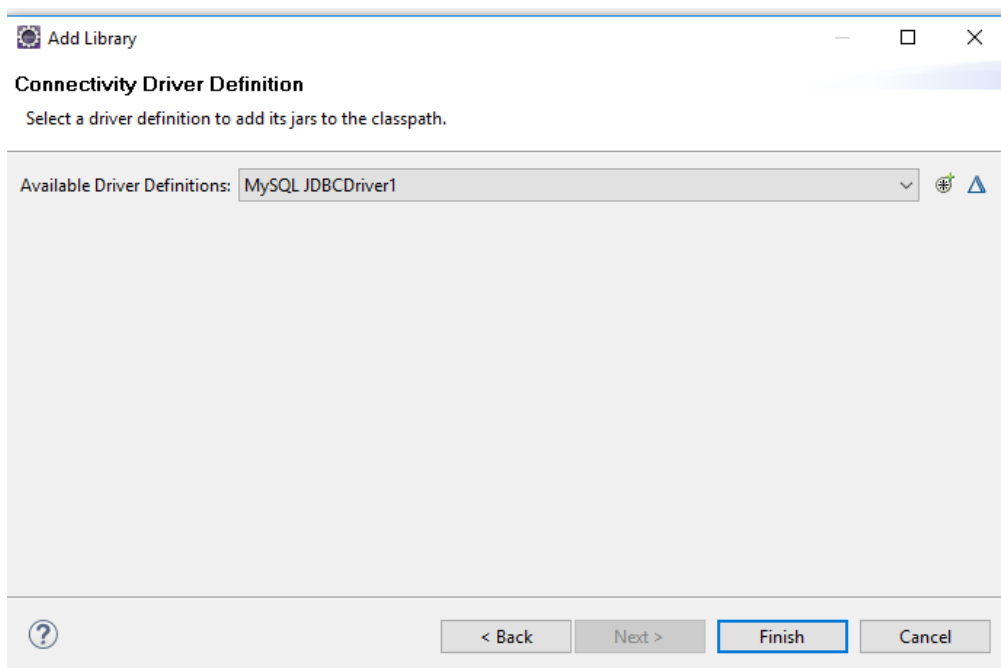
Βλέπουμε ότι και από τις 2 λίστες (build path, deployment assembly) λείπει η βιβλιοθήκη για τον driver της MySQL που πρέπει να προστεθεί χειροκίνητα.

Στο Deployment Assembly περιλαμβάνεται εκτός από τη βιβλιοθήκη του EclipseLink ο κατάλογος WebContent του project όπου τοποθετούμε τις σελίδες HTML, JSP, JSF, Javascript, CSS, εικόνες, κ.λπ. και τον κατάλογο src που περιλαμβάνει τον κώδικα των κλάσεων που γράφουμε για την εφαρμογή μας.

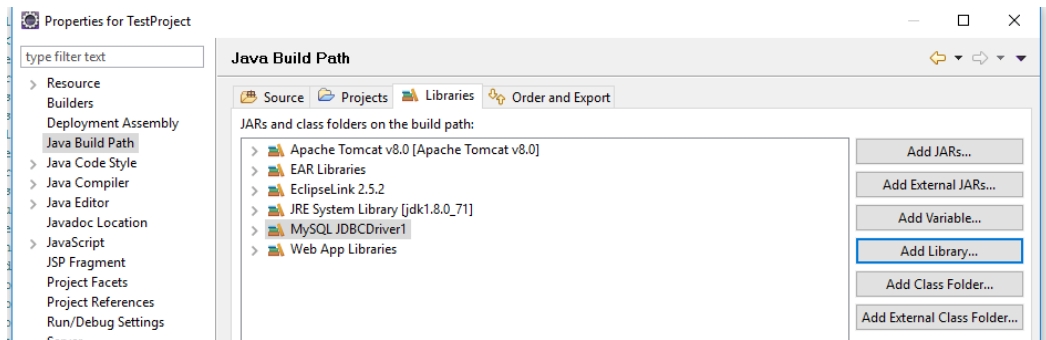
Πλοηγούμαστε στην επιλογή Build Path και πατάμε το κουμπί Add Library



Επιλέγουμε Connectivity Driver Definition και πατάμε Next.

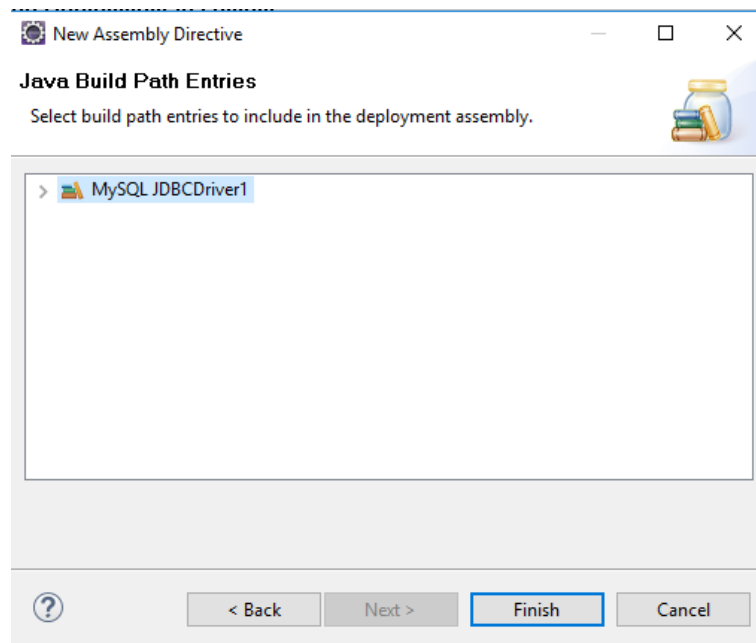
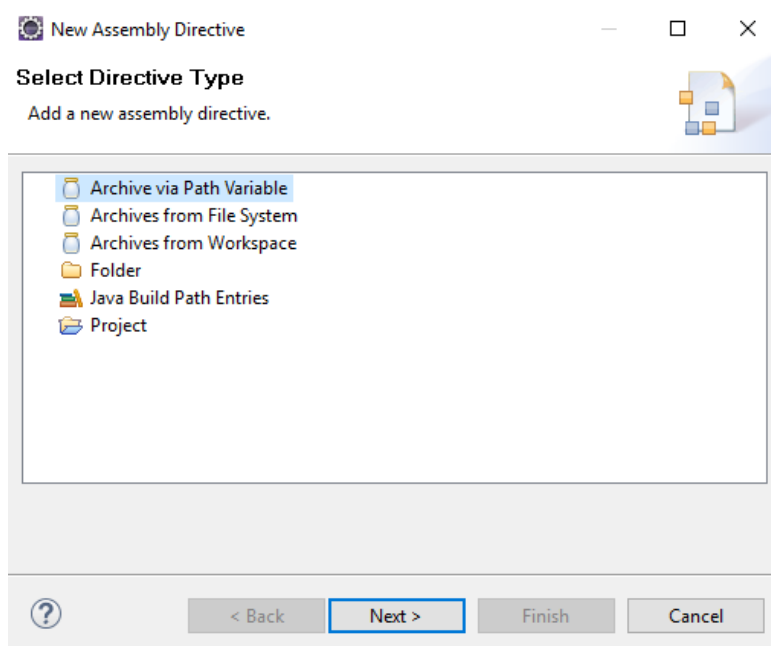


Επιλέγουμε τον Driver που δημιουργήσαμε στα προηγούμενα βήματα (ή είχαμε δημιουργήσει για τις ανάγκες άλλου project) και πατάμε το κουμπί Finish. Βλέπουμε ότι προστέθηκε η βιβλιοθήκη του Driver.

