

TILDA- GETTING STARTED

2019-09-25

INTRO

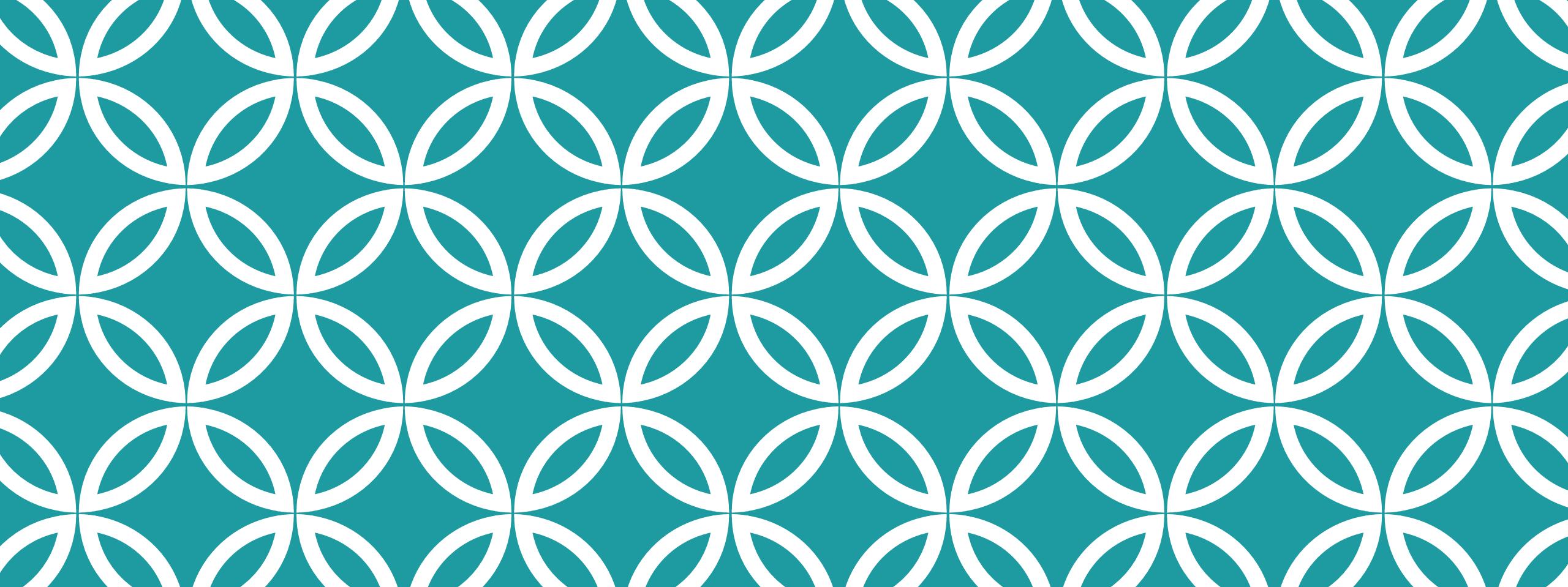
- This presentation should allow you to get started with Tilda
 - Should take 1h to 90mn
 - Set up your development environment with Java, Eclipse, Tomcat and Postgres
 - Create a project
 - Learn the Tilda conventions to get started
 - Go through a simple yet non-trivial set of example to build an app
- Pre-requisites
 - You need to be somewhat familiar with Java development and a Java IDE
 - You need some passing experience/familiarity with JSON
 - You need some passing experience/familiarity with SQL

GENERAL ARCHITECTURE

- Tilda rests on human-readable and editable JSON-based model definitions that can be edited in any JSON-capable editor and are processed via a variety of command-line utilities:
 - Gen: validate the model(s), generate ORM code artifacts and searchable/navigatable HTML documentation
 - Docs: Outputs HTML and SQL documentation to the file system from the model(s)
- The ‘T’ in Tilda stands for “Transparent”, and all artifacts are generated in human-readable form, fully commented and documented
 - Although Tilda allows to model complex data, the framework is architected from the ground up so that all artifacts generated are clear and clean so that the connection between an element in a Tilda definition file and its impact on the runtime or database is easy to see
 - Generated code relies heavily on the Java compiler to actualize in code many patterns, so that complex migrations (moving tables, changing indices, adding/removing columns etc...) has a direct compile-time impact that can then be easily remedied with standard code editors and refactoring methodologies.
- The ‘I’ in Tilda stands for “Iterative”, and all processes and utilities when working with the framework encourage a fluid team-based iterative approach from design to deployment.
 - Migrate: migrates a database to the model(s)

DEVELOPMENT ENVIRONMENT

- For this tutorial, we'll use the following software
 - Java SE JDK 11.0.3
 - <https://www.techspot.com/downloads/5553-java-jdk.html>
 - Apache Tomcat 9.0.24
 - <http://tomcat.apache.org>
 - Eclipse IDE for Enterprise Java Developers 2019-06 R
 - <https://www.eclipse.org/downloads/packages/>
 - PostgreSQL 11.5
 - <https://www.postgresql.org/>
- Be aware that
 - There are no direct dependencies on Tomcat or Eclipse. Any Servlet container or Java IDE will do if you prefer something else.
 - Java 8 is the minimum requirement, but we like to use the latest and greatest
 - Postgres 9.6 is the minimum required, but we like to use the latest and greatest.

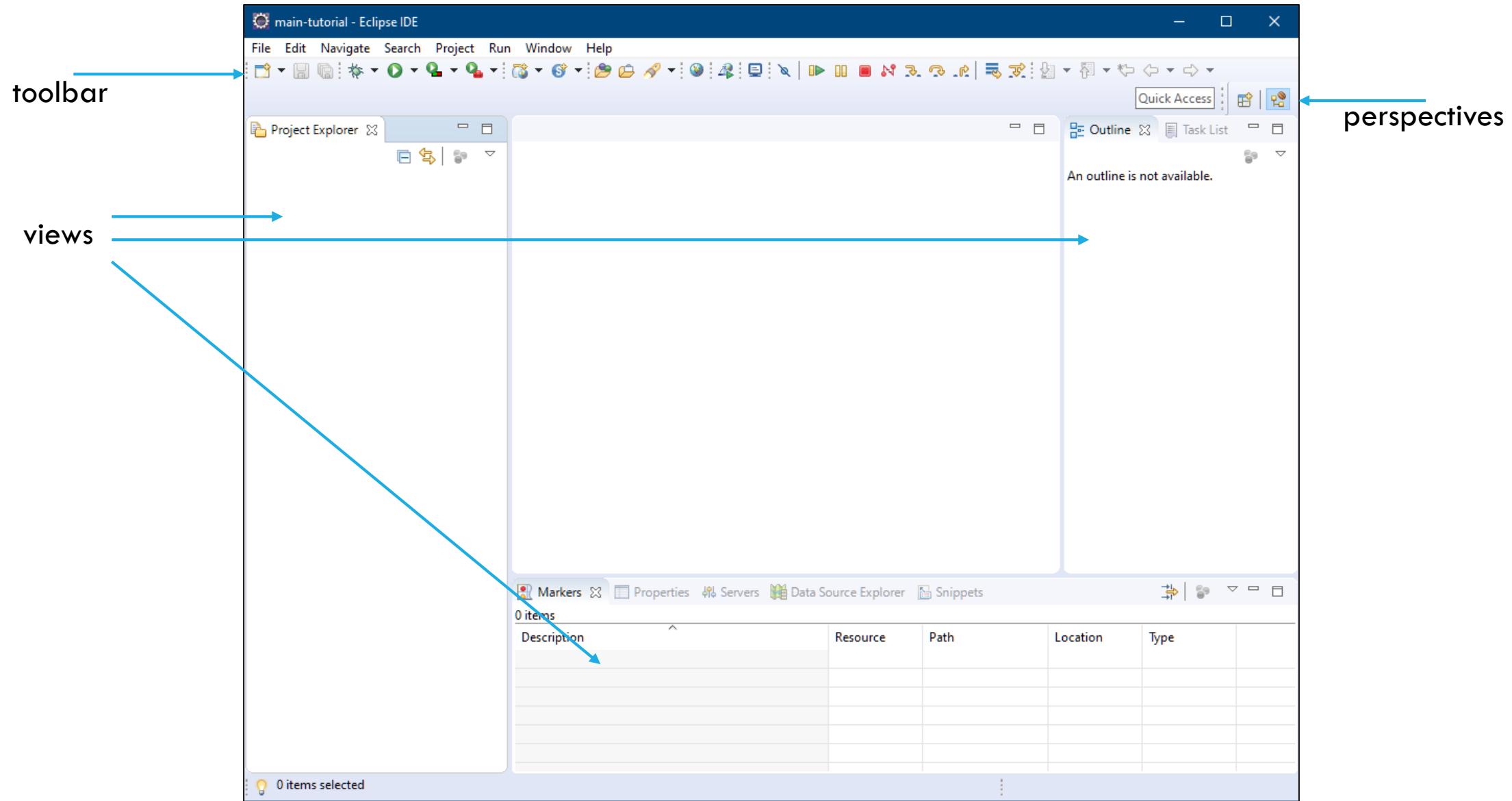


A BEGINNER'S ENVIRONMENT

A BEGINNER'S PROJECT

- Different organizations have different standards in how they set up their Java-based projects.
- We like to set up a base-line project where dependencies and environment configurations are set up and inherited from other projects where application code lives
- The application will be composed of a few JARs and a really basic Web application
- If you are a seasoned developer, then you might want to skim through this section very quickly to get set up.
 - This is really targeted at beginners, younger new employees and so on...

STARTING WITH A CLEAN WORKSPACE

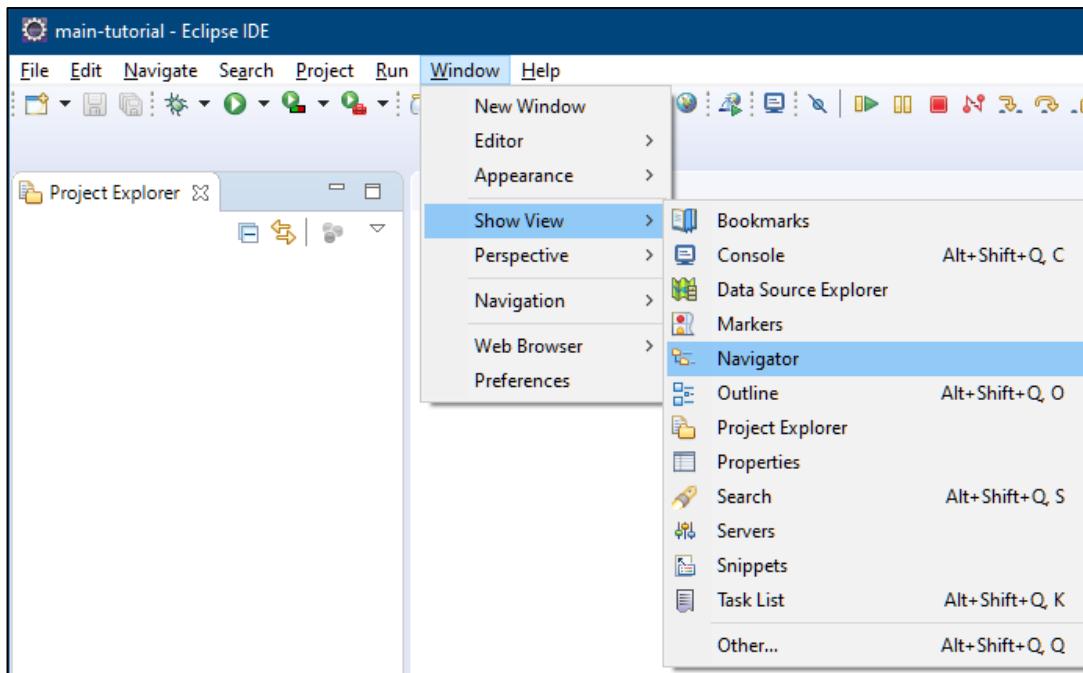


SIMPLE CUSTOMIZATIONS

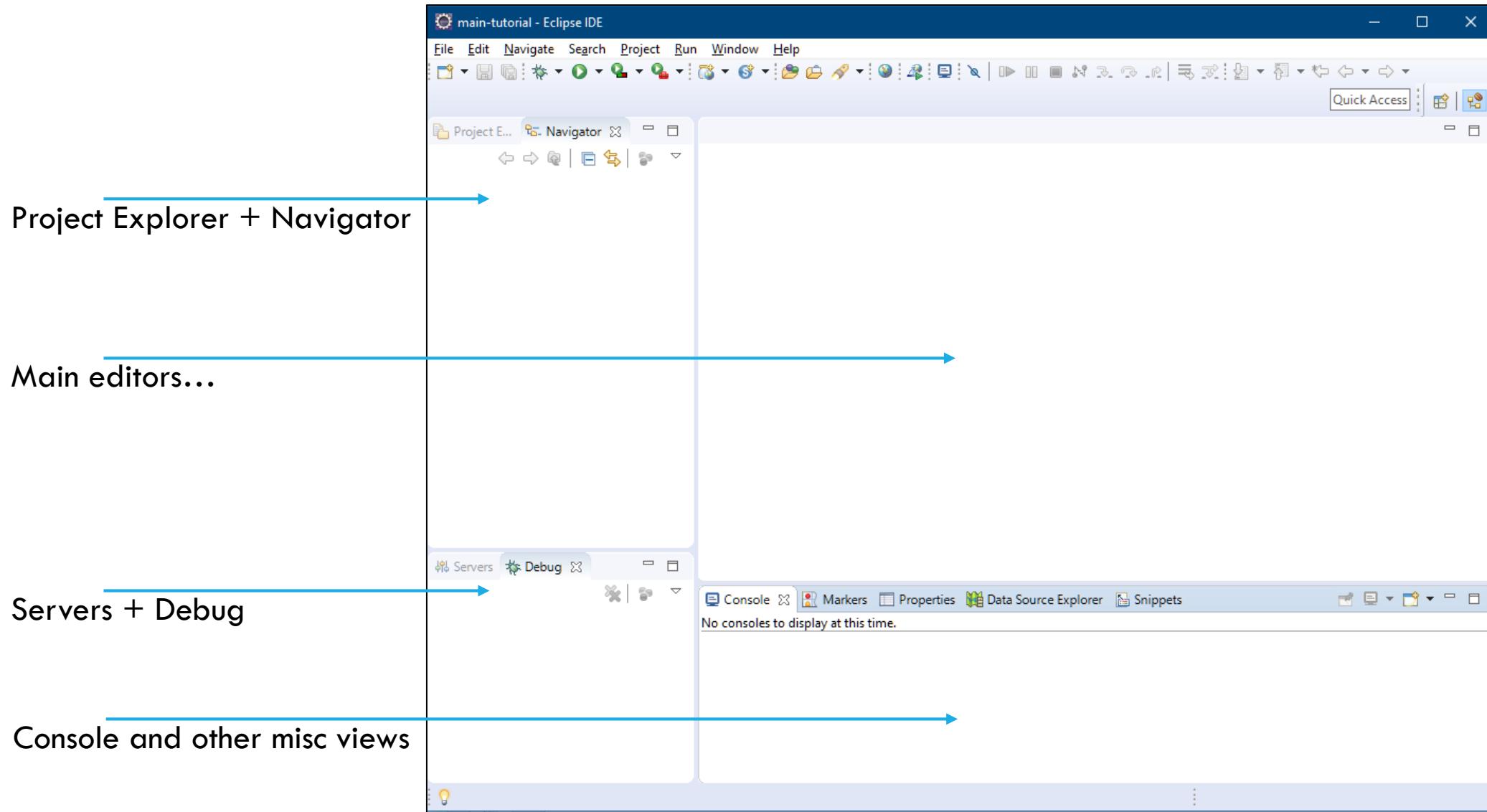
- I have never found a use for the Task List and Outline views.
- The “Project Explorer” is nice, but I also like the “Navigator” as it gives you a view of projects that mirrors more closely the actual file system layout.
- The “Console” view will be critical for us as we use many command-line utilities
- Eclipse supports perspectives where you can customize the views you want. I have found this to not be as useful and like for example to have the “debugger” view in the main perspective
- We’ll configure Tomcat and so we also need the “Servers” view.

EXTRA VIEWS

- You can add new views via the “Window” menu
- Select “other...” to find the other views we like
- You can then drag and drop your views according to taste

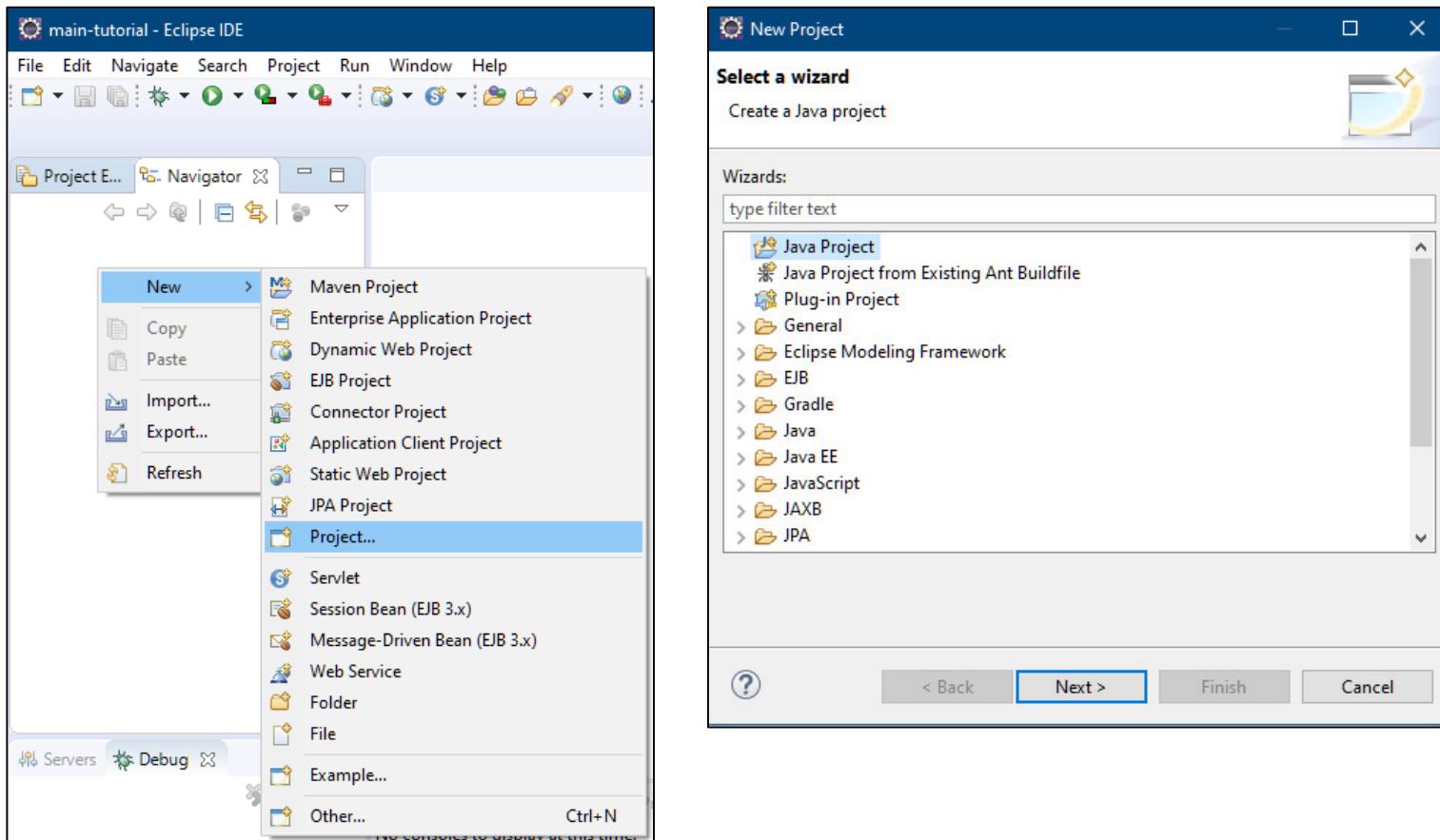


OPTIMIZED PERSPECTIVE (ACCORDING TO SOMEONE!)



