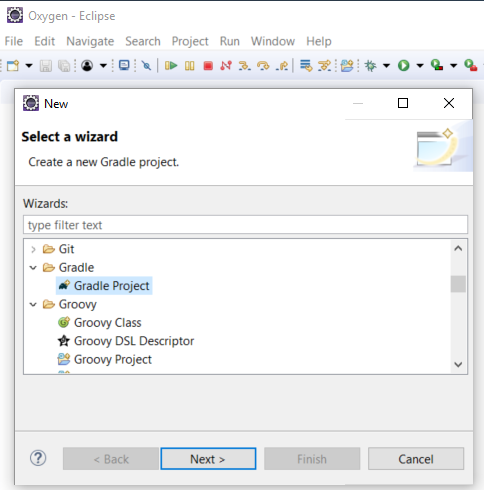
# Prerequisites

1. Install Java JDK
2. Download Tomcat 9
3. Download Eclipse and run a sample Dynamic Web App with Tomcat 9
4. Install Node ( include npm ) <https://nodejs.org/en/>
5. Install Angular CLI (npm install -g @angular/cli}

# Build JEE Service

## Install Eclipse with Gradle plugin

## Create New Gradle Project



## Edit build.gradle

* Add plugins

apply plugin: 'war'

apply plugin: 'eclipse-wtp'

* Add webbApp directory

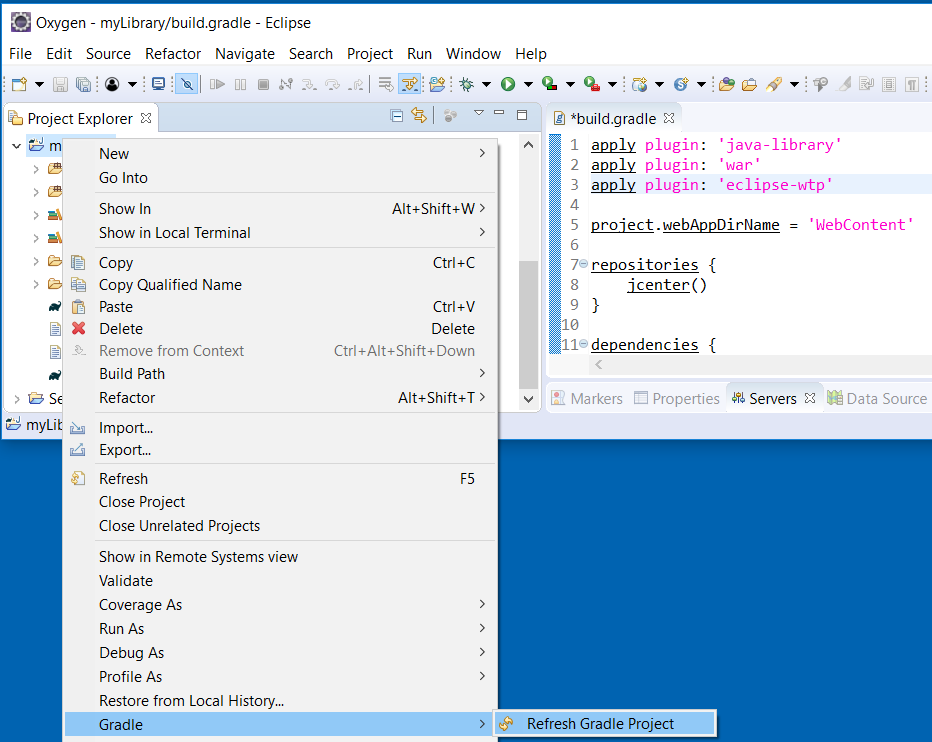
project.webAppDirName = 'WebContent'

* Add dependencies

compile 'org.glassfish.jersey.containers:jersey-container-servlet:2.27'

compile 'org.glassfish.jersey.media:jersey-media-json-jackson:2.27'

## Refresh Gradle Project



## Create Model

### Book

**public** **class** Book {

**private** **int** \_id;

**private** **int** \_authorId;

**private** String \_title;

**private** **int** \_year;

**public** Book() {} // This is required for jersey-media-json-jackson binding

}

### Author

**public** **class** Author {

**private** **int** \_id;

**private** String \_name;

**public** Author() {}

}

### Tag

**public** **class** Tag {

**private** **int** \_id;

**private** String \_name;

**public** Tag(){}

}

## Generate the Getter/Setters

## Add CONSTANTS for the Data Model

Note that you should add a space at the beginning and end of the SQL fragments. The extra spaces are fine.

### Book

**public** **static** **final** String ***SQL\_INSEWRT\_FIELDS*** = " ( bookId, authorId, title, year) ";

**public** **static** **final** String ***SQL\_INSERT\_FIELDS*** = " VALUES (?,?,?,?) ";

**public** **static** **final** String ***SQL\_UPDATE\_FIELDS*** = " bookId=?, authorId=?, title=?, year=? ";

### Author

Similar to above

### Tag

Similar to above

### Add Constructor to build Object from ResultSet

## Create Super Class SQLite DAO

**public** **class** SQLiteDAO {

**protected** **static** **final** String ***DB\_PATH*** = "/opt/apps/data/myLibrary.db";

**protected** Connection openConnection(String filepath) {

Connection conn = **null**;

**try** {

Class.*forName*("org.sqlite.JDBC");

// Enforces Referential Integrity

SQLiteConfig config = **new** SQLiteConfig();

config.enforceForeignKeys(**true**);

conn = DriverManager.*getConnection*("jdbc:sqlite:" +

filepath, config.toProperties());

} **catch** (Exception e) {

e.printStackTrace();

}

**return** conn;

}

}

## Create Sub Classe Book DAO

**public** **class** BookDAO **extends** SQLiteDAO{

**private** **static** **final** String ***SQL\_GET\_BOOKS*** =

"SELECT \* FROM book INNER JOIN author USING (authorId);";

**public** List<Book> getBooks(){

List<Book> books = **new** ArrayList<>();

**try**(

Connection conn = openConnection(***DB\_PATH***);

PreparedStatement getPS = conn.prepareStatement(***SQL\_GET\_BOOKS***);){

ResultSet rs = getPS.executeQuery();

**while** (rs.next()) {

books.add(**new** Book(rs));

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** books;

}

}

## Create GET Resource

@Path("books")

**public** **class** BookResource {

@GET

@Produces(MediaType.***APPLICATION\_JSON***)

**public** Response getBooks() {

List<Book> books = **new** BookDAO().getBooks();

**return** Response.*ok*(books, MediaType.***APPLICATION\_JSON***).build();

}

}

## Test REST Web Service

## Request

<http://localhost:8080/myLibrary/rest/books>

## Response

[{"id":1,"year":1949,"authorId":1,"title":"Nineteen Eighty-Four"}]

# Add CORS Filter

You need this if calling the API from a different <protocol>://<server>:<port>

See: https://community.blackboard.com/thread/5246-configuring-cors-for-rest-apis

## Tomcat includes CORS Filter

Add to web.xml

<filter>

<filter-name>CorsFilter</filter-name>

<filter-class>org.apache.catalina.filters.CorsFilter</filter-class>

<init-param>

<param-name>cors.allowed.origins</param-name>

<param-value>\*</param-value>

</init-param>

….

# Create Angular UI Project

* ng new appAngular
* cd appAngular
* ng serve