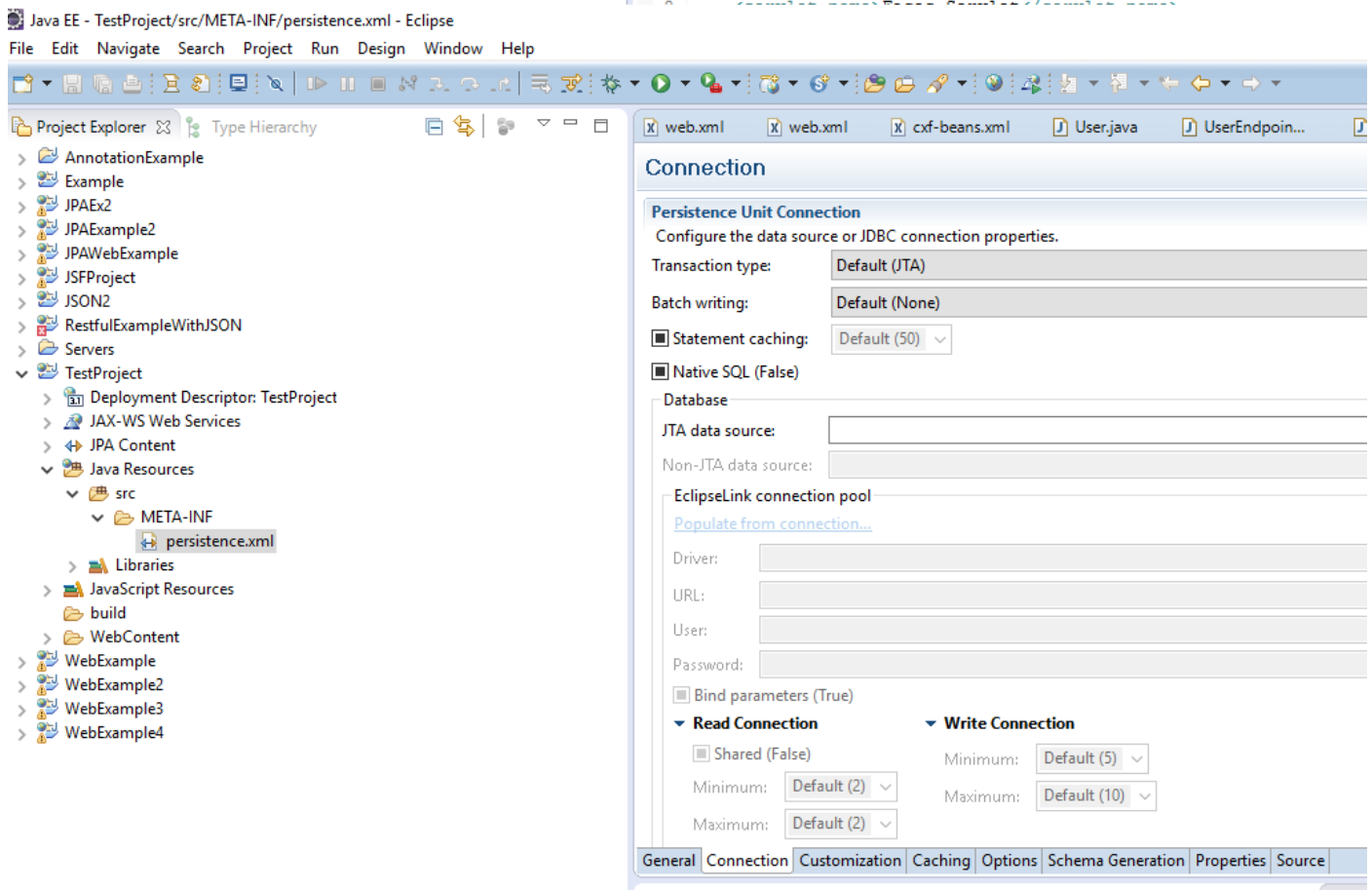


Αυτό δεν αρκεί, πρέπει ο driver να προστεθεί και στο Deployment Assembly. Πατάμε το κουμπί Apply και επιλέγουμε τη ρύθμιση του Deployment Assembly.

Πατάμε το κουμπί Add και επιλέγουμε Java Build Path Entries και πατάμε το κουμπί Next.

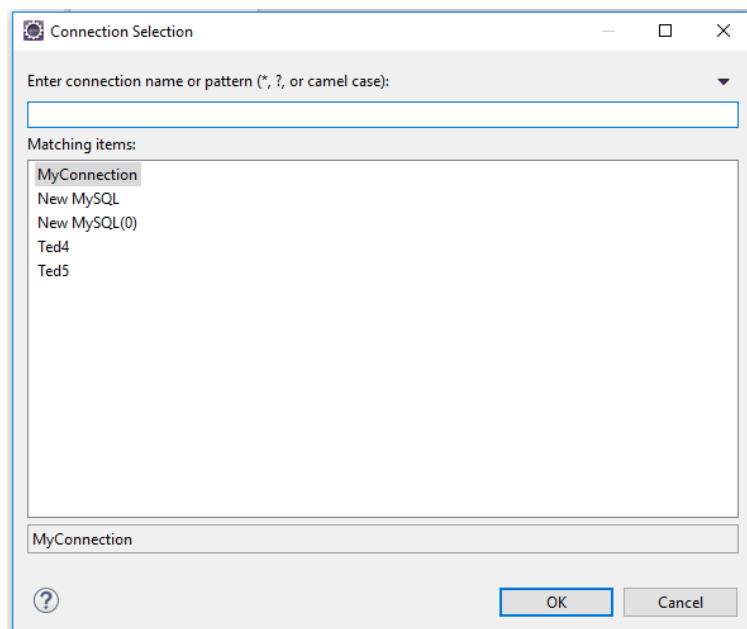


Πατάμε το κουμπί Finish και Apply στη συνέχεια.



Στη συνέχεια πλοηγούμαστε στον κατάλογο Java Resources / src /META-INF και επιλέγουμε το αρχείο persistence.xml που περιλαμβάνει τις ιδιότητες ρύθμισης του JPA.

Αλλάζουμε το Transaction Type σε Resource Local και επιλέγουμε με το ποντίκι τον σύνδεσμο Populate from connection



Επιλέγουμε τη σύνδεση που δημιουργήσαμε στα προηγούμενα βήματα και αντιστοιχεί στη βάση μας. Στο πρώτο screenshot της επόμενης σελίδας εμφανίζεται ο Wizard, στο 2^ο screenshot ο πηγαίος κώδικας του persistence.xml που αντιστοιχεί στις ρυθμίσεις που επιλέξαμε.