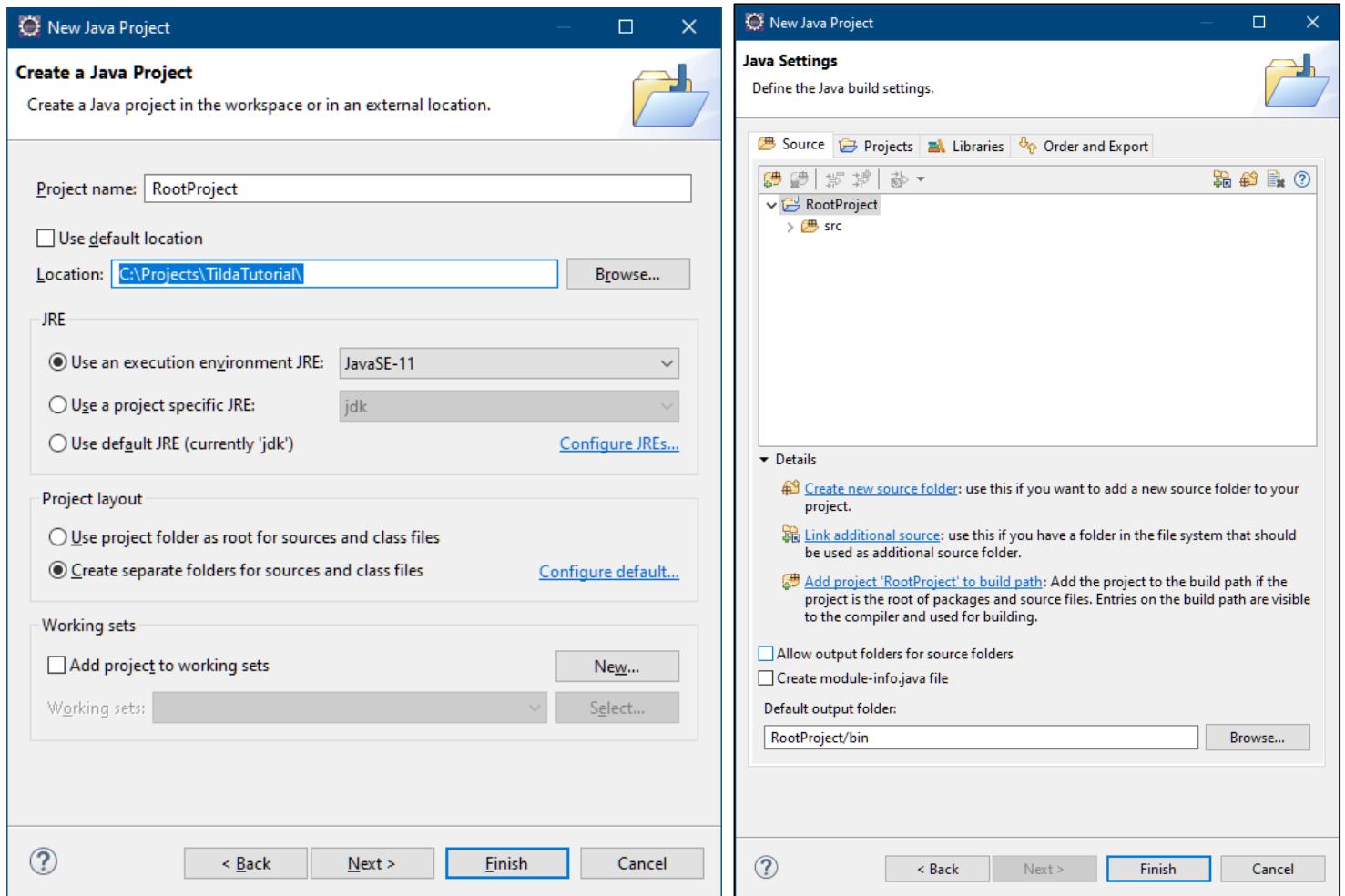
CREATE A “ROOT” PROJECT

- The next few slides show the details of creating a root project.
- Right-click in the Navigator.



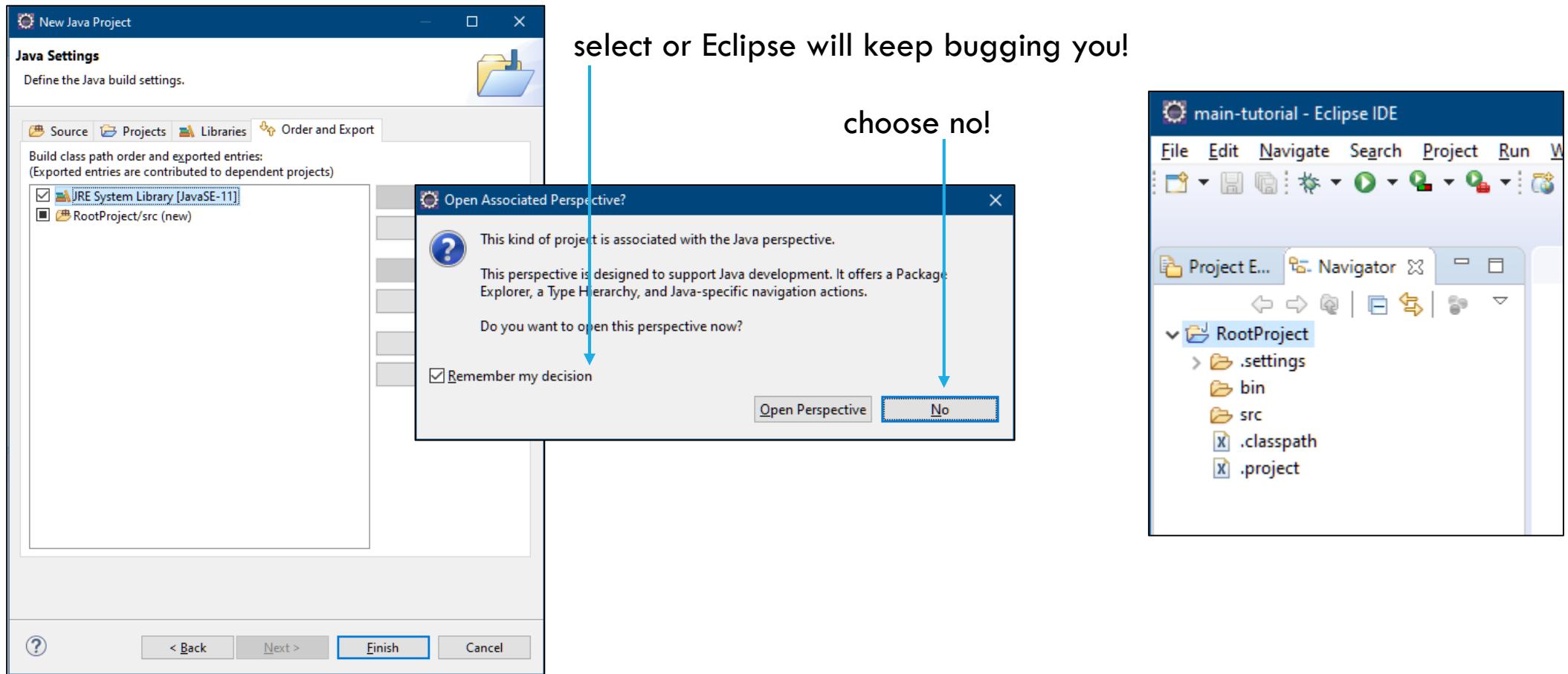
CREATE A “ROOT” PROJECT

- “RootProject” name
- C:\Projects\TildaTutorial
- Disable the Module file



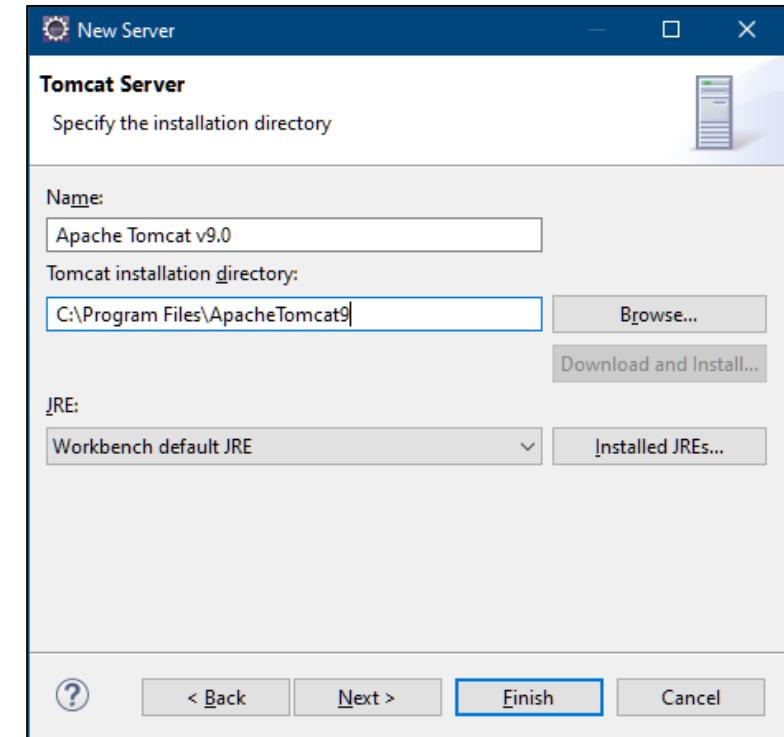
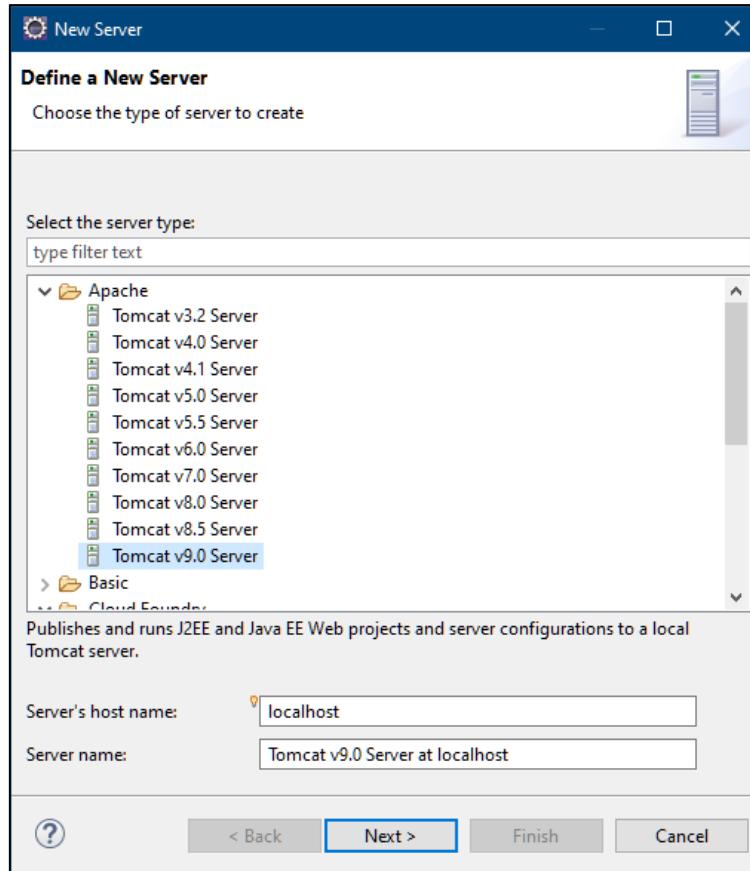
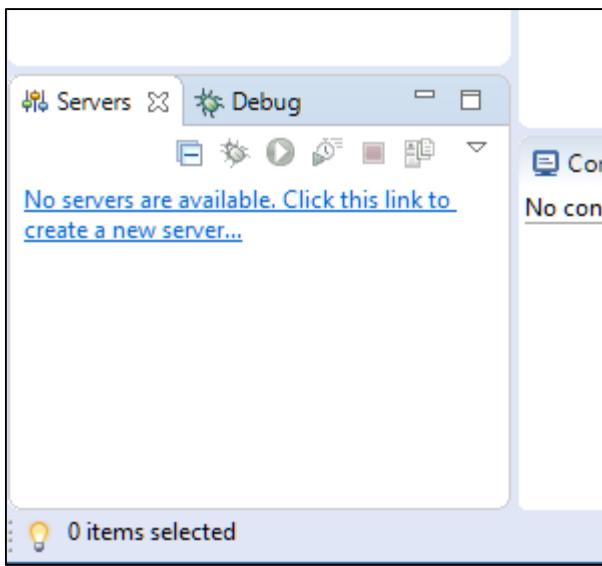
CREATE A “ROOT” PROJECT

- Export the JRE and don’t switch perspective



SET UP TOMCAT

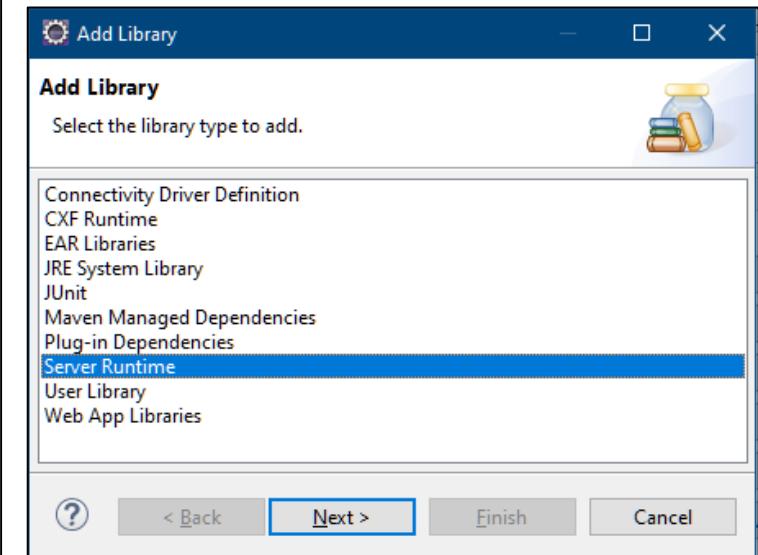
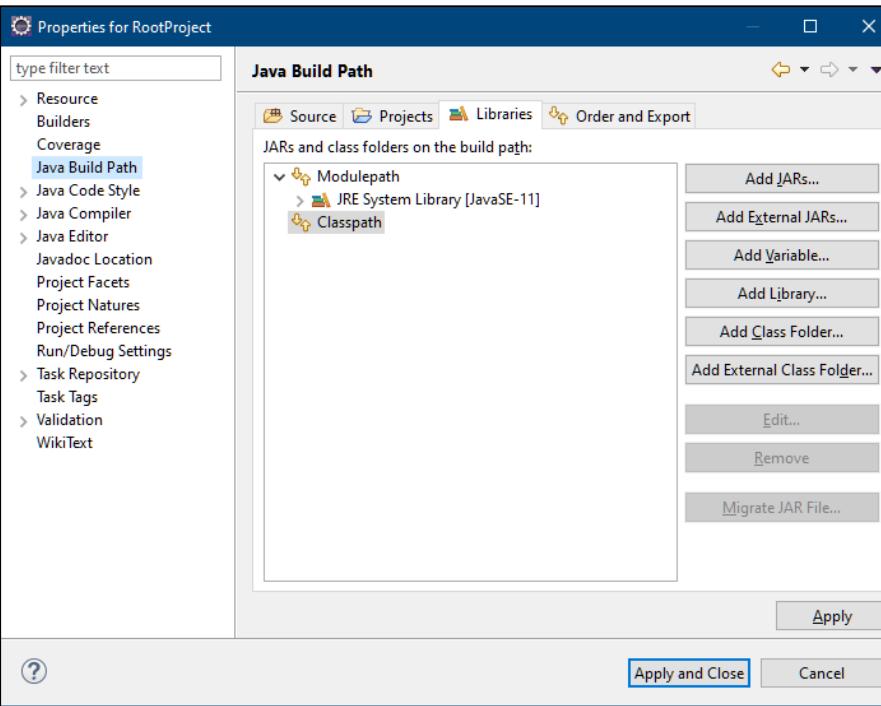
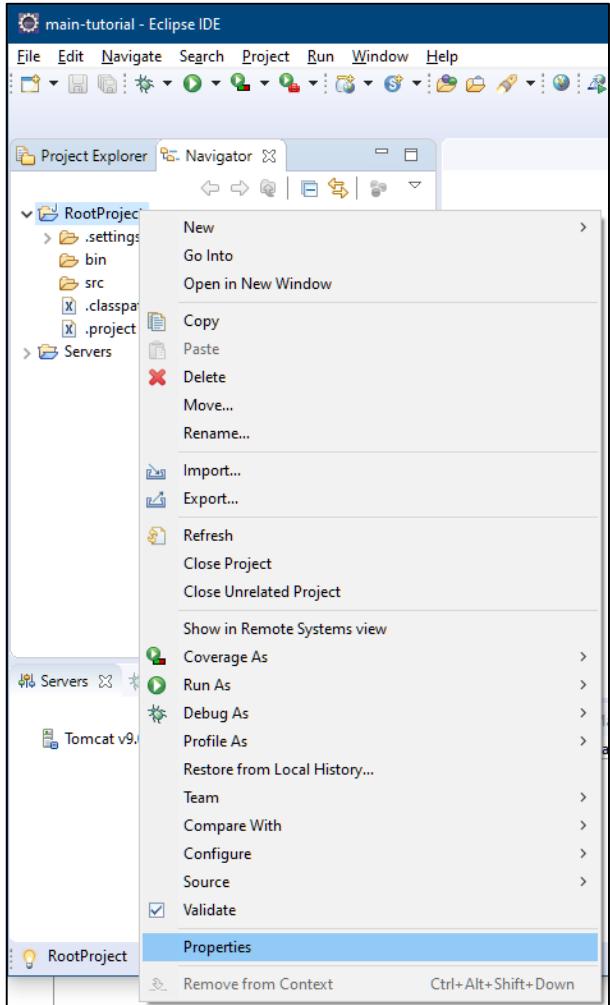
- On the “Servers” view, add a new server
- pick where you installed Tomcat



SET UP TOMCAT

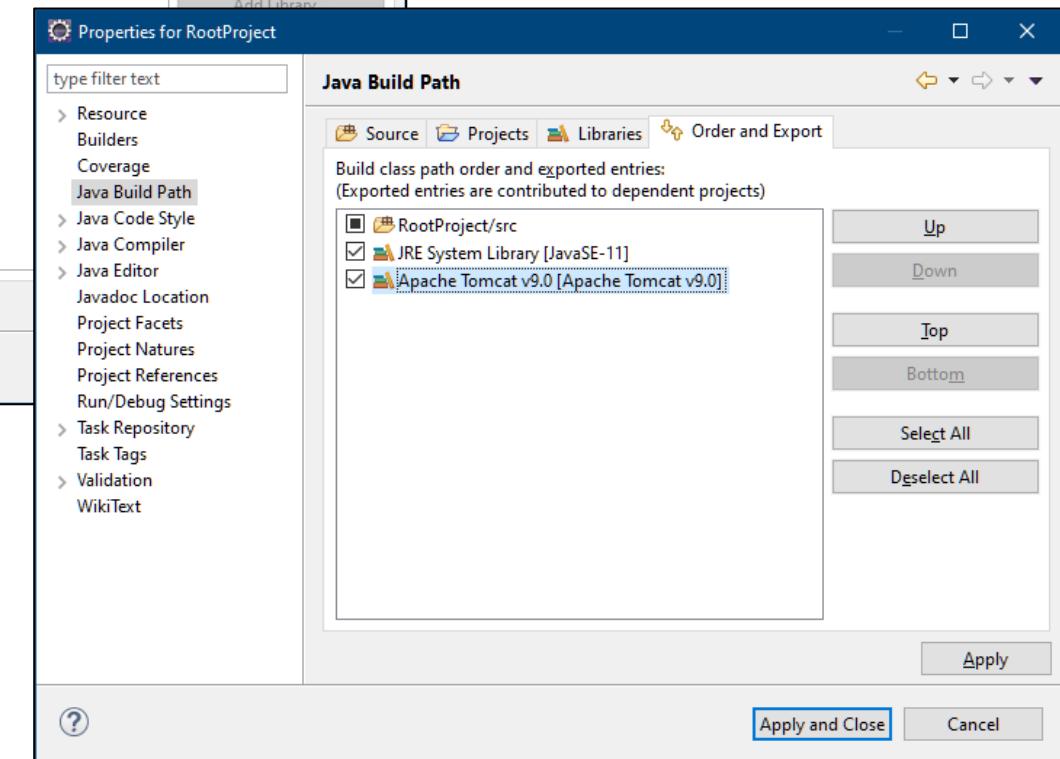
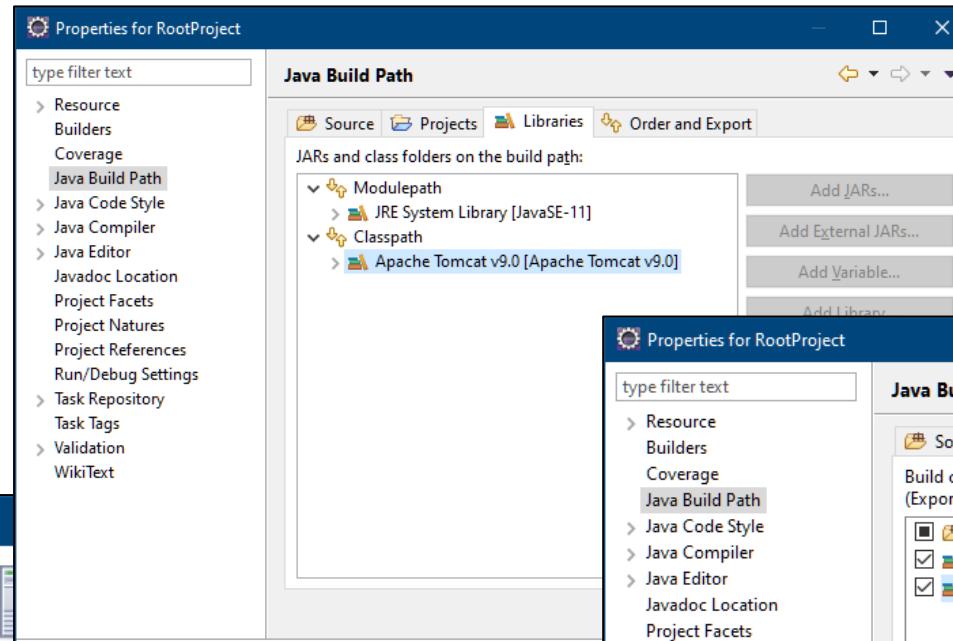
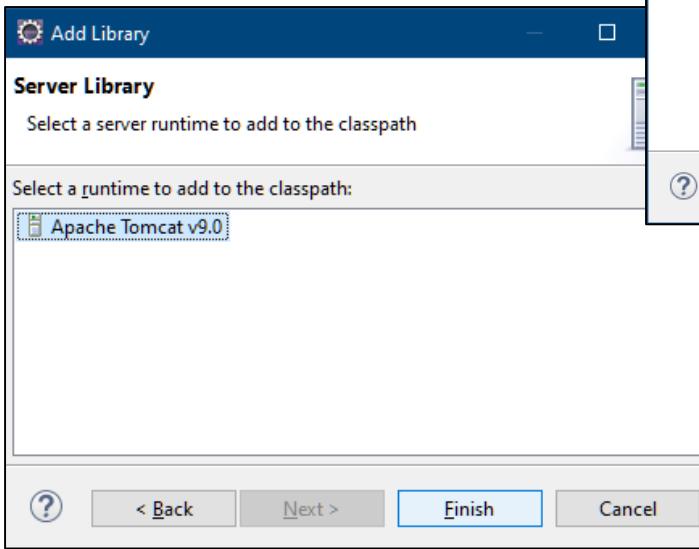
- Right click on “RootProject” and select “Properties”

- “Java Build Path”, “Libraries”, “Add Library”



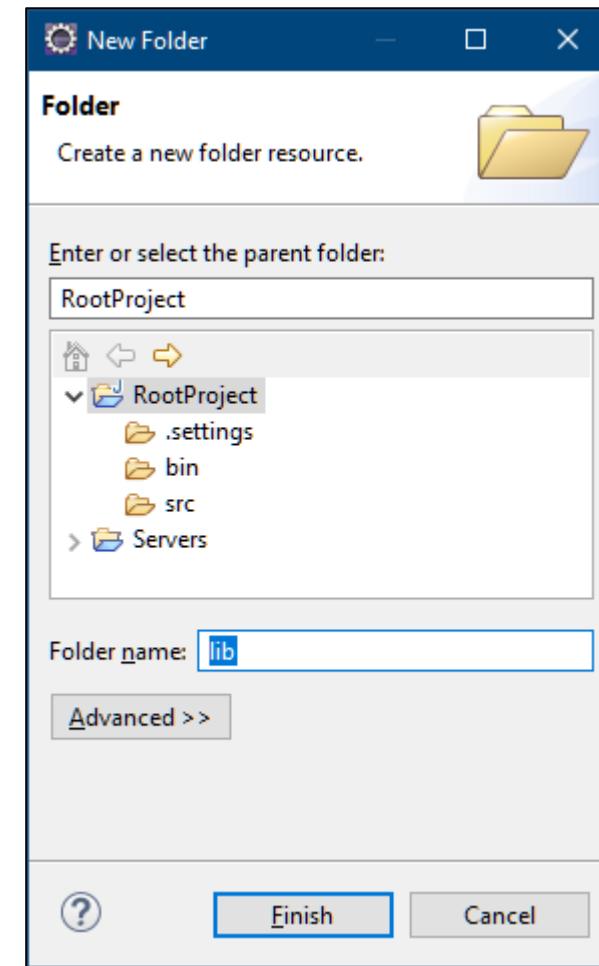
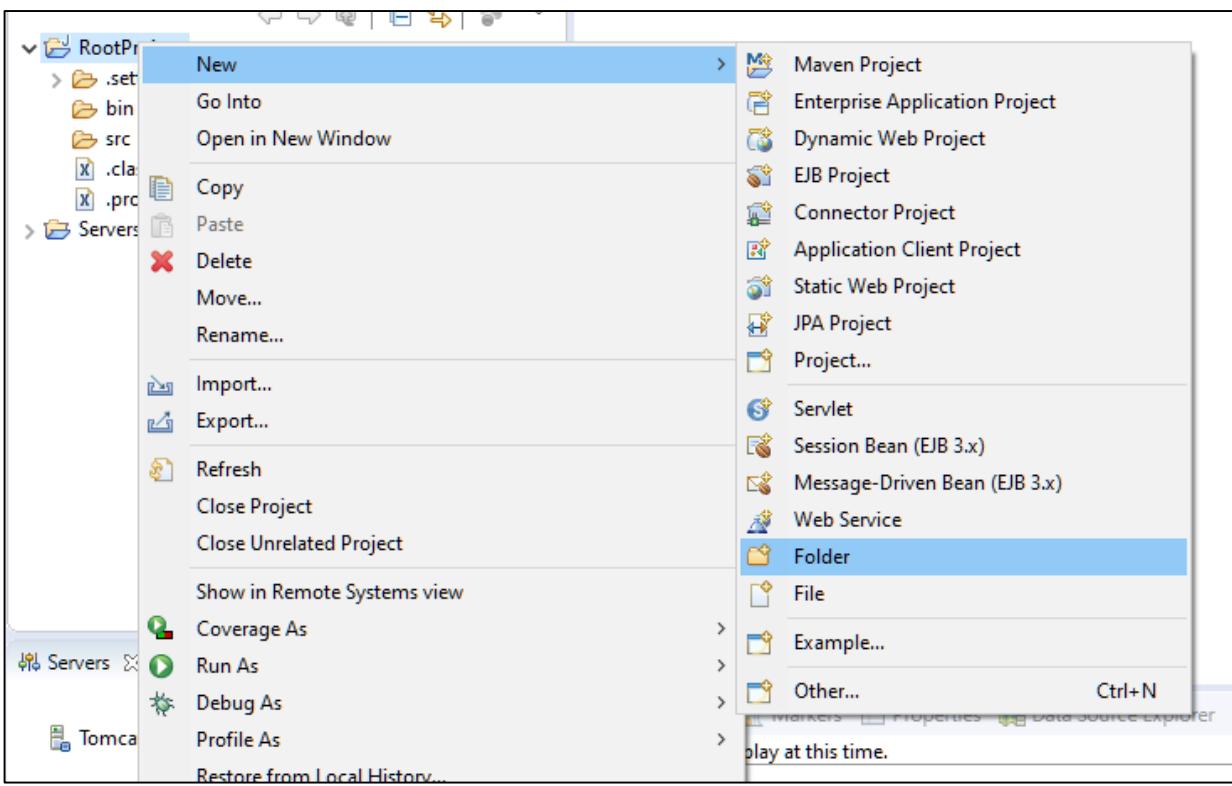
SET UP TOMCAT

- Select Tomcat 9
- Export
- Apply And Close



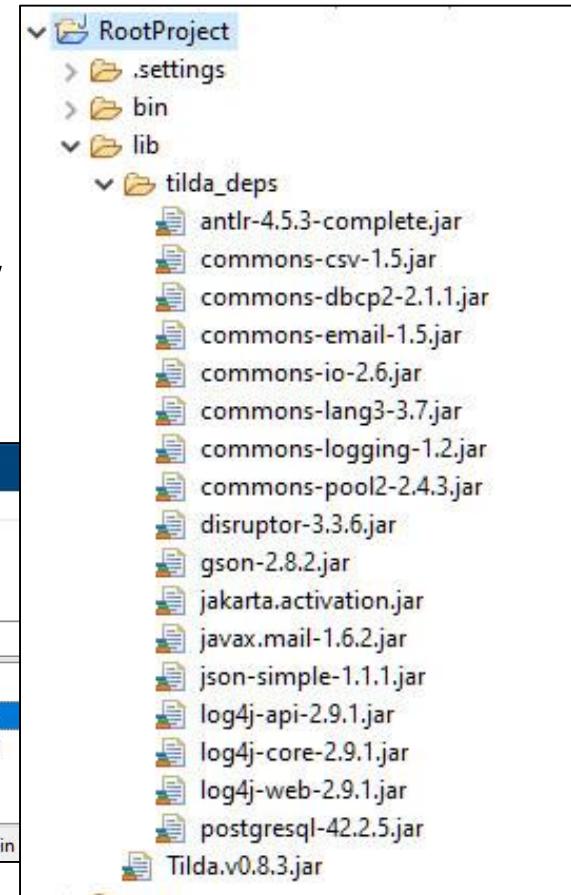
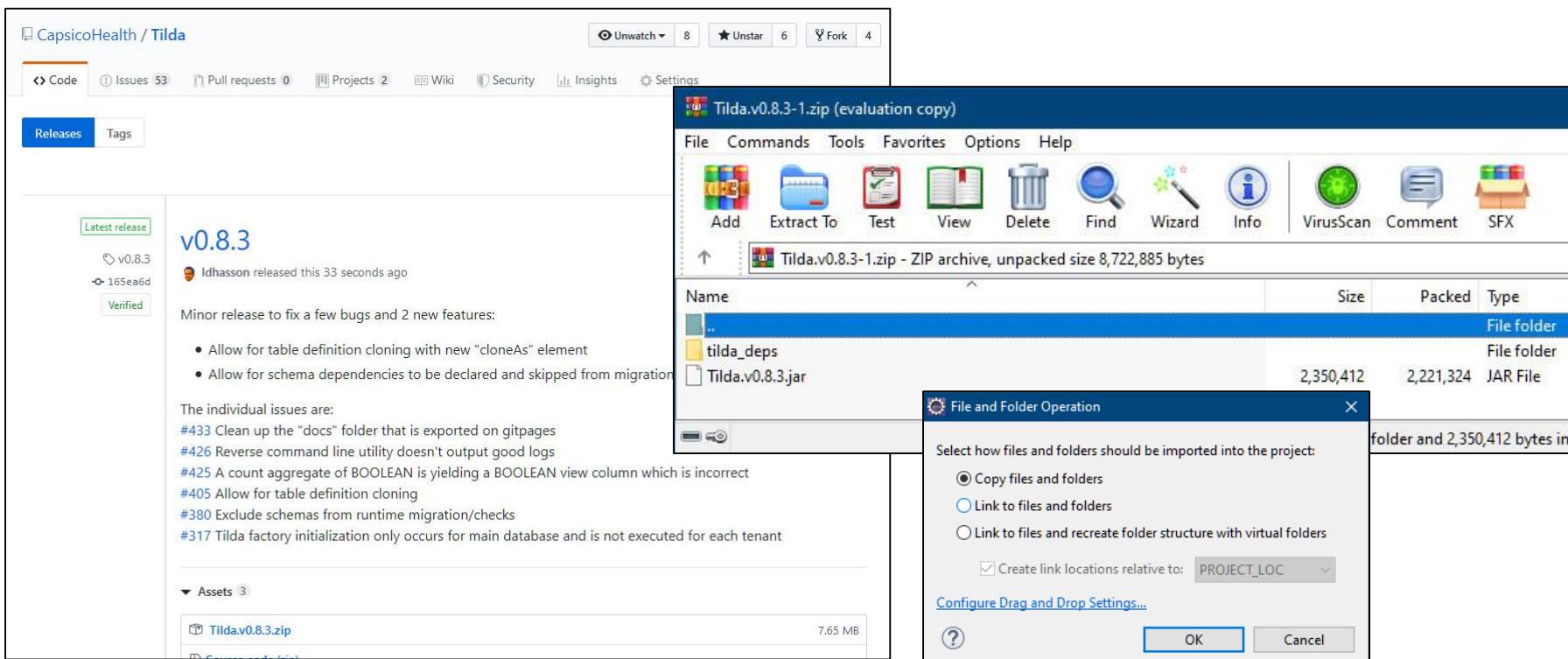
ADD THE TILDA LIBRARIES

- Add a “lib” folder to the project



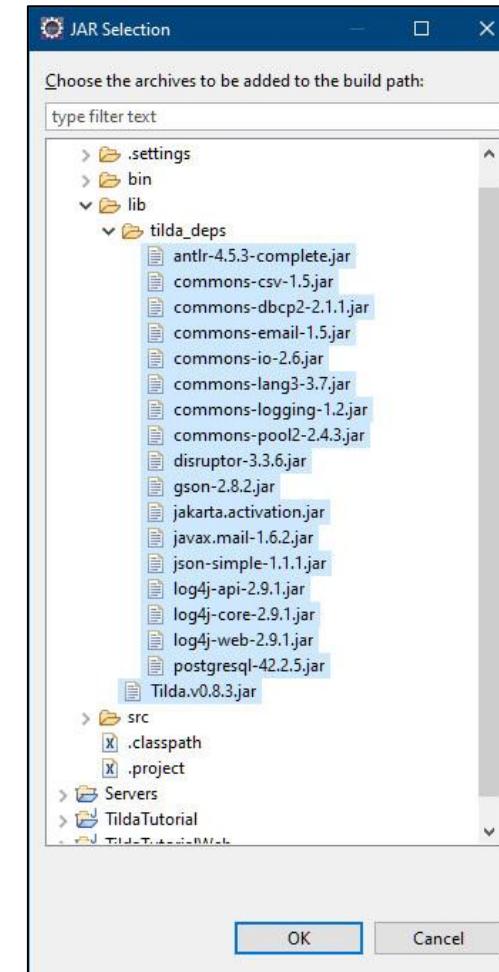
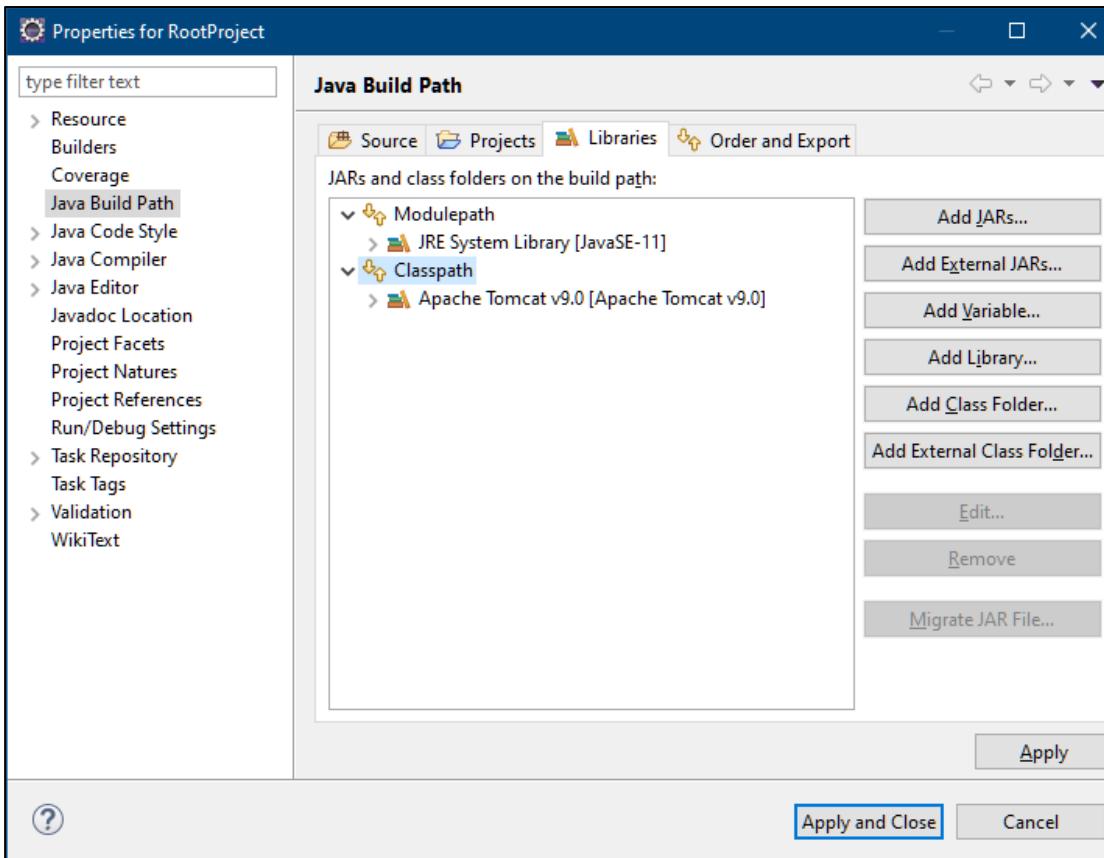
ADD THE TILDA LIBRARIES

- Download the latest Tilda release
 - <https://github.com/CapsicoHealth/Tilda/releases>
- Drag'n'Drop the contents of the Zip to the “lib” folder