The screenshot shows the Eclipse IDE with the 'Persistence Unit Connection' dialog box open. The dialog is titled 'Connection' and contains the following settings:

- Persistence Unit Connection**: Configure the data source or JDBC connection properties.
- Transaction type**: Resource Local
- Batch writing**: Default (None)
- Statement caching**: ☒ Default (50)
- Native SQL**: ☒ (False)
- Database**:
 - JTA data source: (empty)
 - Non-JTA data source: (empty)
- EclipseLink connection pool**:
 - [Populate from connection...](#)
 - Driver**: com.mysql.jdbc.Driver
 - URL**: jdbc:mysql://83.212.123.209:3306/ted
 - User**: ted
 - Password**: (masked with dots)
- Bind parameters**: ☒ (True)
- Read Connection**:
 - ☒ Shared (False)
 - Minimum**: Default (2)
 - Maximum**: Default (2)
- Write Connection**:
 - Minimum**: Default (5)
 - Maximum**: Default (10)

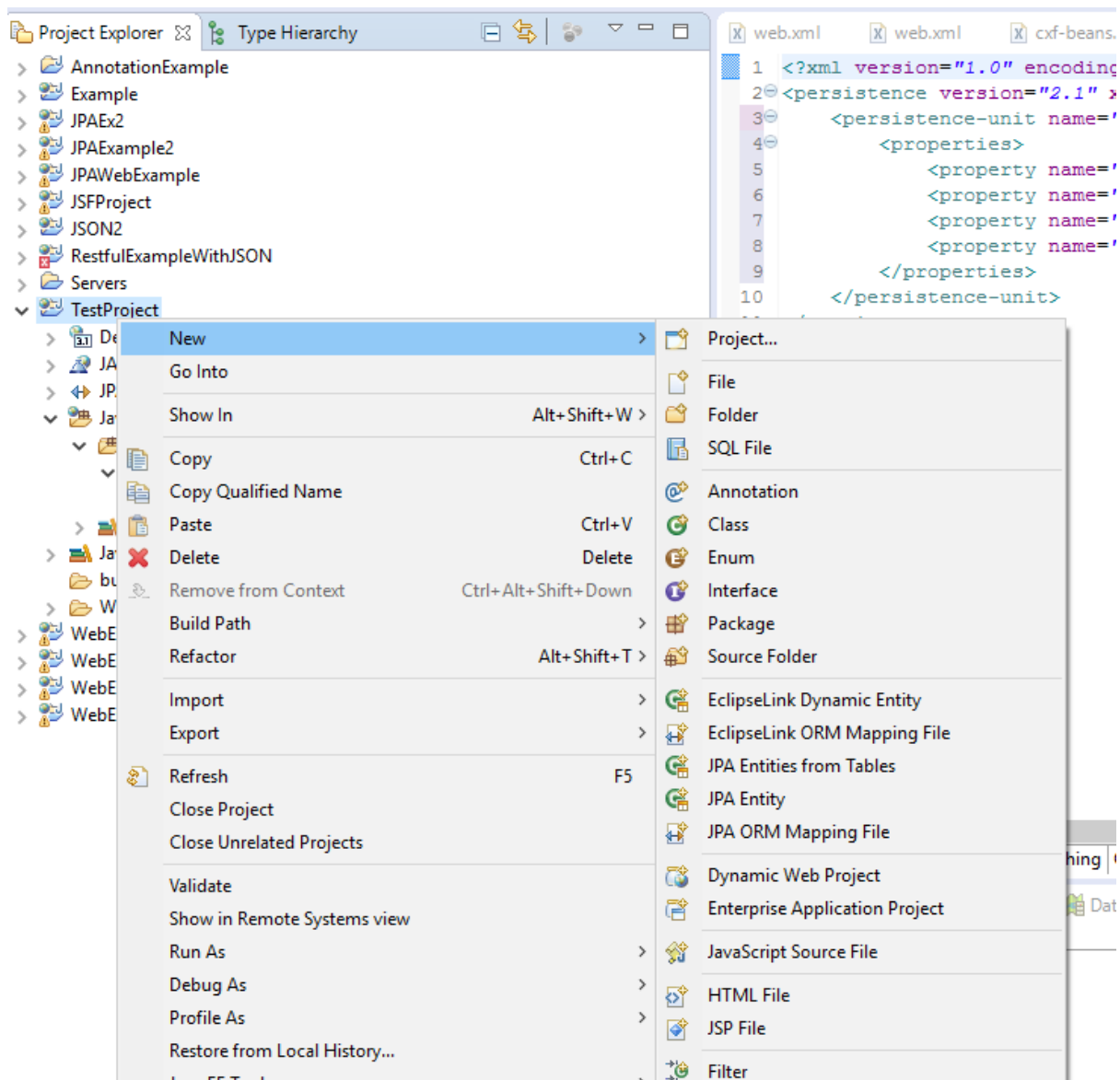
The screenshot shows the Eclipse IDE with the 'persistence.xml' file open. The XML content is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/
3   <persistence-unit name="TestProject" transaction-type="RESOURCE_LOCAL">
4     <properties>
5       <property name="javax.persistence.jdbc.url" value="jdbc:mysql://83.212.123.209:3306/ted"/>
6       <property name="javax.persistence.jdbc.user" value="ted"/>
7       <property name="javax.persistence.jdbc.password" value="ted"/>
8       <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
9     </properties>
10  </persistence-unit>
11 </persistence>
```

The bottom of the IDE shows a tabbed interface with the following tabs: General, Connection, Customization, Caching, Options, Schema Generation, Properties, and Source. The 'Source' tab is currently selected.

6. Δημιουργία κλάσεων Entity με τα κατάλληλα JPA Annotations.

Αφού ολοκληρώθηκε επιτυχώς το βήμα 5 μπορούν να δημιουργηθούν με αυτοματοποιημένο τρόπο τα entity classes που αντιστοιχούν στους πίνακες της βάσης μας.



Με δεξί κουμπί πάνω στο Project επιλέγουμε New -> JPA Entities from Tables. Αν αυτό δεν φαίνεται στο παράθυρο επιλογής, πατάμε την επιλογή Other και το αναζητούμε εκεί. Επιλέγουμε τους πίνακες για τους οποίους θέλουμε να δημιουργήσουμε Entity Classes. Πατάμε την επιλογή Next για να δούμε τα Table Associations που προκύπτουν από τους περιορισμούς ξένου κλειδιού που θα πρέπει υποχρεωτικά να έχουν οριστεί στο επίπεδο της βάσης, ώστε να εξετάσουμε αν είναι σωστά.

Generate Custom Entities

Select Tables

Select tables from which entities will be generated.

Connection: **Ted4**

(Note: You must have an active connection to select schema.)

Schema: **ted4**

Tables: type filter text

- ☒ course
- ☒ student
- ☒ studentcourse

☒ List generated classes in persistence.xml

Restore Defaults

< Back Next > Finish Cancel

Generate Custom Entities

Table Associations

Edit a table association by selecting it and modifying the controls in the editing panel.


Table associations

studentcourse * 1 student
Each student has many studentcourse.

studentcourse * 1 course
Each course has many studentcourse.

< Back Next > Finish Cancel

Στη συνέχεια ρυθμίζουμε διάφορα θέματα της απεικόνισης (π.χ. αν θα γίνεται γέννηση του κλειδιού (identity, sequence, table) ή όχι (none)). Αν τα πεδία που αφορούν τις συσχετίσεις 1-N θα εισάγονται σε συλλογή Set ή List και άλλες σχετικές ρυθμίσεις.

 Generate Custom Entities

Customize Defaults

Optionally customize aspects of entities that will be generated by default from database tables. A Java package should be specified.

Mapping defaults

Key generator:

none

Sequence name:

You can use the patterns \$table and/or \$pk in the sequence name. These patterns will be replaced by the table name and the primary key column name when a table mapping is generated.

Entity access:

☒ Field ☐ Property

Associations fetch:

☒ Default ☐ Eager ☐ Lazy

Collection properties type:

☐ java.util.Set ☒ java.util.List

☐ Always generate optional JPA annotations and DDL parameters

Domain java class