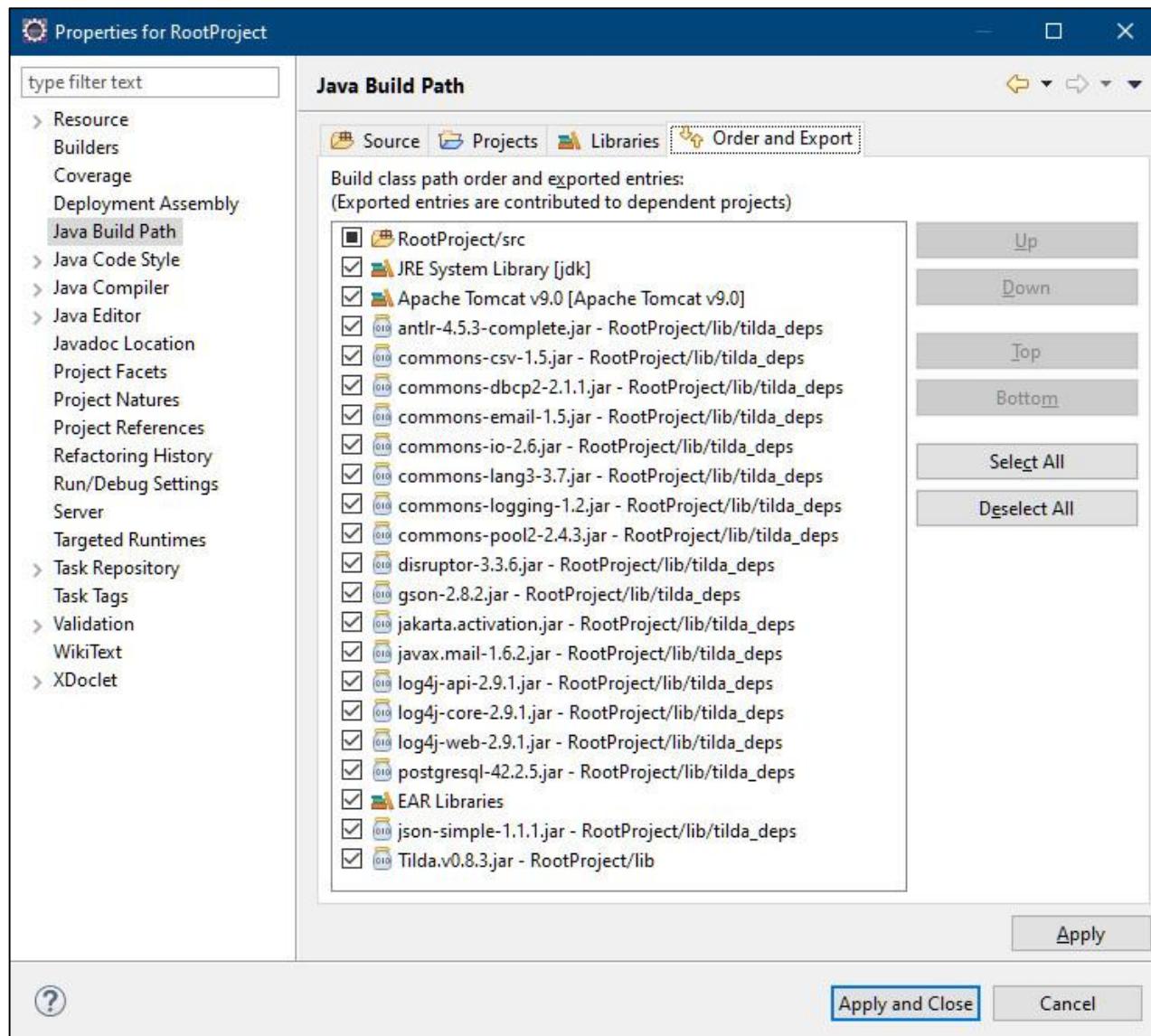
ADD THE TILDA LIBRARIES

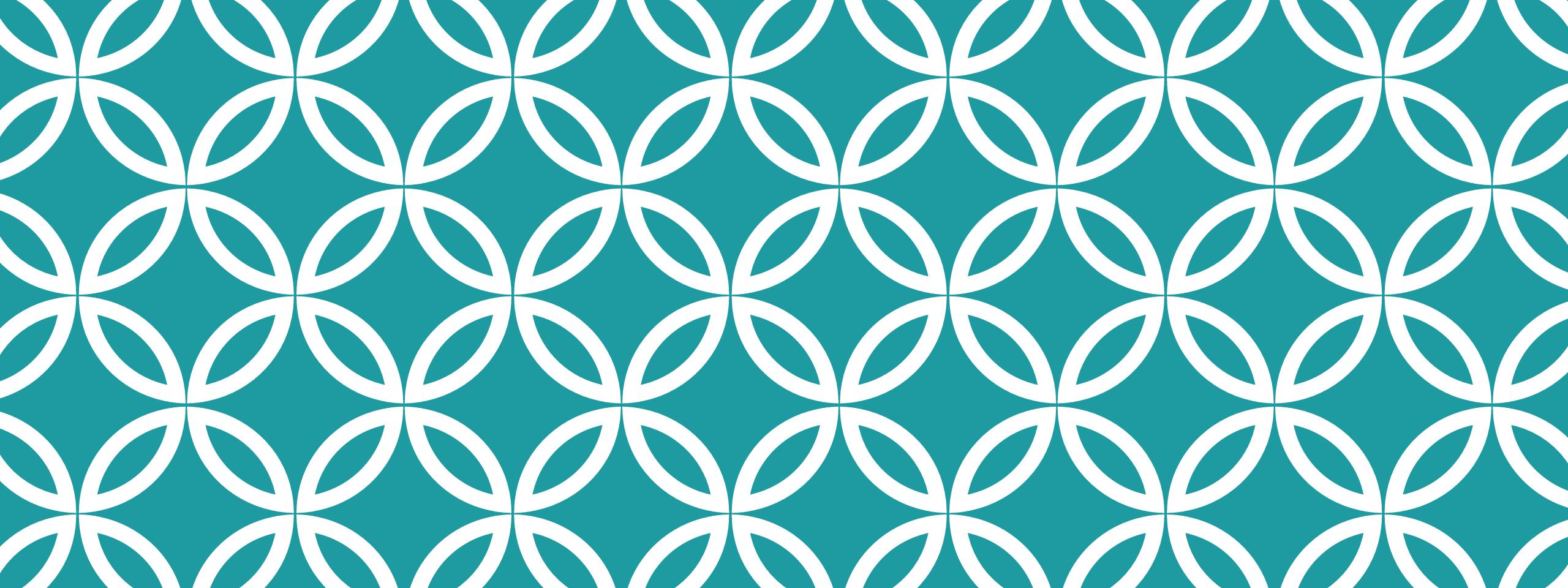
- Right click on “RootProject”, select “Properties”, “Java Build Path”
- Select “Libraries”, “Add JARs”



ADD THE TILDA LIBRARIES

- Export all
- Apply and close





FIRST TILDA PROJECT

FIRST TILDA PROJECT

- In this section, we are going to be creating a bit of infrastructure:
 - A Java project to hold our Tutorial sample
 - A Dynamic Web project to serve as our runtime baseline and which we'll build upon in a follow-up tutorial for Java-based usages of Tilda.
 - Configuration files log4j2.xml and tilda.config.json
 - Setting up “Run Configurations” for Gen and Migrate
 - Skeleton _tilda.TildaTutorial.json file
 - Running Gen
 - Creating the TildaTutorial database
 - Running Migrate
- These steps are typically a one-time affair for any new project and take a few minutes to set up.

FIRST TILDA PROJECT

- We want to set up a simple relationship hierarchy of projects where in the future:
 - It's easy to add new dependencies in the Root project that automatically percolate across the entire project
 - Each "component" of the tutorial is packaged as a JAR and we can add more in the future
 - The Web project provides an overall runtime for execution.

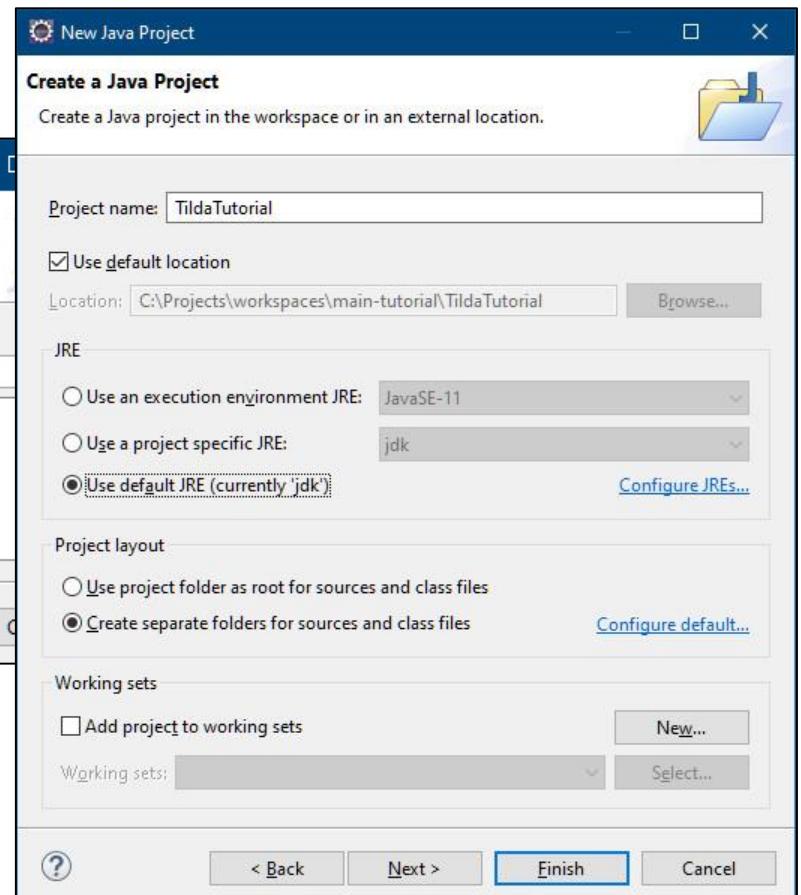
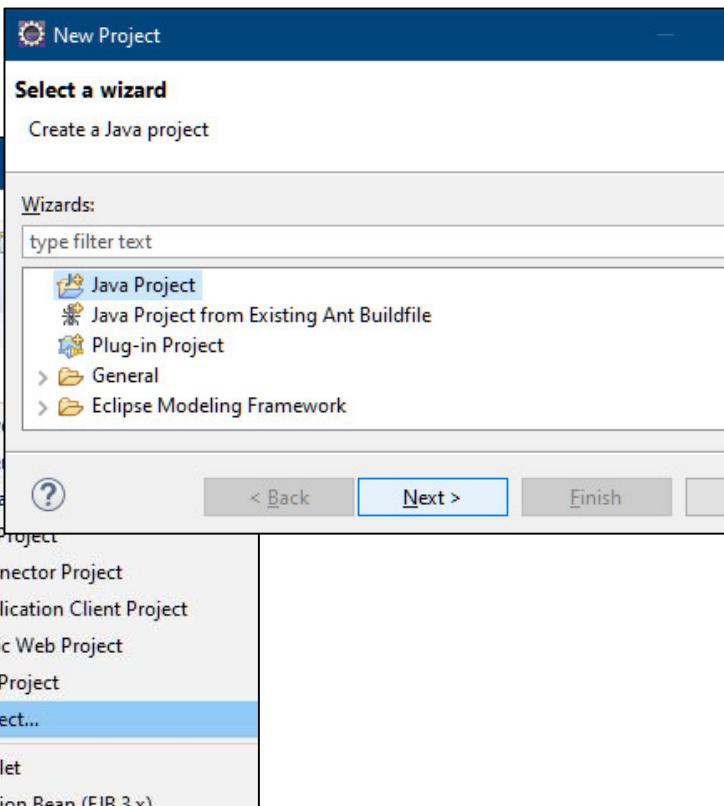
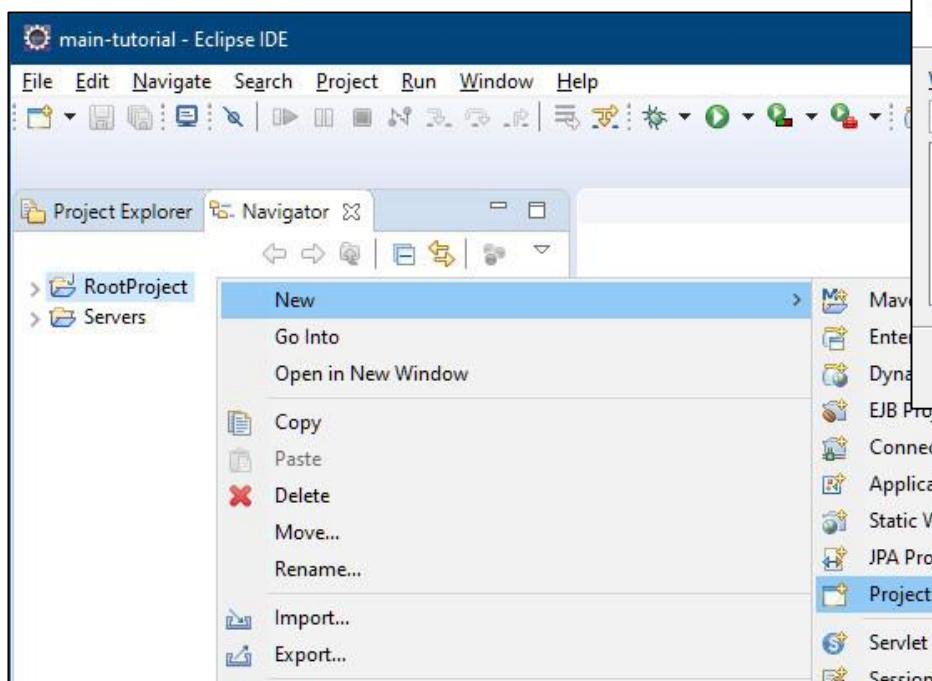


FIRST TILDA PROJECT

- Sample files can be found in
 - https://github.com/CapsicoHealth/Tilda/tree/master/docs/documentation/tutorial_sample_files
- Documentation for Gen can be found in
 - <https://github.com/CapsicoHealth/Tilda/wiki/Tilda-Command-Line-Utilities%3A-Gen>
- Documentation for Migrate can be found in
 - <https://github.com/CapsicoHealth/Tilda/wiki/Tilda-Command-Line-Utilities%3A-Migrate>

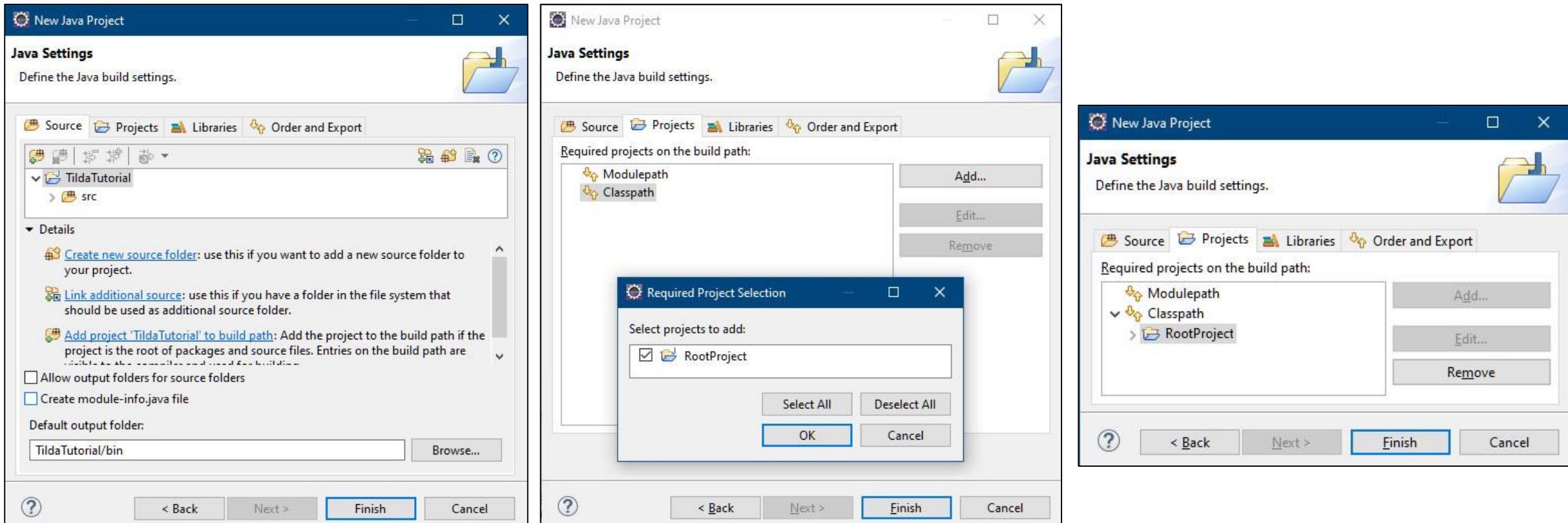
TUTORIAL PROJECT

- Create a new Java project “TildaTutorial”



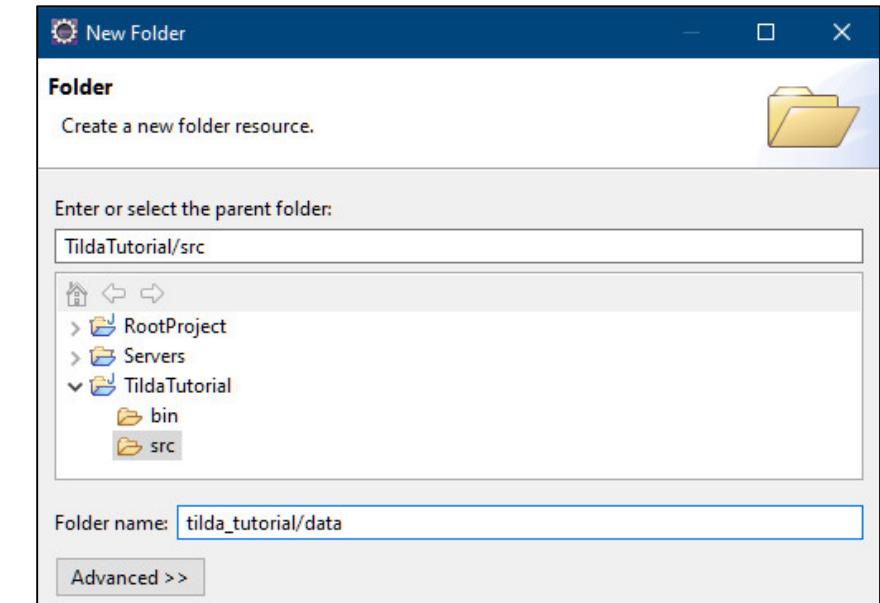
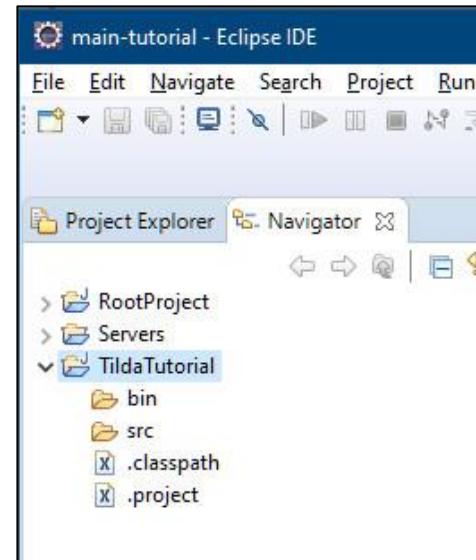
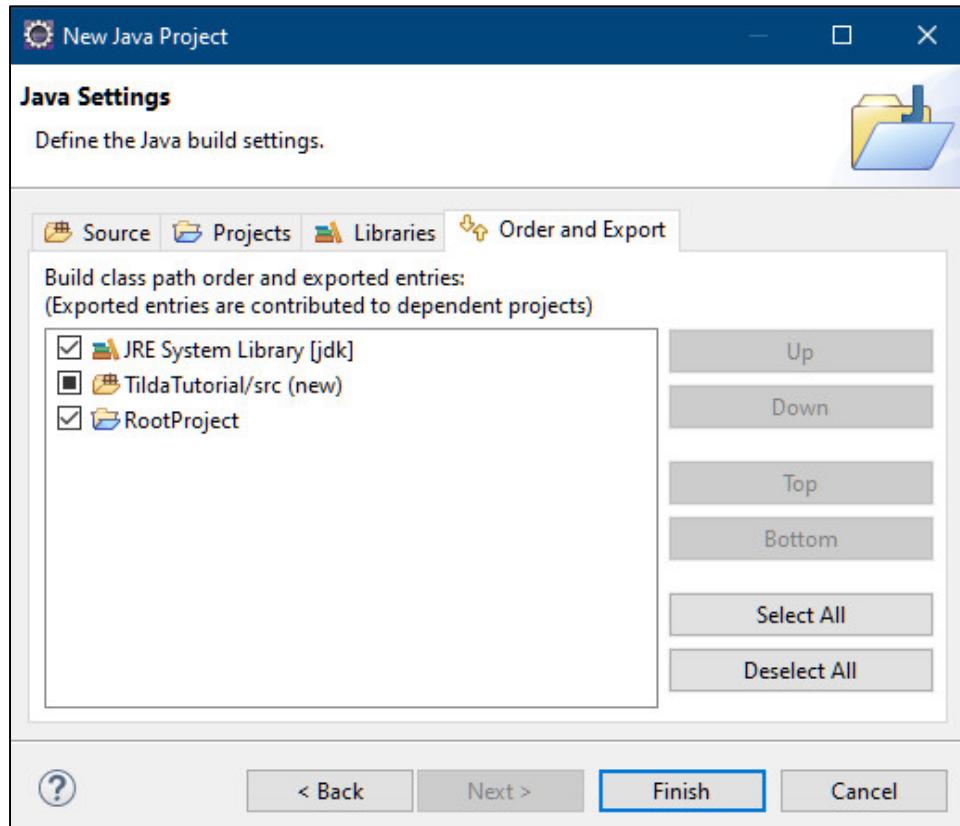
TUTORIAL PROJECT