Source folder:

TestProject/src

Browse...

Package:


model

Browse...

Superclass:

Browse...

Interfaces:

 java.io.Serializable

Add...

Remove



< Back

Next >

Finish

Cancel

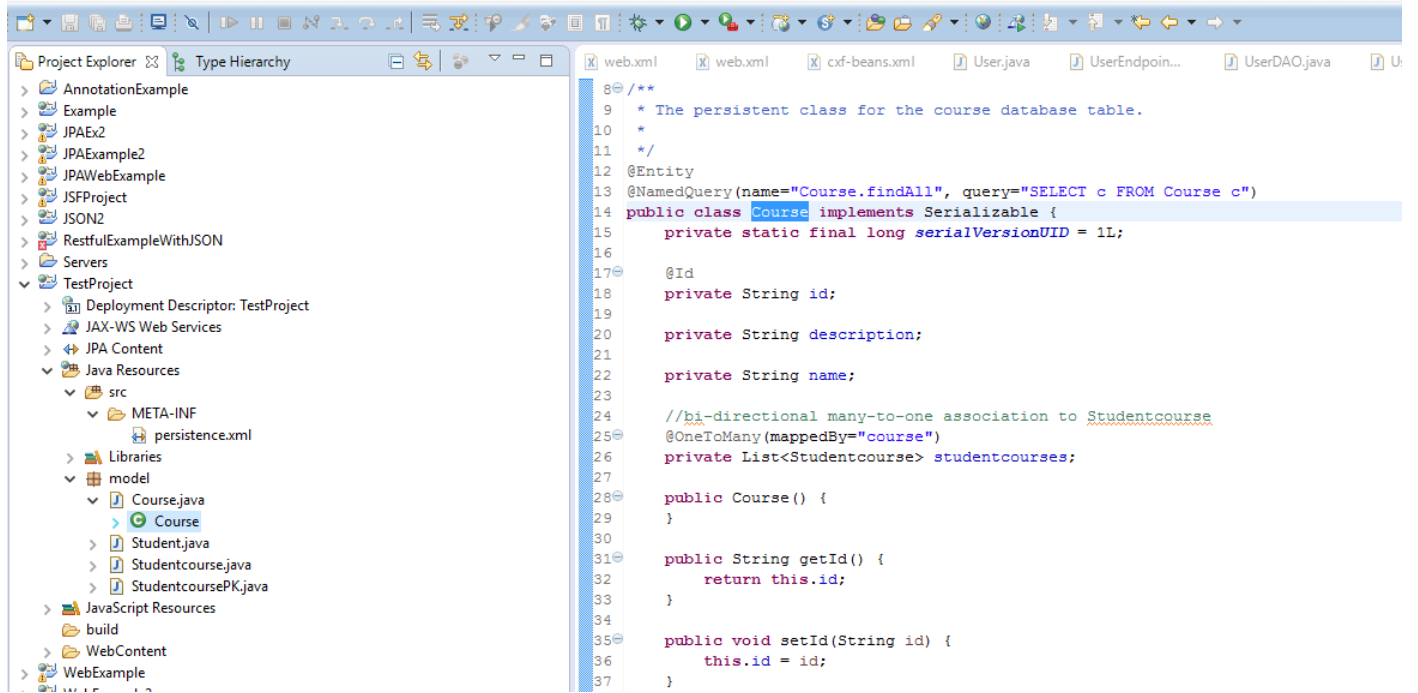
Επιλέγοντας Next μπορούμε να ρυθμίσουμε τις επιμέρους οντότητες μία-μία αντί για τη συνολική ρύθμιση που γίνεται στο προηγούμενο βήμα:

The screenshot shows a window titled "Generate Custom Entities" with a subtitle "Customize Individual Entities". The window is divided into several sections:

- Tables and columns:** A list of database tables with expandable icons: "course", "student", and "studentcourse".
- Mapping defaults:** A section for configuring default mapping settings.
 - Class name:** A text field containing "Course".
 - Key generator:** A dropdown menu set to "none".
 - Sequence name:** An empty text field.
 - Help text:** "You can use the patterns \$table and/or \$pk in the sequence name. These patterns will be replaced by the table name and the primary key column name when a table mapping is generated."
 - Entity access:** Two radio buttons, "Field" (selected) and "Property".
- Domain java class:** A section for configuring the Java domain class.
 - Superclass:** An empty text field with a "Browse..." button to its right.
 - Interfaces:** A list box containing "java.io.Serializable" with an information icon to its left. To the right of the list box are "Add..." and "Remove" buttons.

At the bottom of the window, there is a navigation bar with a help icon, and four buttons: "< Back", "Next >", "Finish" (highlighted with a blue border), and "Cancel".

Πατώντας το κουμπί Finish δημιουργούνται οι κλάσεις.

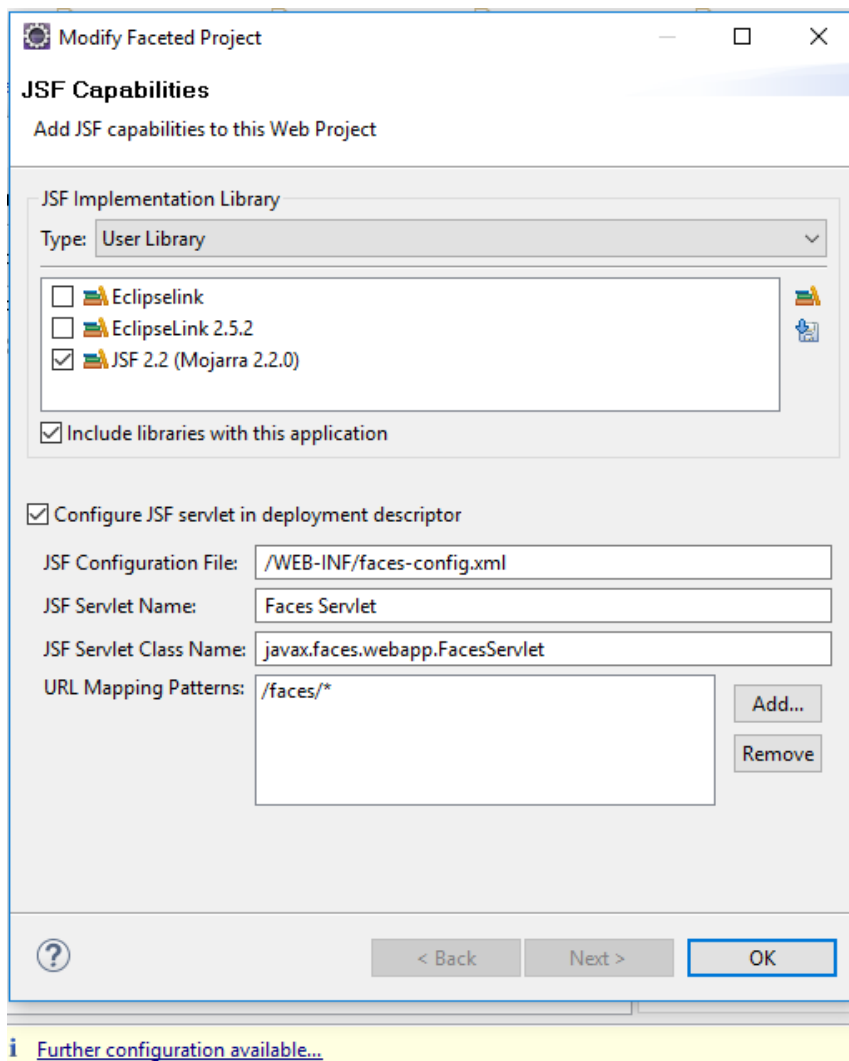


The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays the project structure for 'TestProject'. The 'model' package contains the 'Course.java' file, which is currently selected. The main editor window shows the code for 'Course.java'. The code is a JPA entity class that implements 'Serializable'. It includes annotations for '@Entity', '@NamedQuery', and '@OneToMany'. The class has private fields for 'id', 'description', and 'name', and a private list for 'studentcourses'. It also has a constructor, a 'getId()' method, and a 'setId()' method.

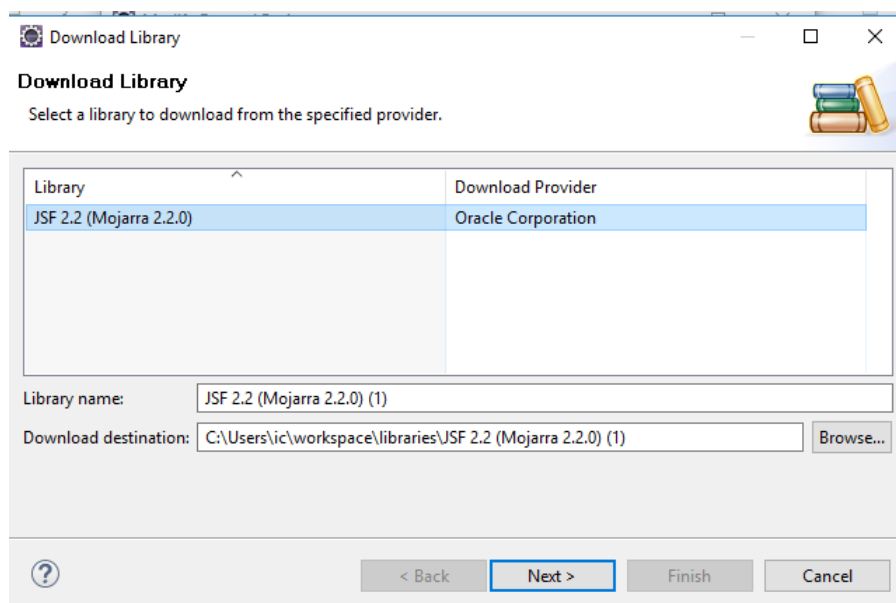
```
8 /**
9  * The persistent class for the course database table.
10  *
11  */
12 @Entity
13 @NamedQuery(name="Course.findAll", query="SELECT c FROM Course c")
14 public class Course implements Serializable {
15     private static final long serialVersionUID = 1L;
16
17     @Id
18     private String id;
19
20     private String description;
21
22     private String name;
23
24     //bi-directional many-to-one association to Studentcourse
25     @OneToMany(mappedBy="course")
26     private List<Studentcourse> studentcourses;
27
28     public Course() {
29     }
30
31     public String getId() {
32         return this.id;
33     }
34
35     public void setId(String id) {
36         this.id = id;
37     }
```


7. Ρύθμιση Java Server Faces

Πλοηγούμαστε πάλι στα Project Facets από τα Properties του Project (σελ. 7) και τσεκάρουμε την επιλογή Java Server Faces και πλοηγούμαστε πάλι στον σύνδεσμο Further Configuration Available



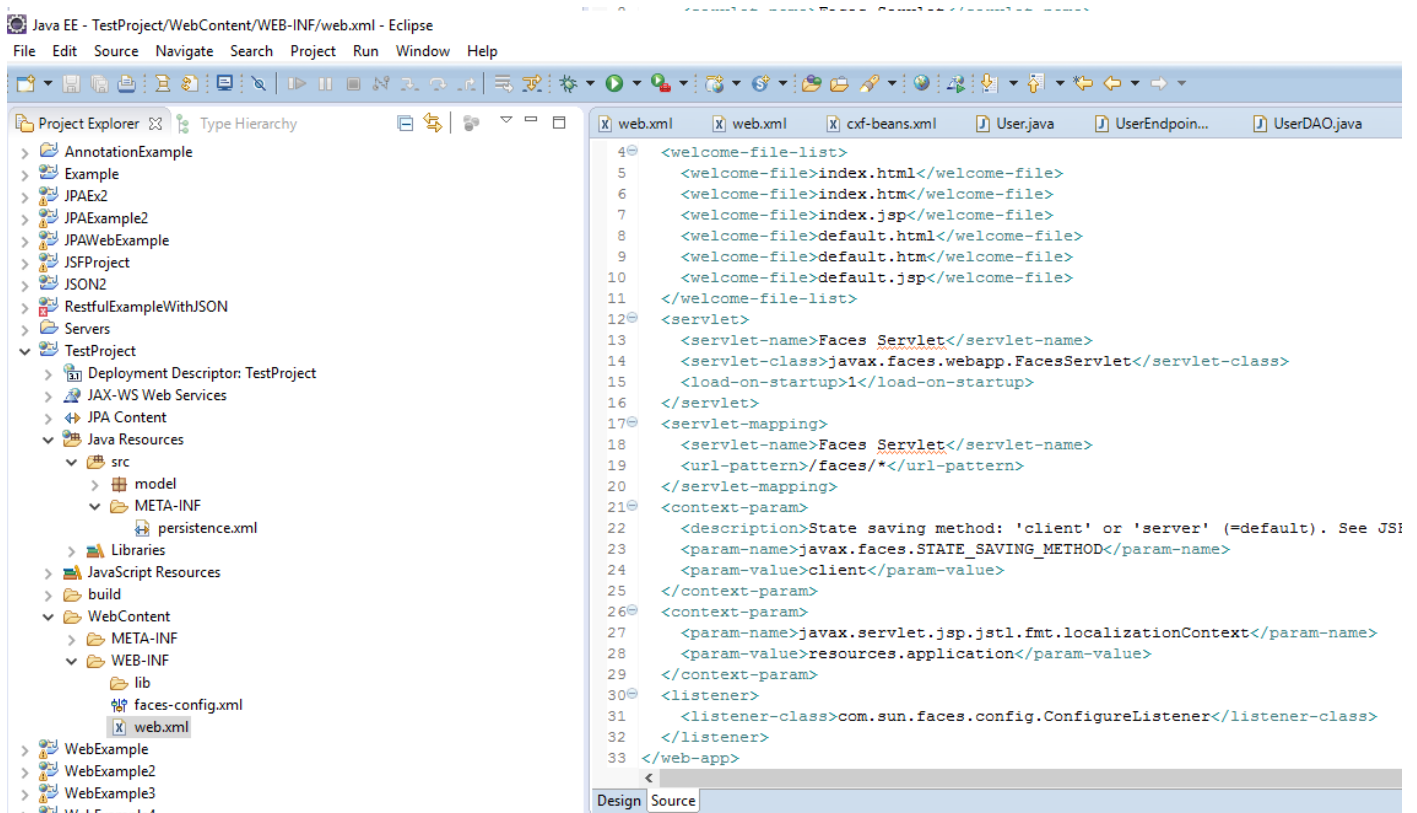
Αν δεν εμφανίζεται κάποια βιβλιοθήκη επιλέγουμε το εικονίδιο με τη δισκέτα και το βέλος στο παράθυρο που εμφανίζεται:



Επιλέγουμε να κατεβάσουμε μία από τις βιβλιοθήκες που εμφανίζονται και πατάμε Next. Αποδεχόμαστε την άδεια και πατάμε Finish.

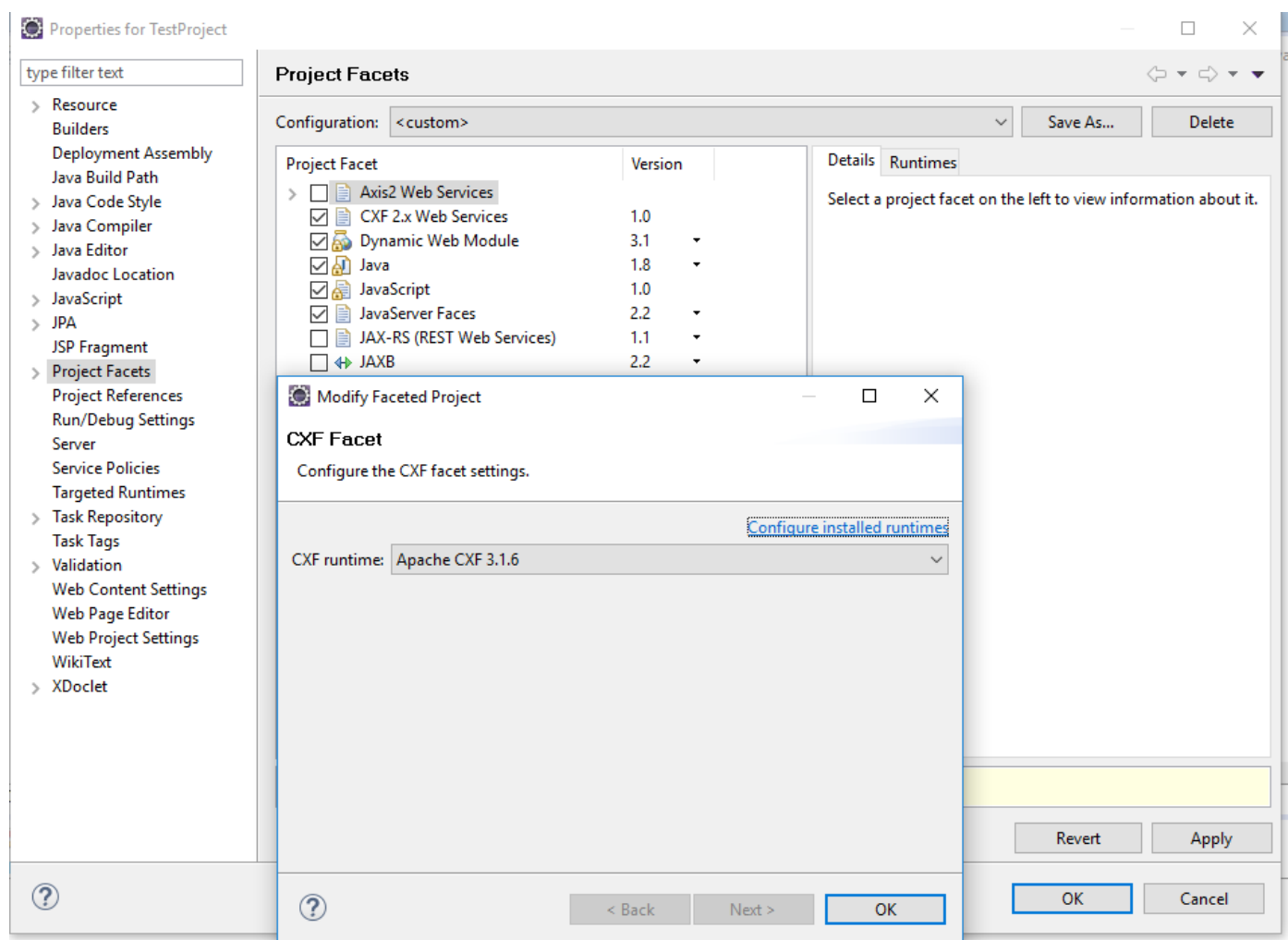
Στο 1^ο screenshot της προηγούμενης σελίδας εμφανίζονται στο κατώτερο τμήμα οι ρυθμίσεις του JSF. Δηλαδή ποιο θα είναι το URL Mapping των facelets (δηλαδή http://localhost:8080/TestProject/faces/*). Με άλλα λόγια αν έχουμε κάποιο foo.xhtml που έχει JSF tags μέσα στο WebContent για να ενεργοποιηθεί ως facelet θα πρέπει αντί για το κανονικό του URL που αντιστοιχεί στη θέση του στο σύστημα αρχείων να δώσουμε τη διεύθυνση [CONTEXT_URL]/faces/foo.xhtml. Αυτό στην πράξη σημαίνει ότι το HTTP request αποστέλλεται πρώτα στο Faces Servlet του JSF (δηλαδή τον Controller) ο οποίος στη συνέχεια διαχειρίζεται τα facelets.