- Make sure that “module-info.java” is unchecked
- In the Projects tab, select “Classpath” and add “RootProject”



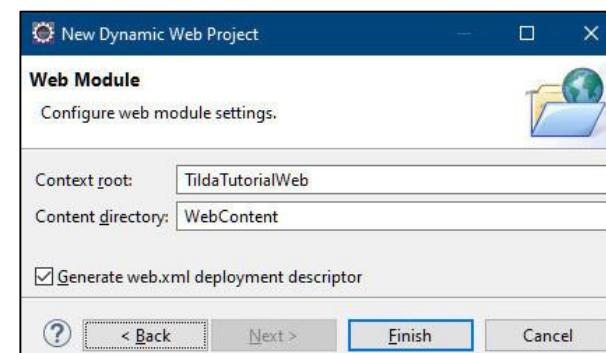
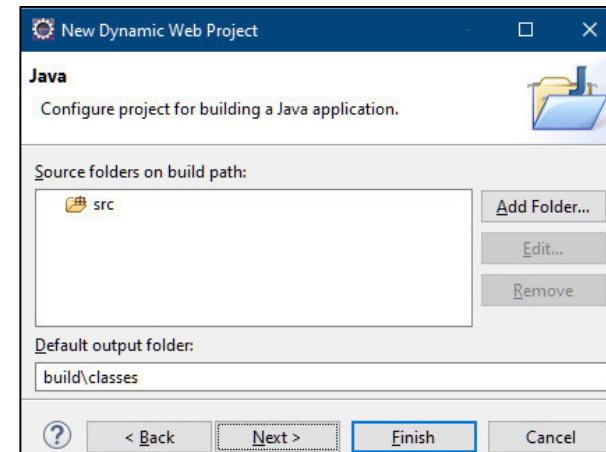
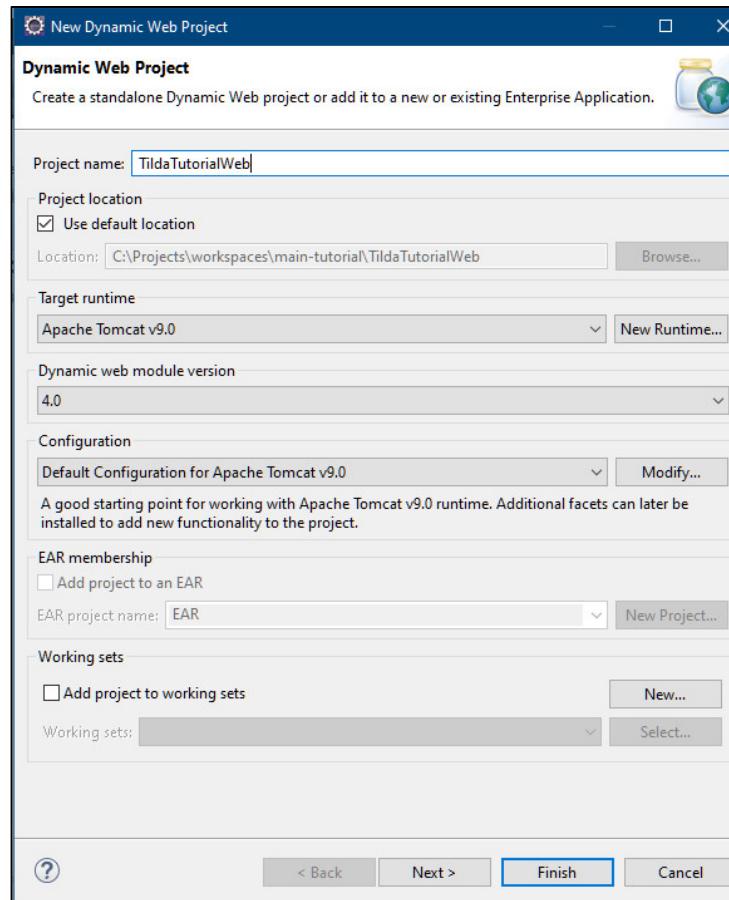
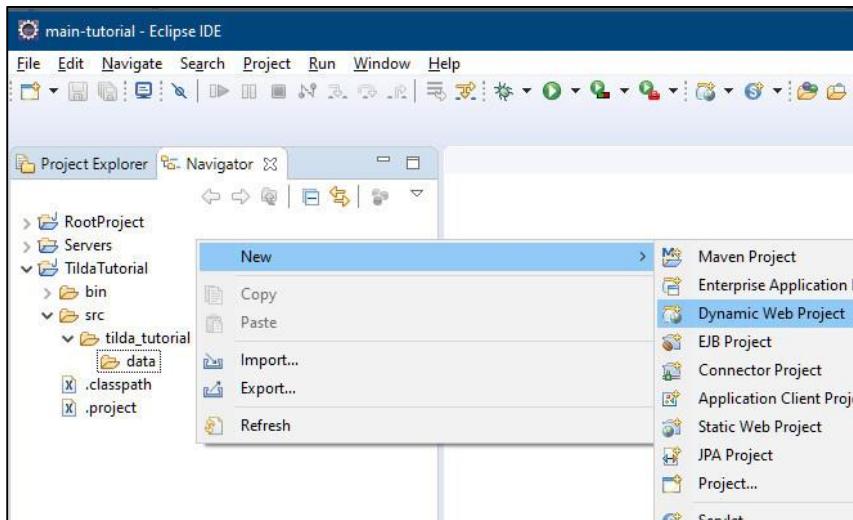
TUTORIAL PROJECT

- In the “Order and Export” tab, make sure all are exported
- Then create a folder tilda_tutorial/data



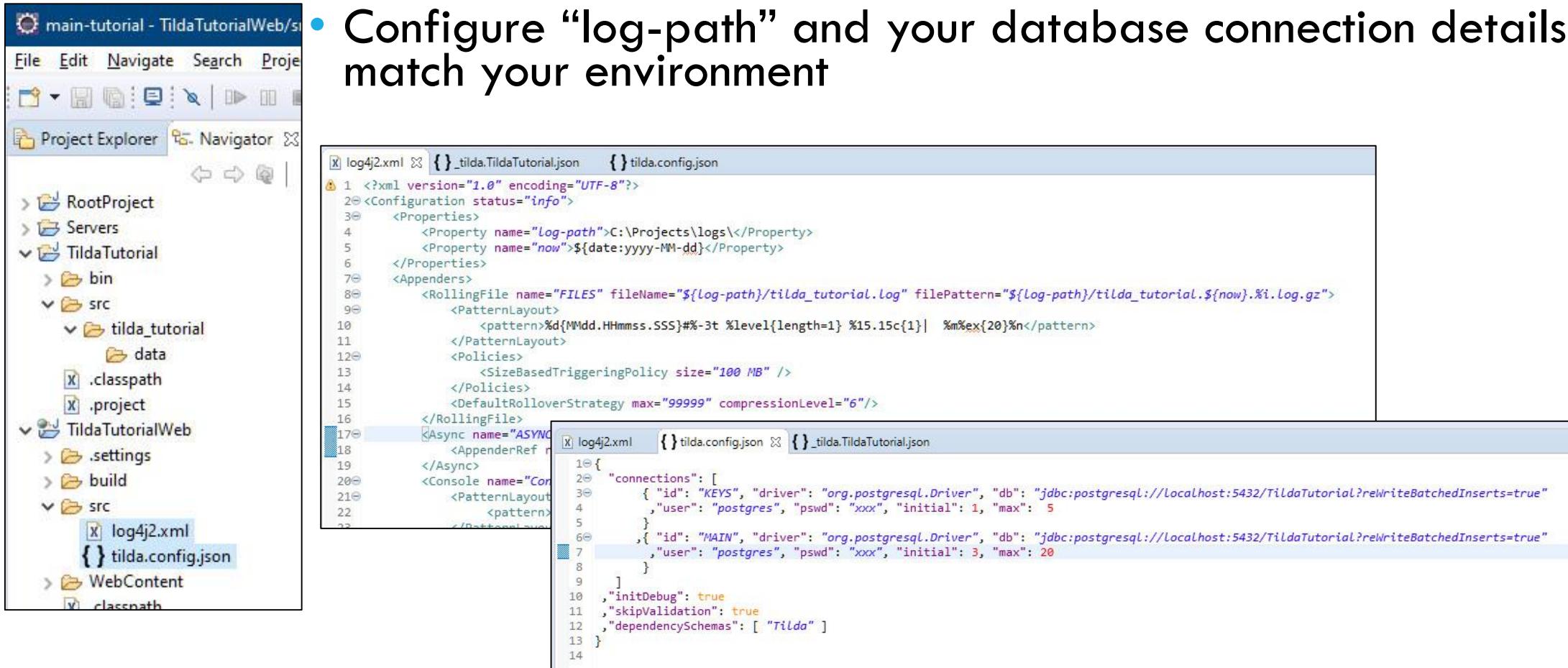
TUTORIAL WEB PROJECT

- Create a new Dynamic Web Project “TildaTutorialWeb”
- Select “Generate web.xml deployment descriptor”.



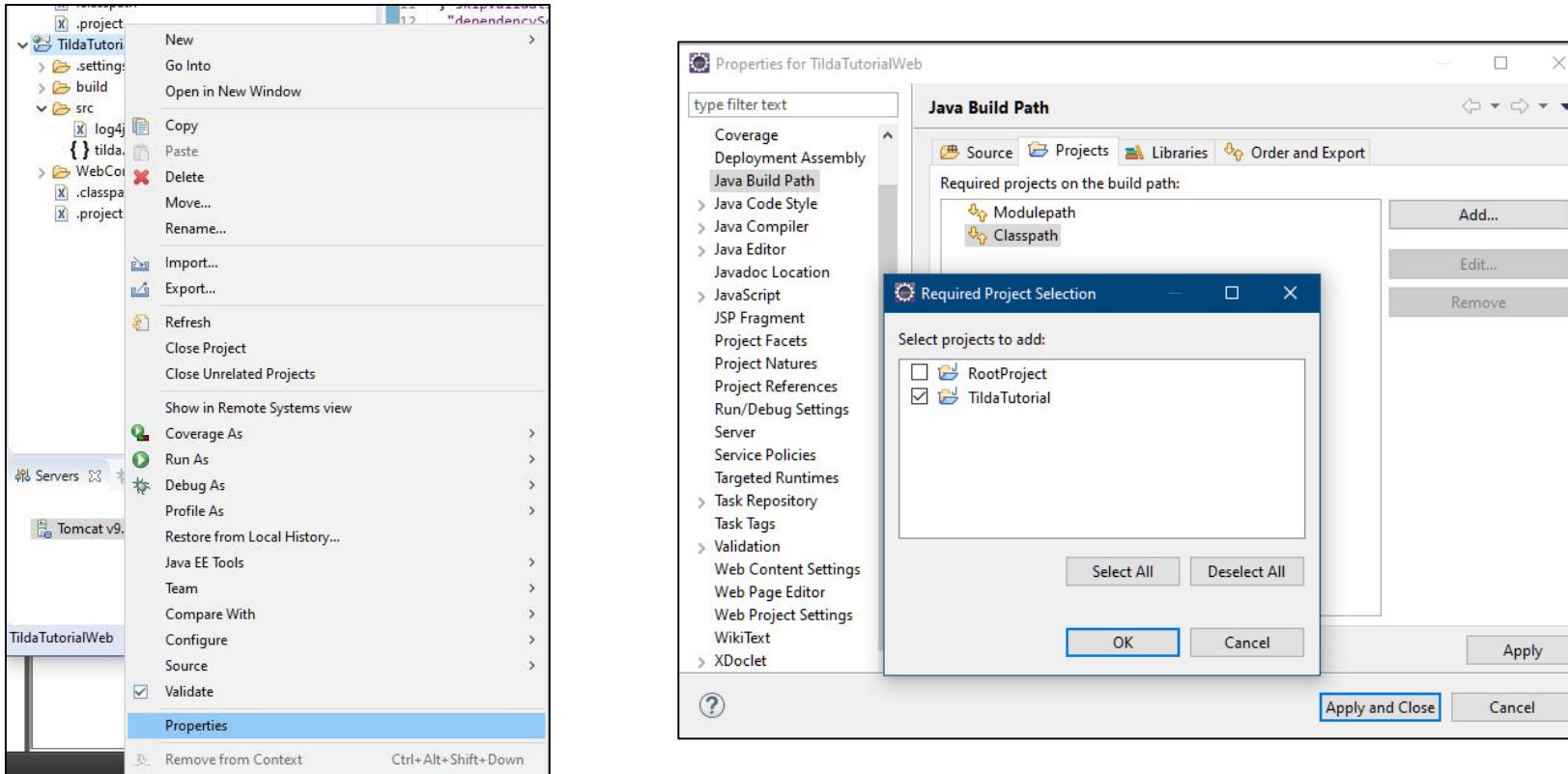
TUTORIAL WEB PROJECT

- Copy the files log4j2.xml and tilda.config.json from the sample files folder into your project's src folder
- Configure “log-path” and your database connection details to match your environment



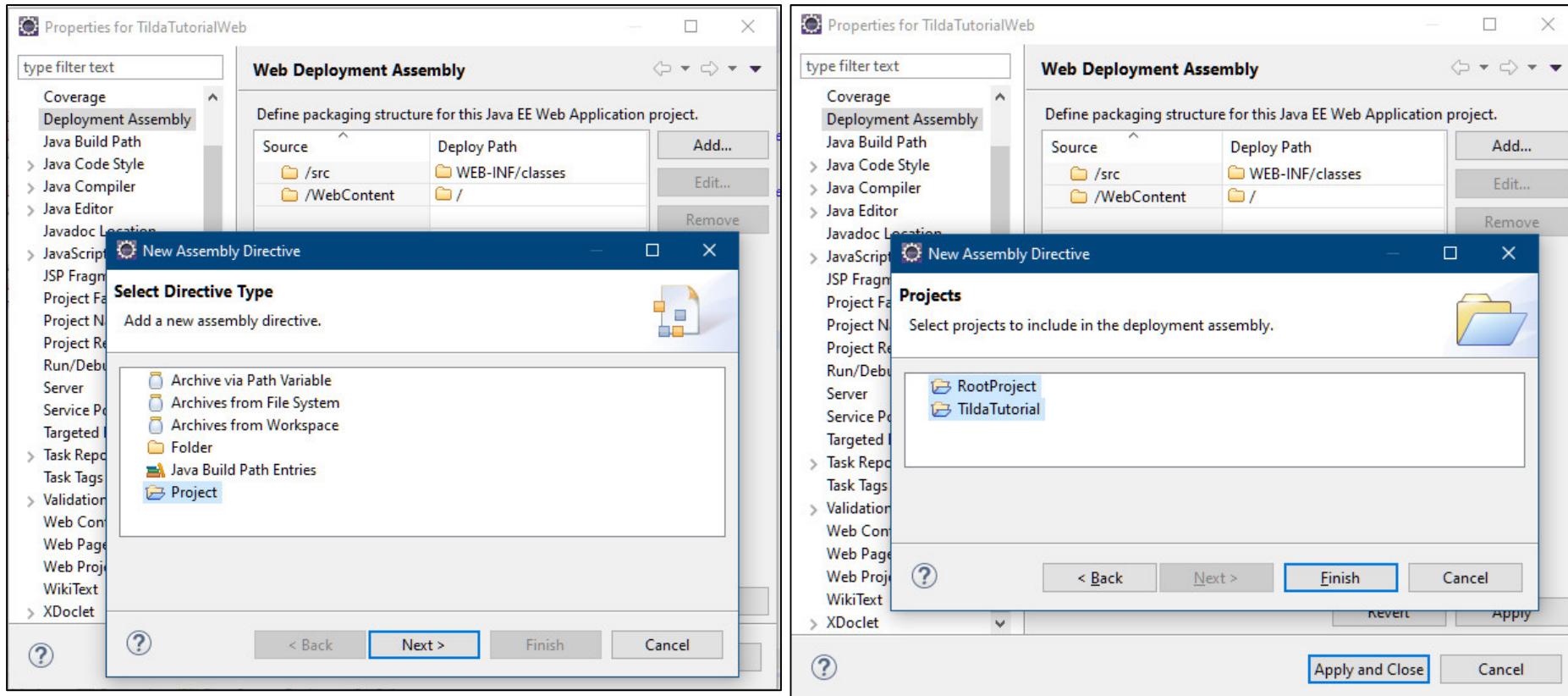
TUTORIAL WEB PROJECT

- Configure the project's java build path properties
- Select the “Project” tab and add “TildaTutorial” to the Classpath



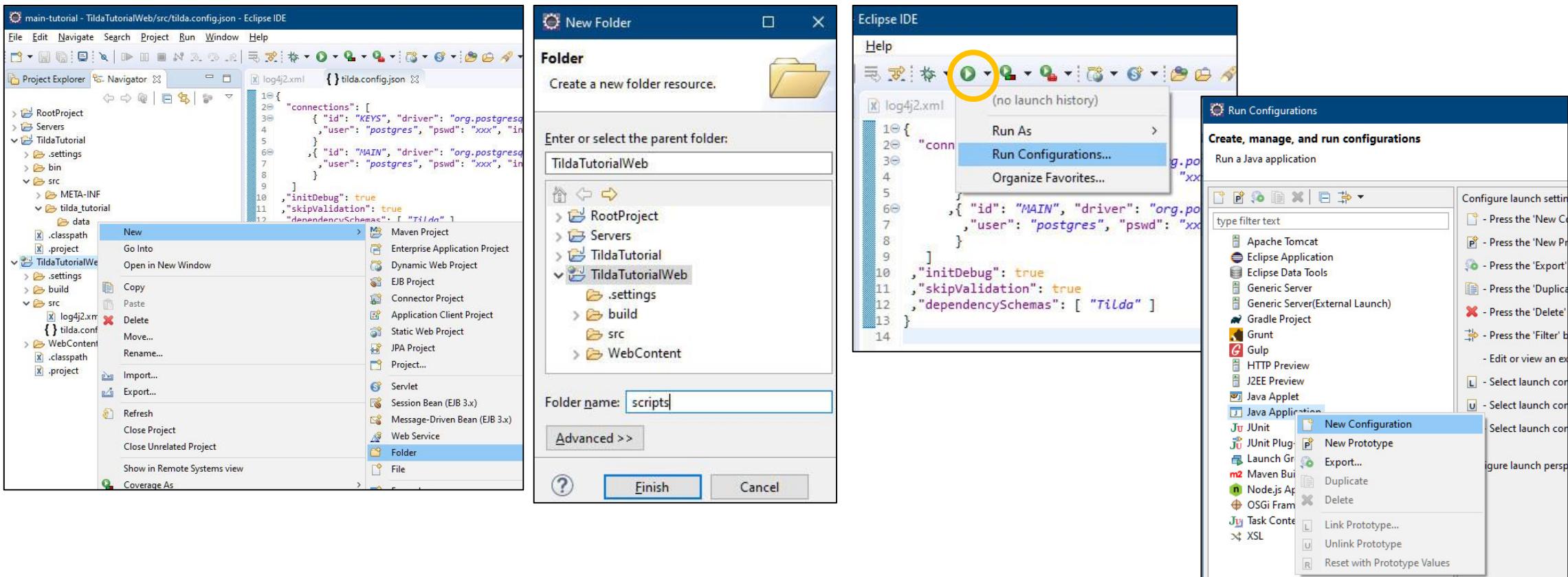
TUTORIAL WEB PROJECT

- Configure the Web Deployment Assembly details and add the 2 projects we have created previously.