Αφού ολοκληρώσουμε τη ρύθμιση πατάμε το κουμπί Ok και στη συνέχεια το κουμπί Apply στο παράθυρο Project Properties.



8. Ρύθμιση CXF RESTful Services

Πλοηγούμαστε πάλι στα Project Facets από τα Properties του Project (σελ. 7), τσεκάρουμε την επιλογή CXF 2.x Web Services και πλοηγούμαστε πάλι στον σύνδεσμο Further Configuration Available



Σε αυτή την περίπτωση πρέπει υποχρεωτικά να κατεβάσουμε το CXF Runtime από το Web (<http://www.apache.org/dyn/closer.lua/cxf/3.1.6/apache-cxf-3.1.6.zip>) και να το αποσυμπιέσουμε σε κατάλογο που μπορεί να διαβάσει το Eclipse.

Στη συνέχεια πλοηγούμαστε στον σύνδεσμο Configure installed runtimes αν δεν έχουμε ήδη εγκαταστήσει το CXF runtime. Πατάμε το κουμπί Add και εισάγουμε τον κατάλογο όπου έχουμε αποσυμπιέσει το CXF Runtime. Πατάμε το κουμπί Finish και στο παράθυρο CXF 2.x Preferences πατάμε το κουμπί Apply και το κουμπί OK.

Στη συνέχεια πατάμε το κουμπί OK στο παράθυρο Modify Faceted Project και τα κουμπιά Apply και OK στο παράθυρο Project Facets.

Για τη ρύθμιση όμως του CXF θα πρέπει να προσθέσουμε χειροκίνητα ορισμένα πράγματα στο web.xml. Όπως θα δείτε στο παράδειγμα RestfulExampleWithJSON που έχει αναρτηθεί στο e-class το web.xml περιέχει τα στοιχεία του CXF Servlet που στην πραγματικότητα διαχειρίζεται τις αιτήσεις προς τα REST ή SOAP CXF Services. Και σε αυτή την περίπτωση υπάρχει ένα URL Mapping που αντιστοιχεί στο παράδειγμά μας σε <http://localhost:8080/TestProject/services/>. Με άλλα λόγια, τα μονοπάτια που ορίζουμε μέσα στις υλοποιήσεις των services ακολουθούν το παραπάνω URL.

Επίσης, θα πρέπει να είναι ορισμένο το cxf-beans.xml όπου ορίζονται οι κλάσεις που παρέχουν τη λειτουργικότητα των services αλλά και ο listener που εμφανίζεται αμέσως παρακάτω στο web.xml.

Το αρχείο cxf-beans.xml έχει την κάτωθι μορφή:

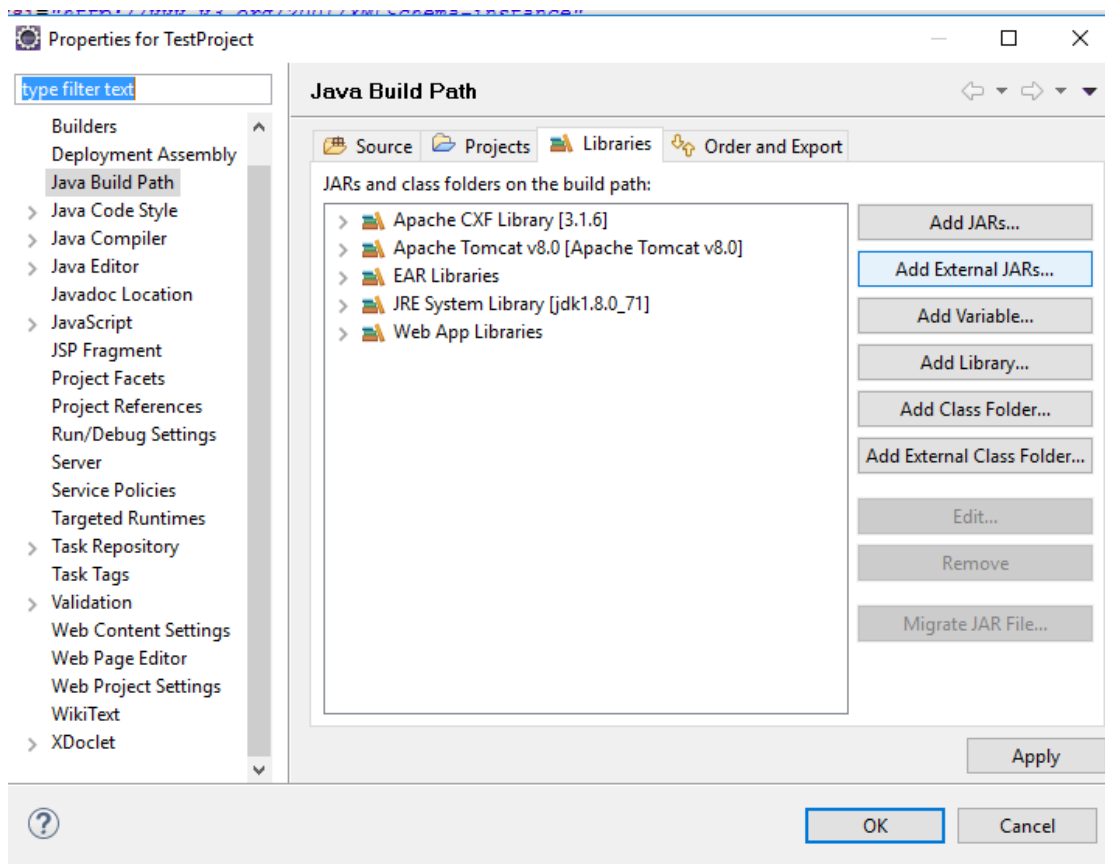
Εντός του στοιχείου `jaxrs:serviceBeans` ορίζονται τα beans που υλοποιούν τη λειτουργικότητα των υπηρεσιών. Θα πρέπει για κάθε τέτοια εγγραφή μέσα στο `serviceBeans` να υπάρχει και ένα στοιχείο `bean` που συσχετίζεται το `id` που δίνεται εντός του `serviceBeans` με μια συγκεκριμένη κλάση, π.χ. `Users` και `rest.UserEndPoint`.

Επιπρόσθετα, θα πρέπει εντός του στοιχείου `jaxrs:providers` να ορίζεται ανώνυμο στοιχείο `bean` του οποίου ορίζεται η κλάση που αναλαμβάνει να μετασχηματίζει τα αντικείμενα σε JSON strings και το αντίστροφο.

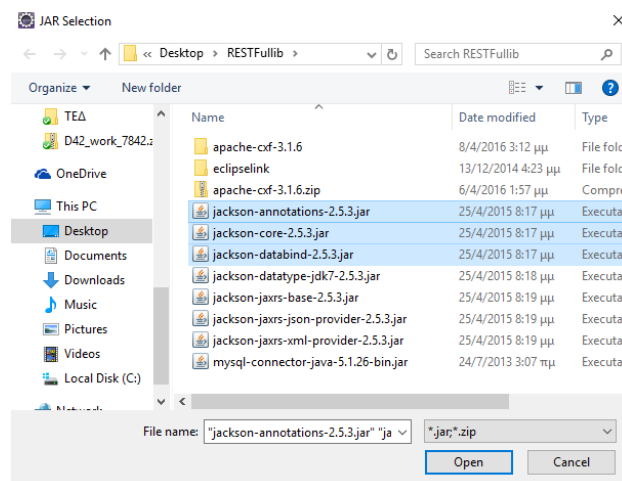
Στην υλοποίηση του παραδείγματος χρησιμοποιούμε τη βιβλιοθήκη Jackson. Θα πρέπει να έχουμε κατεβάσει τη βιβλιοθήκη αυτή από το Web (τουλάχιστον τα jars

<http://repo1.maven.org/maven2/com/fasterxml/jackson/core/jackson-core/2.7.0/jackson-core-2.7.0.jar>,
<http://repo2.maven.org/maven2/com/fasterxml/jackson/core/jackson-annotations/2.7.0/jackson-annotations-2.7.0.jar> και <http://repo1.maven.org/maven2/com/fasterxml/jackson/core/jackson-databind/2.7.0/jackson-databind-2.7.0.jar>).

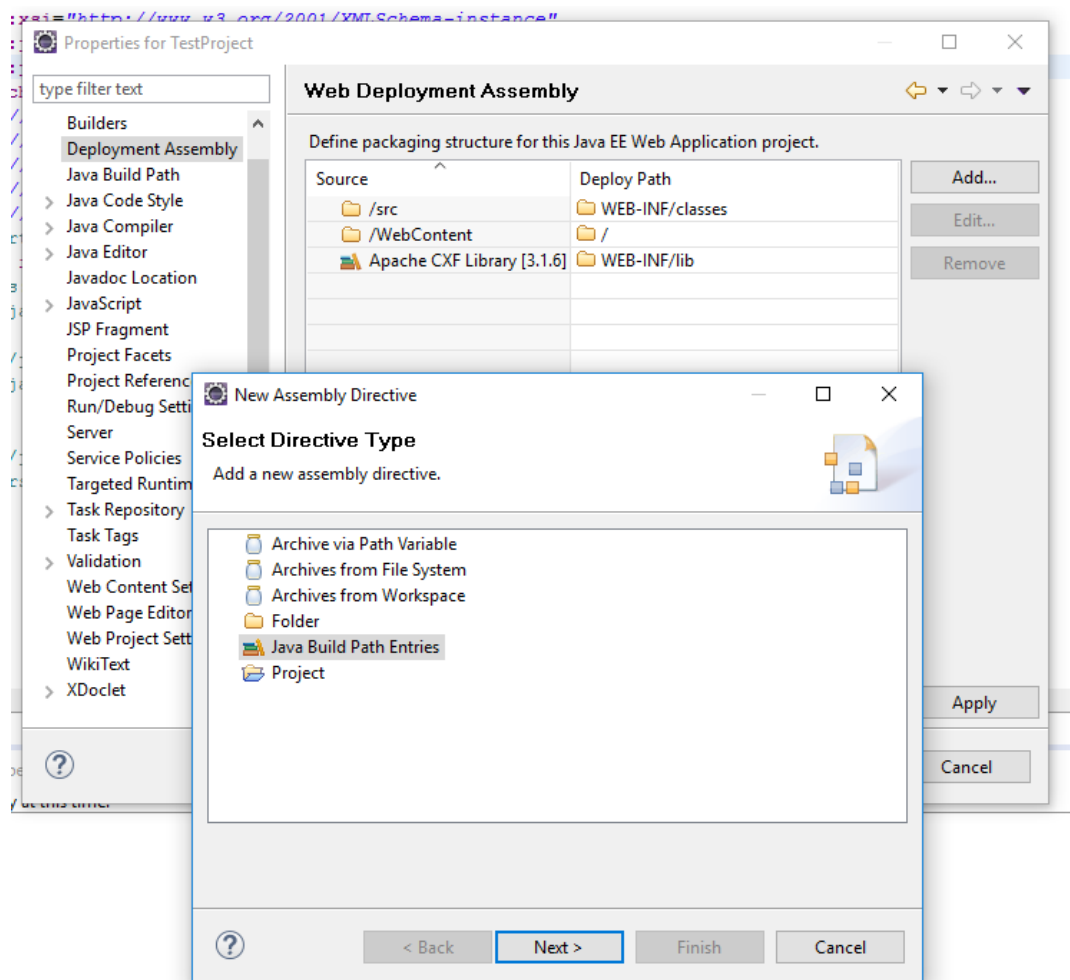
Αφού κατεβάσουμε αυτά τα jars θα πρέπει να τα εισάγουμε στο build path του project και στο deployment assembly:



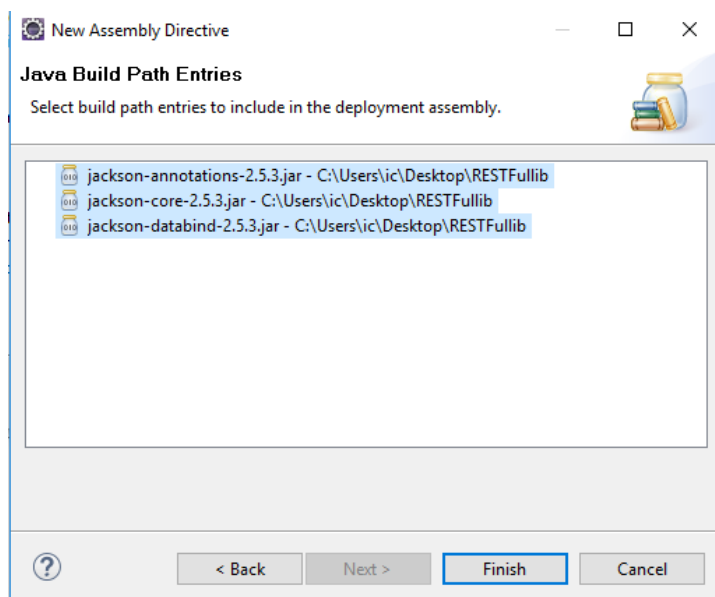
Πατάμε το κουμπί `Add External JARs` και επιλέγουμε αυτά που χρειάζονται:



Στη συνέχεια πατάμε το κουμπί Apply και OK. Δεν ξεχνάμε στη συνέχεια να τα εισάγουμε και στο Deployment Assembly:



Επιλέγουμε και πάλι Java Build Path Entries > Next και στο επόμενο παράθυρο επιλέγουμε τα JARs που θέλουμε να εισάγουμε στο deployment assembly και πατάμε το κουμπί Finish.



Τέλος, πατάμε το κουμπί Apply και OK στο παράθυρο Properties for TestProject και έτσι η διαδικασία εγκατάστασης ολοκληρώνεται.

Οι παραπάνω οδηγίες είναι ιδιαίτερα λεπτομερείς και αν τις κατανοήσετε μπορείτε αντίστοιχα να εισάγετε άλλες βιβλιοθήκες και δυνατότητες που ενδεχομένως θα χρειαστείτε.