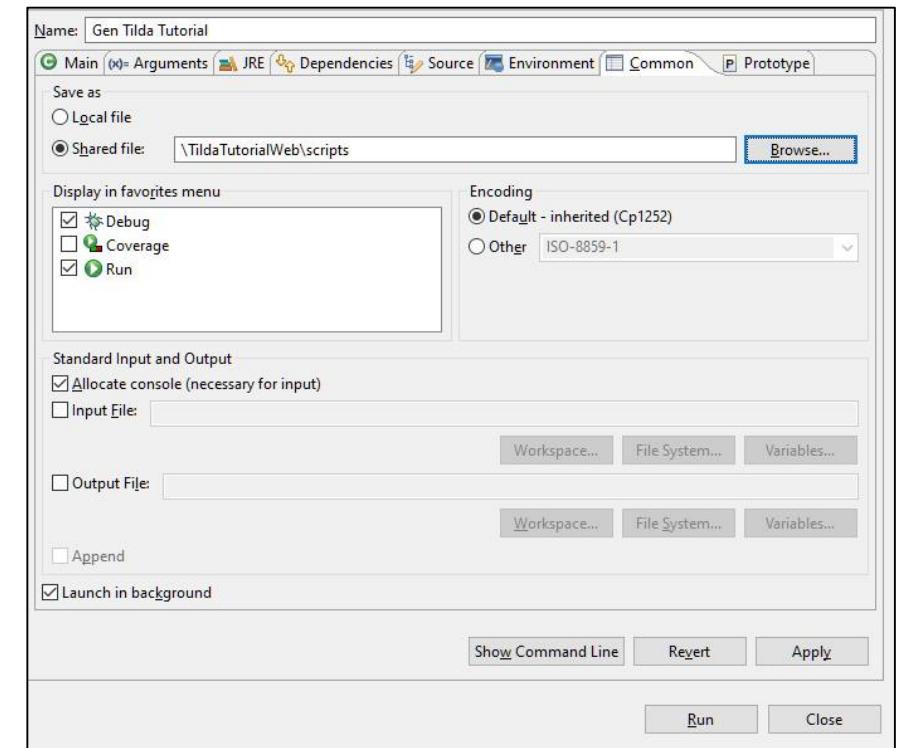
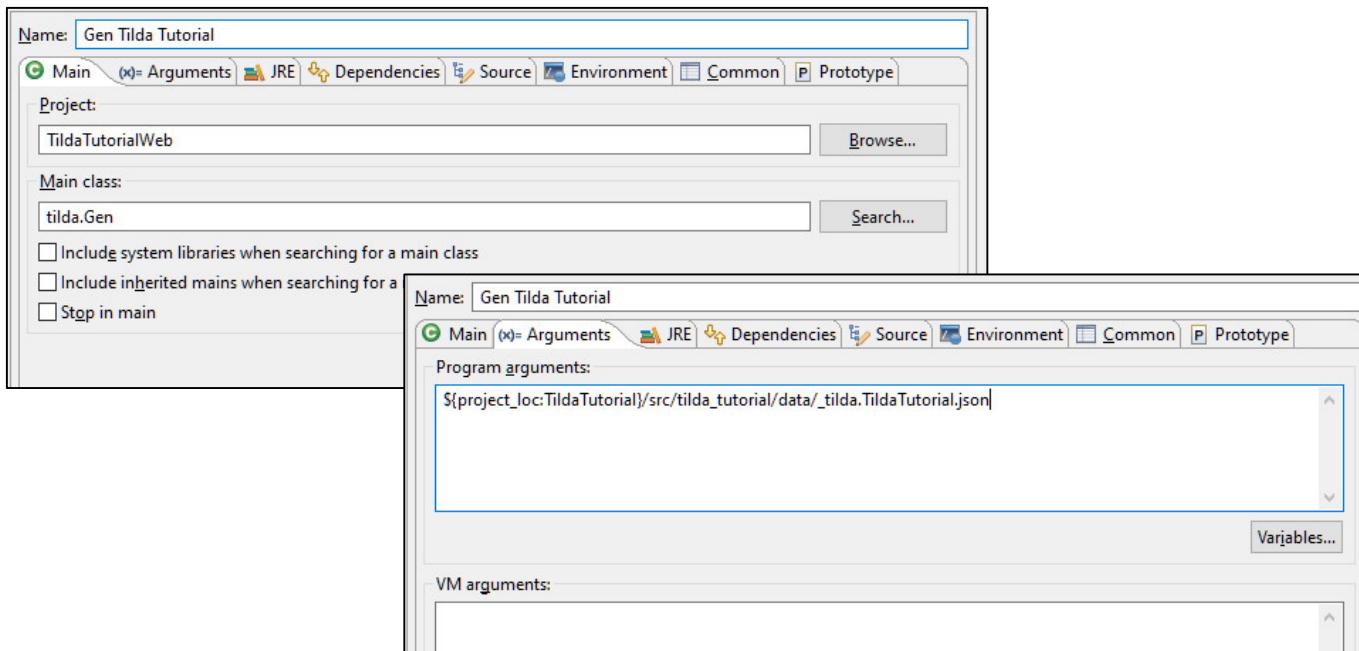
RUN CONFIGURATIONS

- Create a new folder “scripts” for the run configurations we will create.



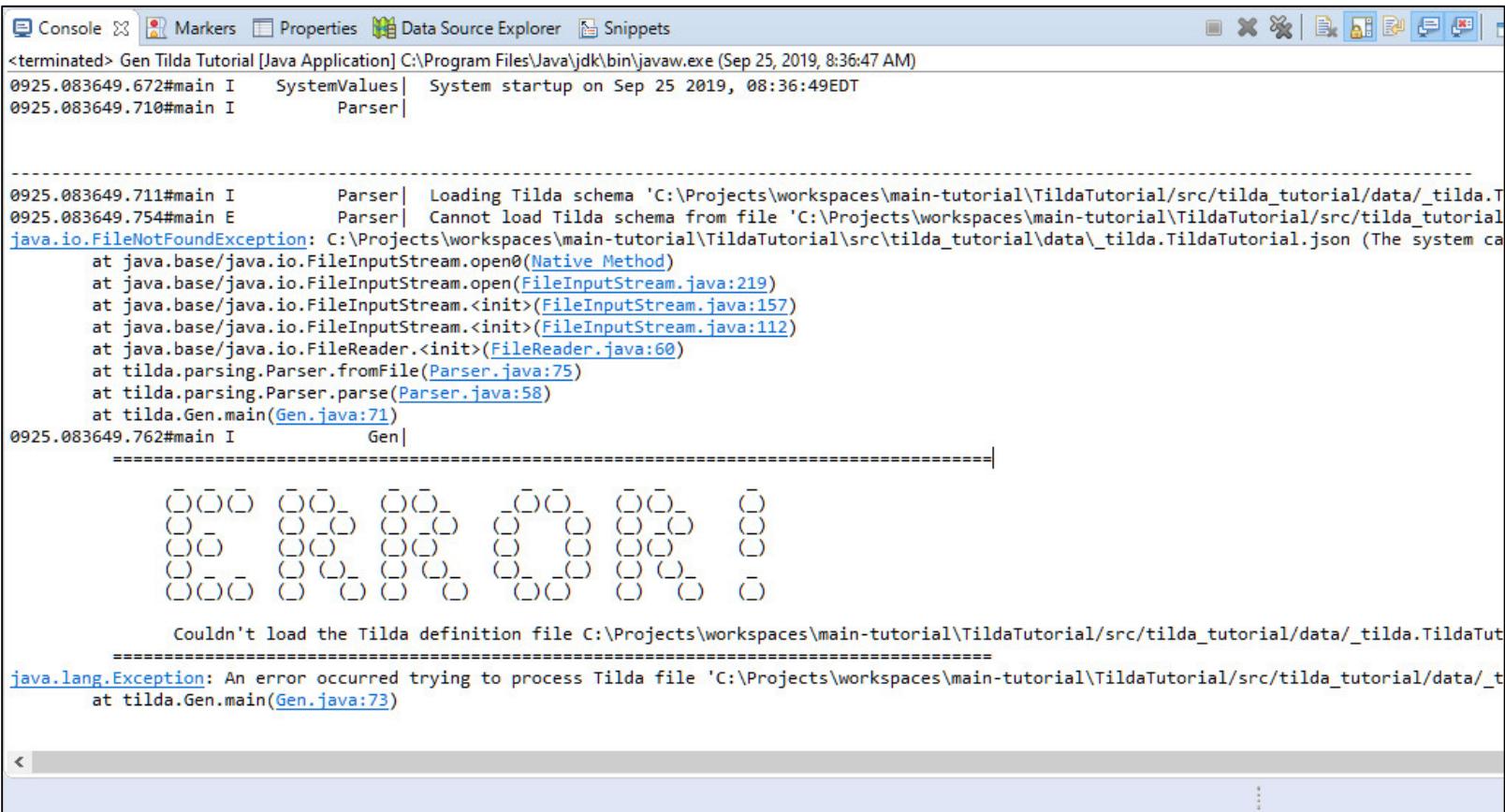
RUN CONFIGURATIONS

- Create a new run configuration for Gen called “Gen Tilda Tutorial”
- Project is “TildaTutorialWeb”, main class is “tilda.Gen”, parameters is
 - \${project_loc:TildaTutorial}/src/tilda_tutorial/data/_tilda.TildaTutorial.json
- Save to the scripts folder



RUN CONFIGURATIONS

- If you run “Gen” at this time, you will get an error since we haven’t created the Tilda definition file for the tutorial yet.



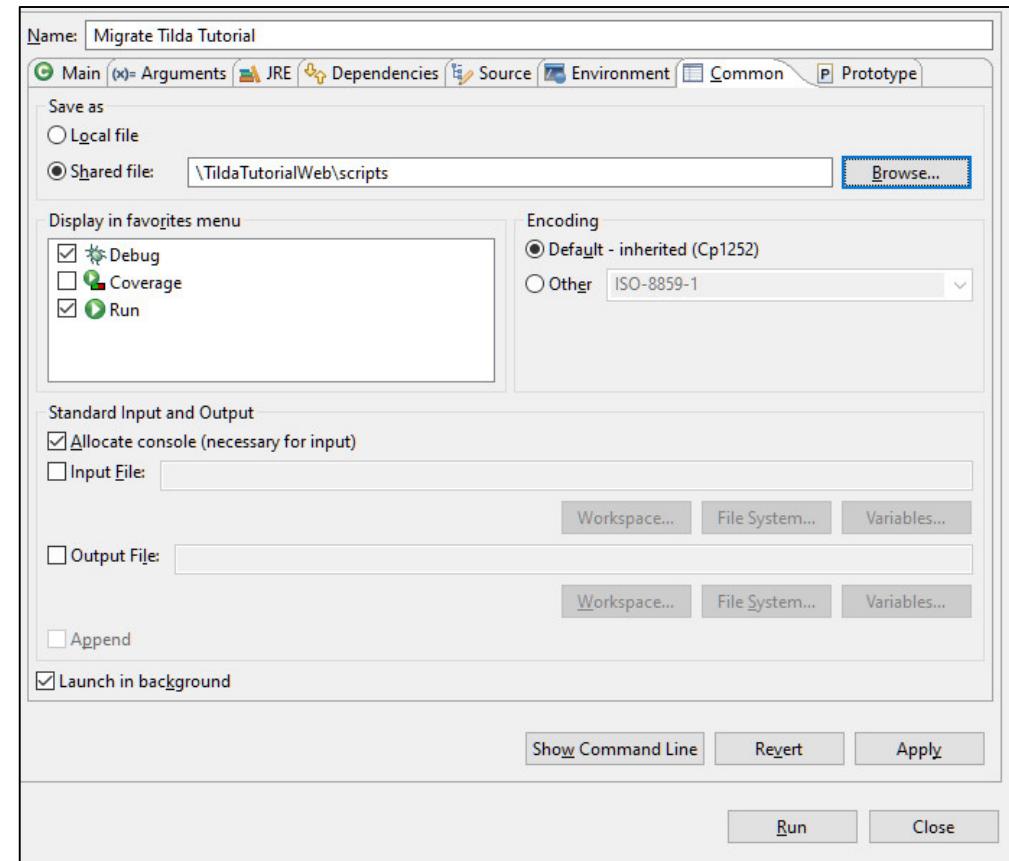
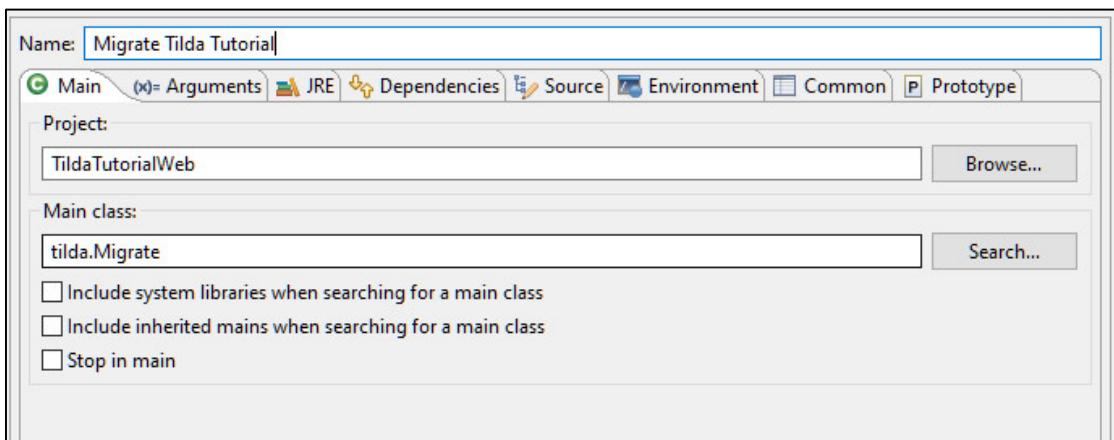
The screenshot shows a Java application console window. The title bar indicates it is a Java Application named "Gen Tilda Tutorial". The console output shows several log entries:

```
<terminated> Gen Tilda Tutorial [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe (Sep 25, 2019, 8:36:47 AM)
0925.083649.672#main I SystemValues| System startup on Sep 25 2019, 08:36:49EDT
0925.083649.710#main I Parser| 

-----
0925.083649.711#main I Parser| Loading Tilda schema 'C:\Projects\workspaces\main-tutorial\TildaTutorial/src/tildaTutorial/data/_tilda.T
0925.083649.754#main E Parser| Cannot load Tilda schema from file 'C:\Projects\workspaces\main-tutorial\TildaTutorial/src/tildaTutorial/
java.io.FileNotFoundException: C:\Projects\workspaces\main-tutorial\TildaTutorial/src/tildaTutorial\data\_tilda.TildaTutorial.json (The system ca
at java.base/java.io.FileInputStream.open0(Native Method)
at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112)
at java.base/java.io.FileReader.<init>(FileReader.java:60)
at tilda.parsing.Parser.fromFile(Parser.java:75)
at tilda.parsing.Parser.parse(Parser.java:58)
at tilda.Gen.main(Gen.java:71)
0925.083649.762#main I Gen| =====
(ಠ_ಠ)_(ಠ_ಠ)_ (ಠ_ಠ)_ (ಠ_ಠ)_ (ಠ_ಠ)_ (ಠ_ಠ)_ (ಠ_
-----|
Couldn't load the Tilda definition file C:\Projects\workspaces\main-tutorial\TildaTutorial/src/tildaTutorial/data/_tilda.TildaTutorial.json
=====
java.lang.Exception: An error occurred trying to process Tilda file 'C:\Projects\workspaces\main-tutorial\TildaTutorial/src/tildaTutorial/data/_tilda.TildaTutorial.json'
at tilda.Gen.main(Gen.java:73)
```

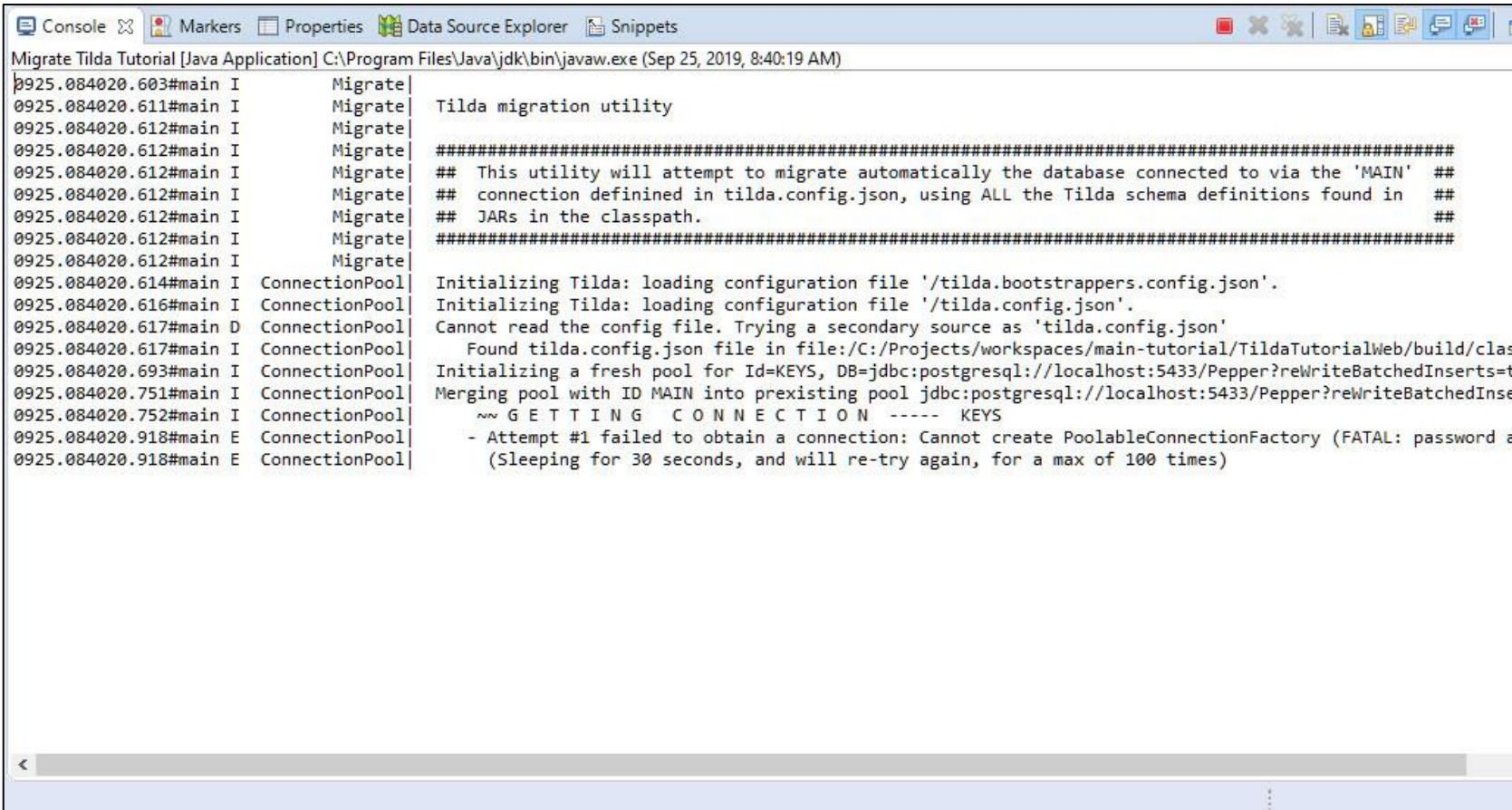
RUN CONFIGURATIONS

- Create a new run configuration for Migrate called “Migrate Tilda Tutorial”
- Project is “TildaTutorialWeb”, main class is “tilda.Migrate”, no parameters
- Save to the scripts folder



RUN CONFIGURATIONS

- If you run “Migrate” at this time, the utility will hang because we haven’t created a database yet.

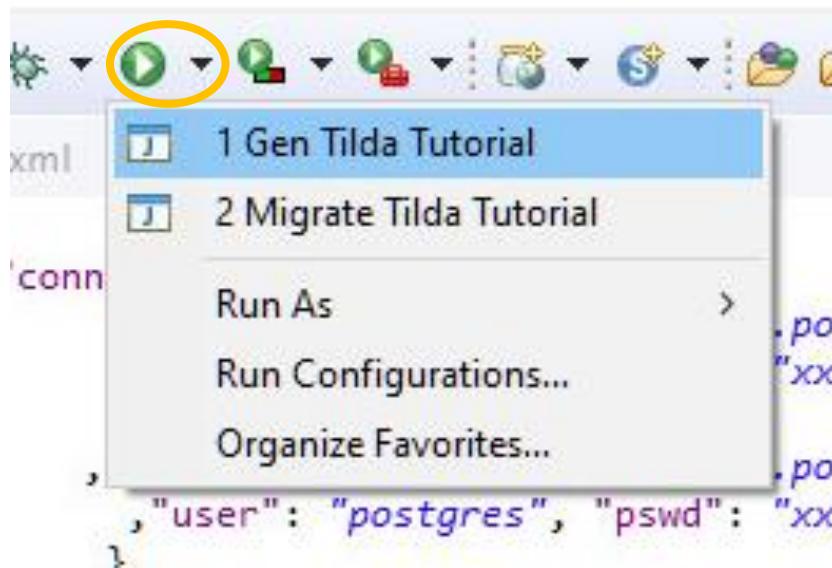


The screenshot shows a Java application console window titled "Migrate Tilda Tutorial [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe (Sep 25, 2019, 8:40:19 AM)". The window displays the following log output:

```
0925.084020.603#main I Migrate|
0925.084020.611#main I Migrate| Tilda migration utility
0925.084020.612#main I Migrate|
0925.084020.612#main I Migrate| #####
0925.084020.612#main I Migrate| ## This utility will attempt to migrate automatically the database connected to via the 'MAIN' ##
0925.084020.612#main I Migrate| ## connection defined in tilda.config.json, using ALL the Tilda schema definitions found in ##
0925.084020.612#main I Migrate| ## JARs in the classpath. ##
0925.084020.612#main I Migrate| #####
0925.084020.612#main I Migrate|
0925.084020.614#main I ConnectionPool| Initializing Tilda: loading configuration file '/tilda.bootstrappers.config.json'.
0925.084020.616#main I ConnectionPool| Initializing Tilda: loading configuration file '/tilda.config.json'.
0925.084020.617#main D ConnectionPool| Cannot read the config file. Trying a secondary source as 'tilda.config.json'
0925.084020.617#main I ConnectionPool| Found tilda.config.json file in file:/C:/Projects/workspaces/main-tutorial/TildaTutorialWeb/build/class
0925.084020.693#main I ConnectionPool| Initializing a fresh pool for Id=KEYS, DB=jdbc:postgresql://localhost:5433/Pepper?reWriteBatchedInserts=true
0925.084020.751#main I ConnectionPool| Merging pool with ID MAIN into preexisting pool jdbc:postgresql://localhost:5433/Pepper?reWriteBatchedInserts=true
0925.084020.752#main I ConnectionPool| ~~ GETTING CONNECTION ----- KEYS
0925.084020.918#main E ConnectionPool| - Attempt #1 failed to obtain a connection: Cannot create PoolableConnectionFactory (FATAL: password authentication failed for user "pepper")
0925.084020.918#main E ConnectionPool| (Sleeping for 30 seconds, and will re-try again, for a max of 100 times)
```

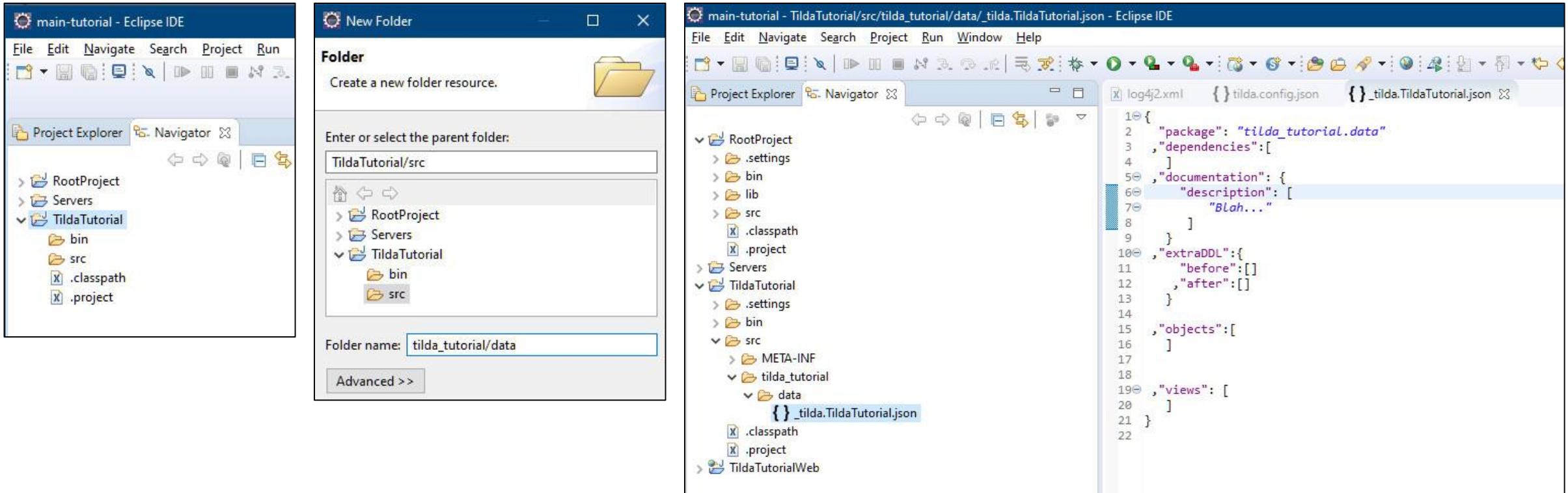
RUN CONFIGURATIONS

- From now on, the 2 run configurations are available through the shortcuts off the menu



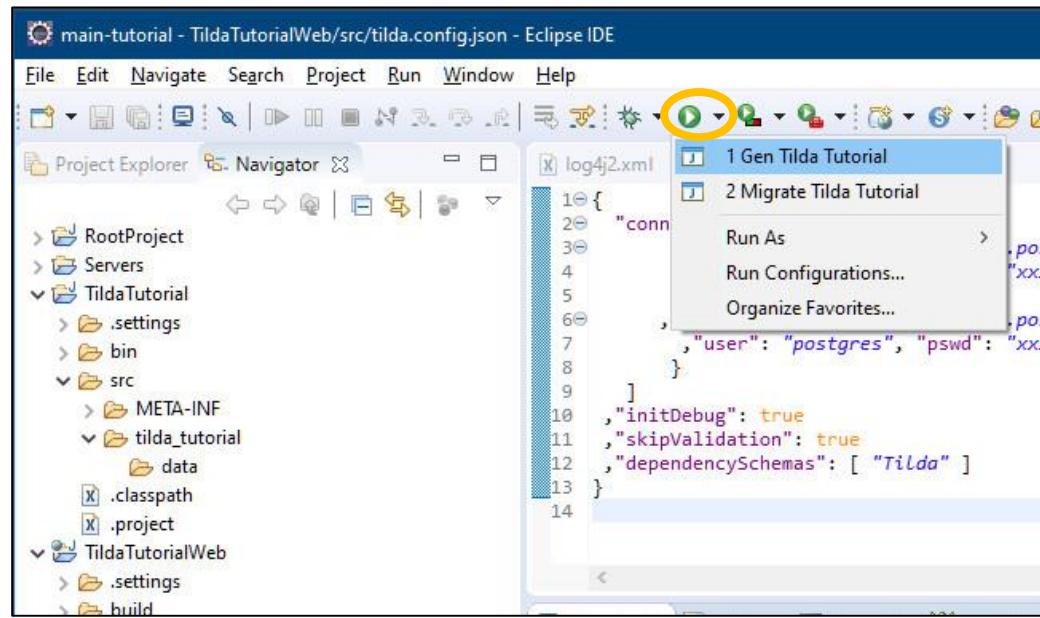
TILDA DEFINITION FILE

- Create your project's folder structure as “/tildaTutorial/data”
- Copy the file _tilda.TildaTutorial.json from the samples file folder



TILDA DEFINITION FILE

- Run Gen and get a Woohoo!

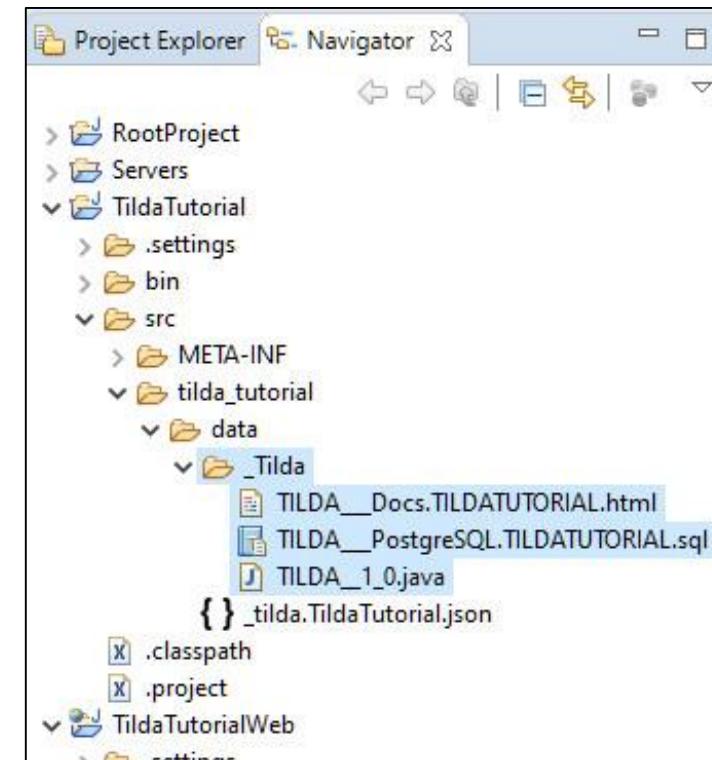
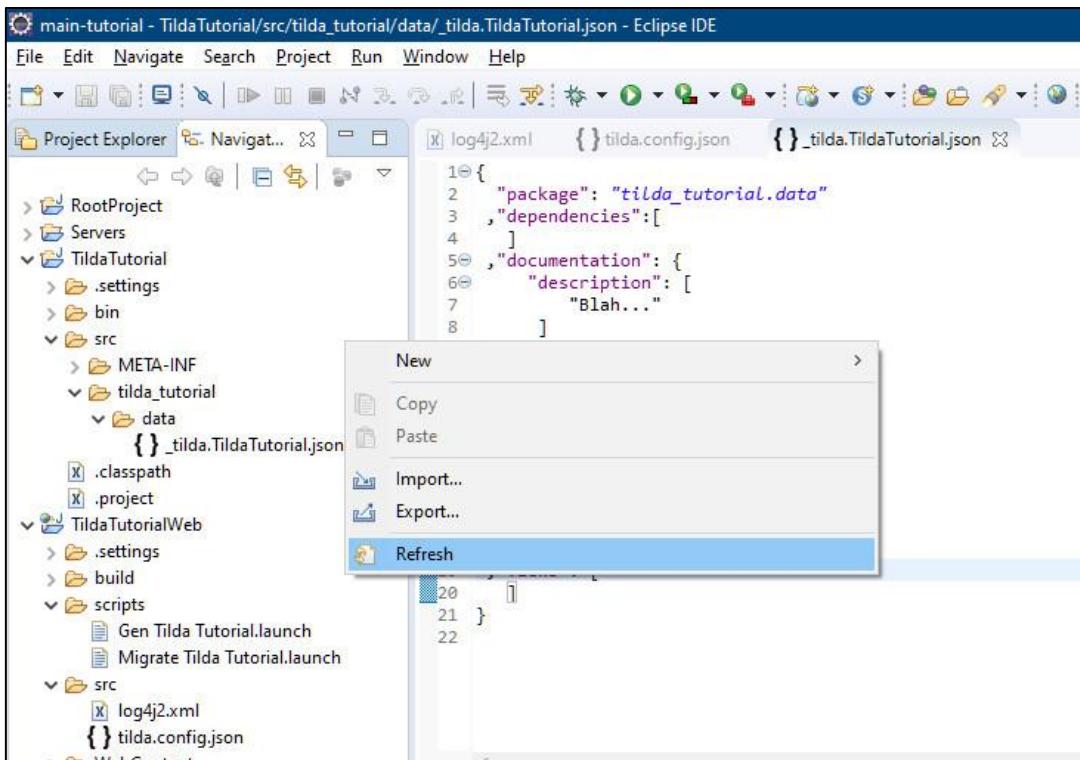


```
<terminated> Gen Tilda Tutorial [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe (Sep 25, 2019, 8:56:53 AM)
0925.085655.546#main D Object| Validating OBJECT tilda.data.TILDA.Tekting4View.
0925.085655.552#main D Object| Validating OBJECT tilda.data.TILDA.Testing4Realized.
0925.085655.554#main D ParserSession| Schema tildaTutorial has not been found in the schema dependency list.
0925.085655.554#main I Schema| Validating Tilda Schema 'tildaTutorial'.
0925.085655.554#main I Gen| Loaded Tilda schema 'tildaTutorial'.
0925.085655.555#main I Gen| Loaded and validated dependencies:
0925.085655.555#main D Gen| - tilda.data.TILDA.
0925.085655.555#main D Gen| - tildaTutorial.data.TILDATUTORIAL.
0925.085655.556#main I Parser| Generating Tilda code for Schema 'tildaTutorial'.
0925.085655.560#main D Parser| Generating the SQL file.
0925.085655.562#main D Parser| Generating base generic Tilda support class.
0925.085655.584#main I Hello| Generating dot file for TILDATUTORIAL.
0925.085655.593#main I Hello| Generating dot file for TILDATUTORIAL done.
0925.085655.593#main I Hello| Generating schema svg file for TILDATUTORIAL.
0925.085655.595#main W Hello| Unable to find the Graphviz 'dot' binary: the documentation for Tilda schemas won't include any ER Diagram.
0925.085655.595#main W Hello| To set up the dependent Graphviz library, please follow the following steps:
0925.085655.595#main W Hello| - Go to http://www.graphviz.org/Download.php
0925.085655.595#main W Hello| - Accept the license
0925.085655.596#main W Hello| - Download the package appropriate for your platform (i.e., Mac, Windows, Linux) the license
0925.085655.596#main W Hello| - Configure your tilda.config.json and add the following element
0925.085655.596#main W Hello|   . "dotBinary": "C:\Program Files (x86)\GraphViz\bin\dot.exe"
0925.085655.596#main I Hello| Deleting C:/Projects/workspaces/main-tutorial/TildaTutorial/src\tildaTutorial\data\_Tilda\TILDA__Docs.TILDATUTORIAL.dot.
0925.085655.599#main I Hello| --> Deleted C:/Projects/workspaces/main-tutorial/TildaTutorial/src\tildaTutorial\data\_Tilda\TILDA__Docs.TILDATUTORIAL.
0925.085655.600#main I Gen| Generated Tilda code for schema 'tildaTutorial'.
0925.085655.600#main I Gen| Generated Tilda code for schema 'tildaTutorial'.
```

All Tilda code, migration scripts and documentation was generated successfully.

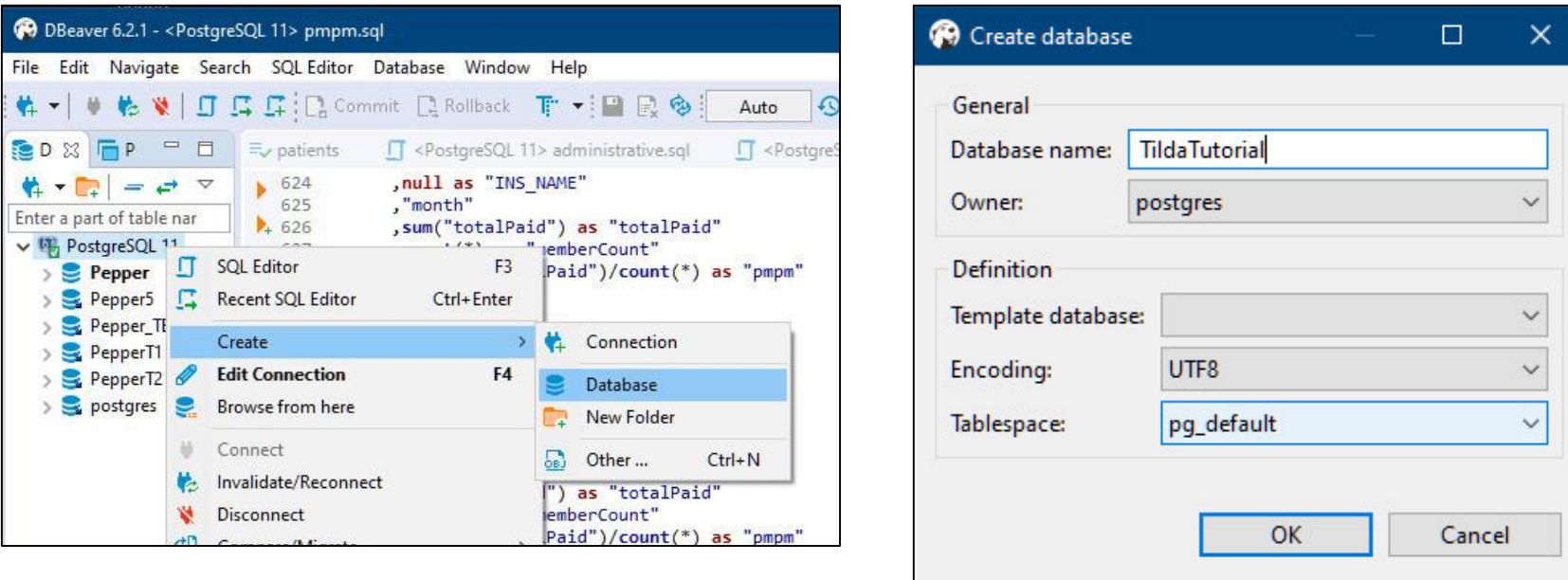
TILDA DEFINITION FILE

- Once complete, refresh your workspace so generated files are picked up.
 - Make sure no node is selected in the Navigator. Simply click on any node and then Ctrl-Click it again to unselect it
 - Right click on a white space in the navigator and select “Refresh”
 - Every Gen requires a Refresh
- You can see a new _Tilda sub-folder has been created with files in there



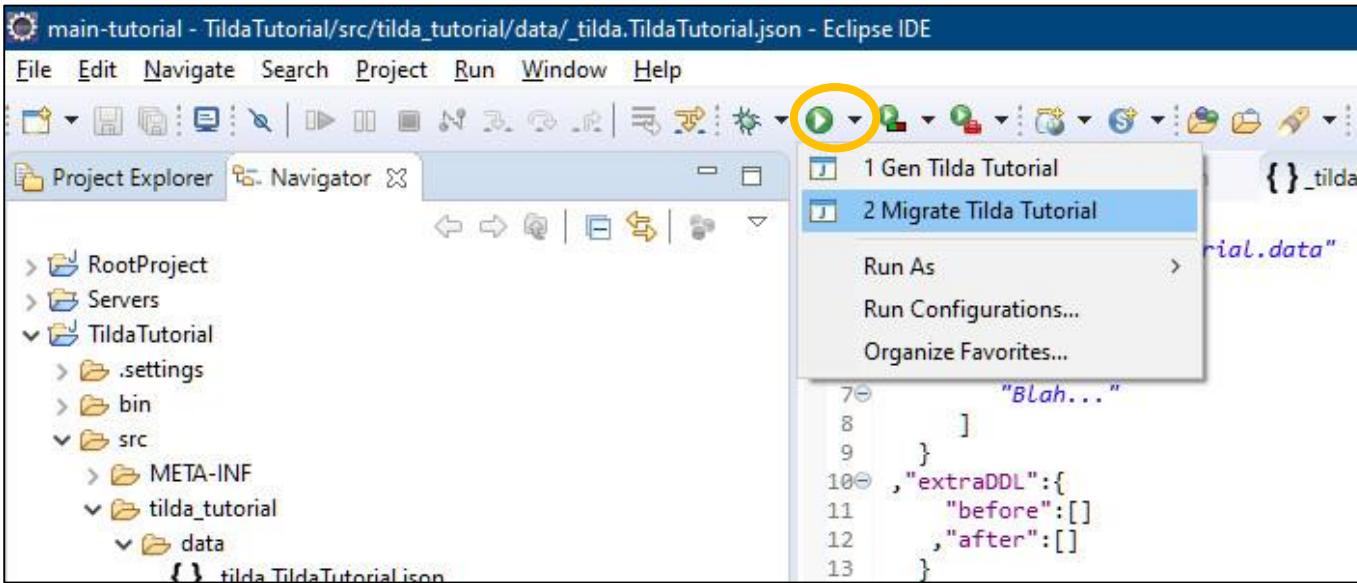
TILDA DEFINITION FILE

- Create the TildaTutorial database using your favorite DB tool
 - PGAdmin, Toad, Azure Studio or DBeaver for example.
 - Here we use DBeaver



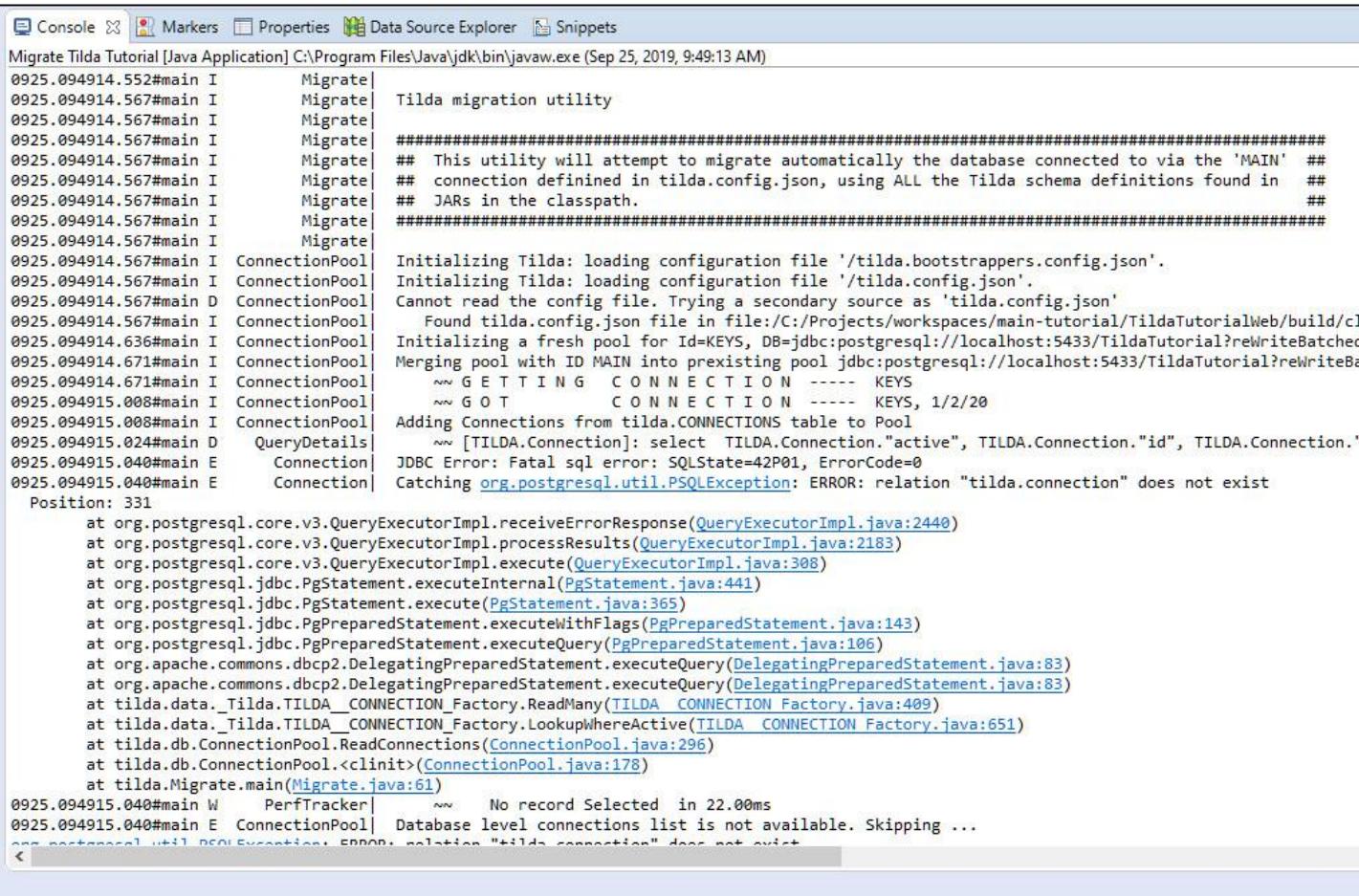
TILDA DEFINITION FILE

- Run Migrate



TILDA DEFINITION FILE

- First time we run Migrate, the database is completely empty, so you'll get some errors that you can ignore.



The screenshot shows a Java application console window titled "Migrate Tilda Tutorial [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe (Sep 25, 2019, 9:49:13 AM)". The console output is as follows:

```
0925.094914.552#main I      Migrate| 
0925.094914.567#main I      Migrate| Tilda migration utility
0925.094914.567#main I      Migrate|
0925.094914.567#main I      Migrate| #####
0925.094914.567#main I      Migrate| ## This utility will attempt to migrate automatically the database connected to via the 'MAIN' ##
0925.094914.567#main I      Migrate| ## connection defined in tilda.config.json, using ALL the Tilda schema definitions found in ##
0925.094914.567#main I      Migrate| ## JARs in the classpath. ##
0925.094914.567#main I      Migrate| #####
0925.094914.567#main I      Migrate| Initializing Tilda: loading configuration file '/tilda.bootstrappers.config.json'.
0925.094914.567#main I      ConnectionPool| Initializing Tilda: loading configuration file '/tilda.config.json'.
0925.094914.567#main D      ConnectionPool| Cannot read the config file. Trying a secondary source as 'tilda.config.json'
0925.094914.567#main I      ConnectionPool|     Found tilda.config.json file in file:/C:/Projects/worksspaces/main-tutorial/TildaTutorialWeb/build/cla
0925.094914.636#main I      ConnectionPool| Initializing a fresh pool for Id=KEYS, DB=jdbc:postgresql://localhost:5433/TildaTutorial?reWriteBatched
0925.094914.671#main I      ConnectionPool| Merging pool with ID MAIN into preexisting pool jdbc:postgresql://localhost:5433/TildaTutorial?reWriteBatched
0925.094914.671#main I      ConnectionPool|     ~~ G E T T I N G   C O N N E C T I O N   ----- KEYS
0925.094915.008#main I      ConnectionPool|     ~~ G O T   C O N N E C T I O N   ----- KEYS, 1/2/20
0925.094915.008#main I      ConnectionPool| Adding Connections from tilda.CONNECTIONS table to Pool
0925.094915.024#main D      QueryDetails|     ~~ [TILDA.Connection]: select TILDA.Connection."active", TILDA.Connection."id", TILDA.Connection."d
0925.094915.040#main E      Connection| JDBC Error: Fatal sql error: SQLState=42P01, ErrorCode=0
0925.094915.040#main E      Connection| Catching org.postgresql.util.PSQLException: ERROR: relation "tilda.connection" does not exist
Position: 331
    at org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:2440)
    at org.postgresql.core.v3.QueryExecutorImpl.processResults(QueryExecutorImpl.java:2183)
    at org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:308)
    at org.postgresql.jdbc.PgStatement.executeInternal(PgStatement.java:441)
    at org.postgresql.jdbc.PgStatement.execute(PgStatement.java:365)
    at org.postgresql.jdbc.PgPreparedStatement.executeQuery(PgPreparedStatement.java:143)
    at org.postgresql.jdbc.PgPreparedStatement.executeQuery(PgPreparedStatement.java:106)
    at org.apache.commons.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:83)
    at org.apache.commons.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:83)
    at tilda.data._Tilda.TILDA_CONNECTION_Factory.ReadMany(TILDA CONNECTION Factory.java:409)
    at tilda.data._Tilda.TILDA_CONNECTION_Factory.LookupWhereActive(TILDA CONNECTION Factory.java:651)
    at tilda.db.ConnectionPool.ReadConnections(ConnectionPool.java:296)
    at tilda.db.ConnectionPool.<clinit>(ConnectionPool.java:178)
    at tilda.Migrate.main(Migrate.java:61)
0925.094915.040#main W      PerfTracker|     ~~ No record Selected in 22.00ms
0925.094915.040#main E      ConnectionPool| Database level connections list is not available. Skipping ...
<  postgresql_util.DBCException: ERROR: relation "tilda.connection" does not exist
```

TILDA DEFINITION FILE

- The utility prompts to start the analysis. Press ‘y’ followed by enter.

TILDA DEFINITION FILE

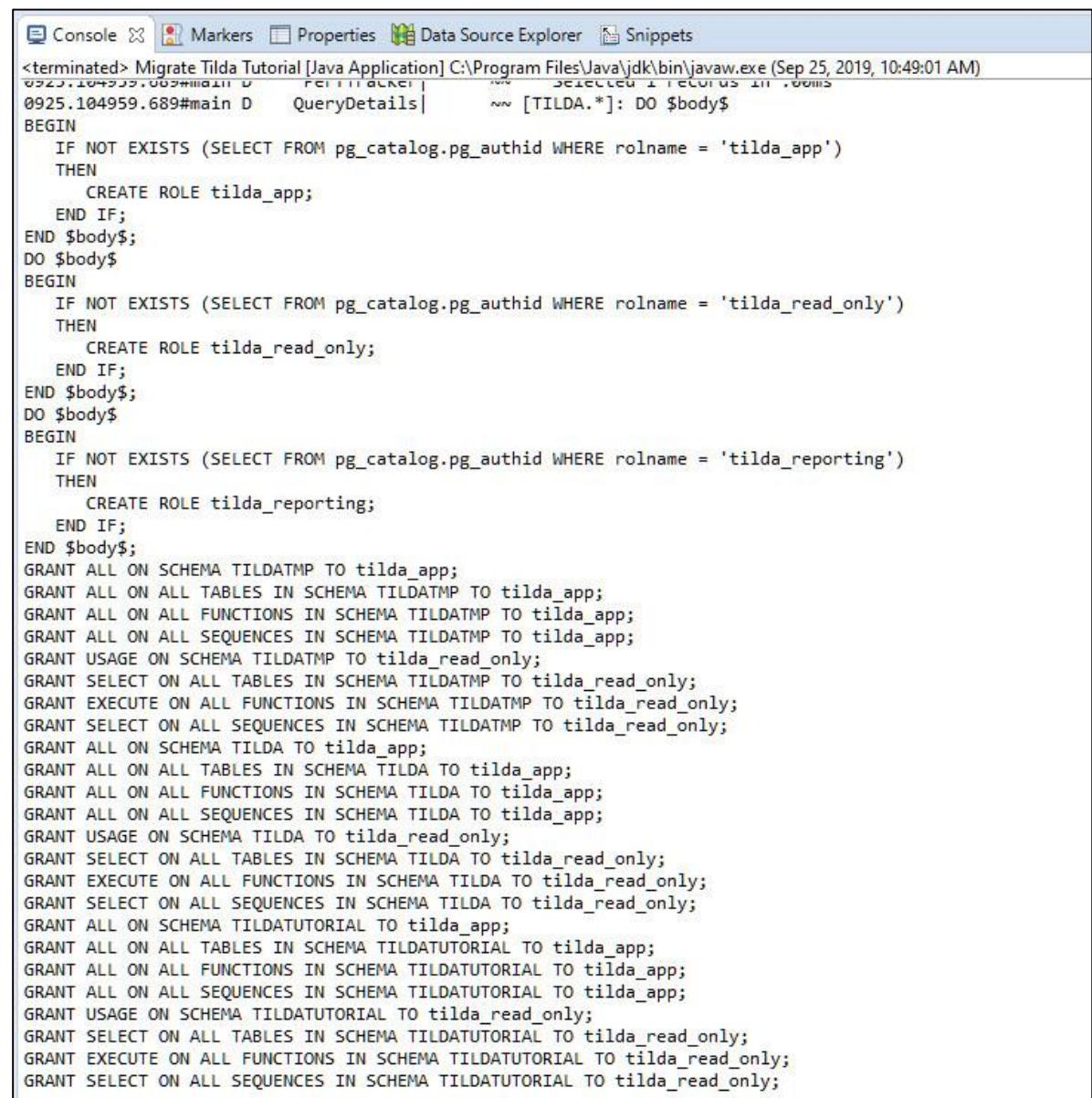
- For Dev, it's OK to run with the “postgres” admin account, but the utility flags this as a risk. In prod, you would not use a super user account.

TILDA DEFINITION FILE

- The analysis yields a number of migration steps for your review.
 - Type “yes” and enter to start the migration proper.
 - Note how so far we only have an empty schema for the TildaTutorial app.

TILDA DEFINITION FILE

- Migrate also manages the necessary roles for further work



The screenshot shows a Java application window titled "Console". The title bar includes tabs for "Console", "Markers", "Properties", "Data Source Explorer", and "Snippets". The main area displays a command-line interface for migrating Tilda roles. The text is as follows:

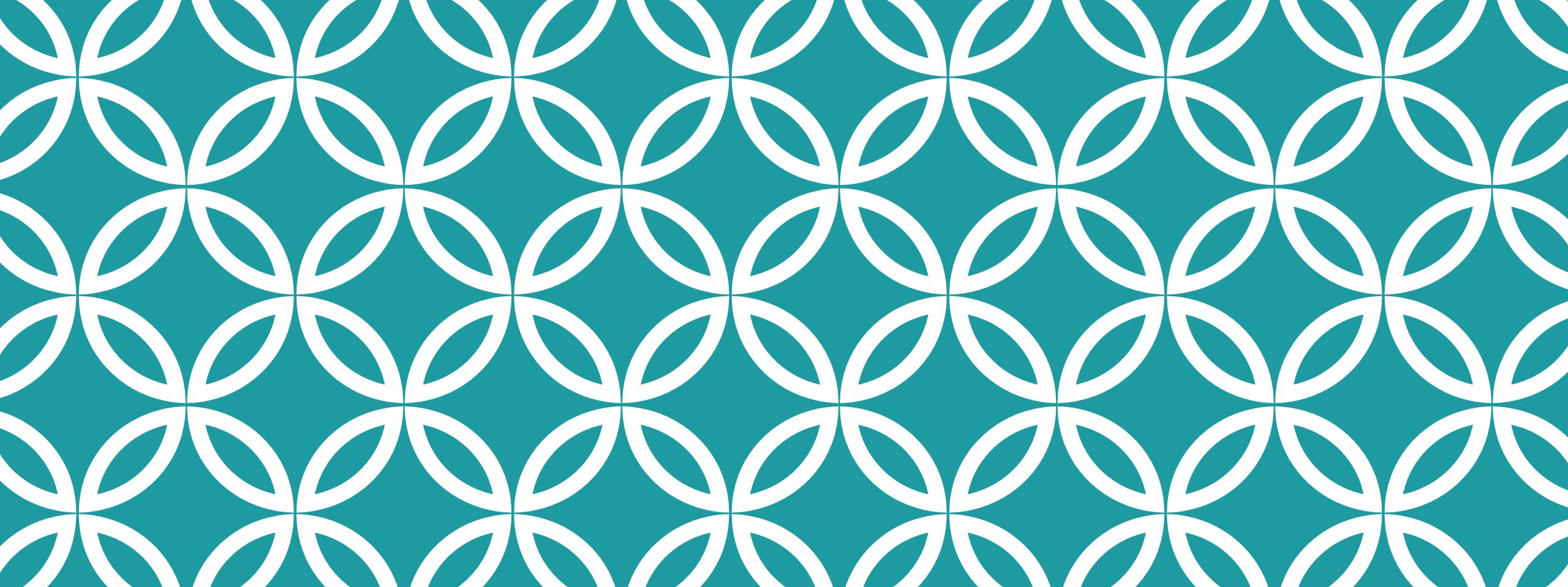
```
<terminated> Migrate Tilda Tutorial [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe (Sep 25, 2019, 10:49:01 AM)
0925.104959.689#main D    QueryDetails|      ~ [TILDA.*]: DO $body$
BEGIN
    IF NOT EXISTS (SELECT FROM pg_catalog.pg_authid WHERE rolname = 'tilda_app')
    THEN
        CREATE ROLE tilda_app;
    END IF;
END $body$;
DO $body$;
BEGIN
    IF NOT EXISTS (SELECT FROM pg_catalog.pg_authid WHERE rolname = 'tilda_read_only')
    THEN
        CREATE ROLE tilda_read_only;
    END IF;
END $body$;
DO $body$;
BEGIN
    IF NOT EXISTS (SELECT FROM pg_catalog.pg_authid WHERE rolname = 'tilda_reporting')
    THEN
        CREATE ROLE tilda_reporting;
    END IF;
END $body$;
GRANT ALL ON SCHEMA TILDATMP TO tilda_app;
GRANT ALL ON ALL TABLES IN SCHEMA TILDATMP TO tilda_app;
GRANT ALL ON ALL FUNCTIONS IN SCHEMA TILDATMP TO tilda_app;
GRANT ALL ON ALL SEQUENCES IN SCHEMA TILDATMP TO tilda_app;
GRANT USAGE ON SCHEMA TILDATMP TO tilda_read_only;
GRANT SELECT ON ALL TABLES IN SCHEMA TILDATMP TO tilda_read_only;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA TILDATMP TO tilda_read_only;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA TILDATMP TO tilda_read_only;
GRANT ALL ON SCHEMA TILDA TO tilda_app;
GRANT ALL ON ALL TABLES IN SCHEMA TILDA TO tilda_app;
GRANT ALL ON ALL FUNCTIONS IN SCHEMA TILDA TO tilda_app;
GRANT ALL ON ALL SEQUENCES IN SCHEMA TILDA TO tilda_app;
GRANT USAGE ON SCHEMA TILDA TO tilda_read_only;
GRANT SELECT ON ALL TABLES IN SCHEMA TILDA TO tilda_read_only;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA TILDA TO tilda_read_only;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA TILDA TO tilda_read_only;
GRANT ALL ON SCHEMA TILDATUTORIAL TO tilda_app;
GRANT ALL ON ALL TABLES IN SCHEMA TILDATUTORIAL TO tilda_app;
GRANT ALL ON ALL FUNCTIONS IN SCHEMA TILDATUTORIAL TO tilda_app;
GRANT ALL ON ALL SEQUENCES IN SCHEMA TILDATUTORIAL TO tilda_app;
GRANT USAGE ON SCHEMA TILDATUTORIAL TO tilda_read_only;
GRANT SELECT ON ALL TABLES IN SCHEMA TILDATUTORIAL TO tilda_read_only;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA TILDATUTORIAL TO tilda_read_only;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA TILDATUTORIAL TO tilda_read_only;
```

TILDA DEFINITION FILE

- Get a Woohoo!
- Check the DB to see the new schemas and tables created.

```
Console ✘ Markers Properties Data Source Explorer Snippets
<terminated> Migrate Tilda Tutorial [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe (Sep 25, 2019, 10:49:01 AM)
0925.104959.718#main I ConnectionPool|    ~~ G O T      CONNECTION ----- KEYS, 3/0/20
0925.104959.720#main D QueryDetails|    ~~ [TILDA.Key]: select TILDA.Key."refnum", TILDA.Key."name", TI
0925.104959.738#main D PerfTracker|    ~~ Selected 5 records in 17.00ms
0925.104959.738#main I Connection|    ~~ R O L L B A C K ----- KEYS: jdbc:po
0925.104959.738#main I Connection|    ~~ C L O S I N G   A N D   R E T U R N I N G   C O N N E C T I O
0925.104959.739#main I Connection|    ~~ C O M M I T ----- MAIN: jdbc:po
0925.104959.739#main I Connection|    ~~ C L O S I N G   A N D   R E T U R N I N G   C O N N E C T I O
0925.104959.739#main I ConnectionPool|    ~~ C L O S I N G   A N D   R E T U R N I N G   C O N N E C T I O

0925.104959.739#main I     Connection|    ~~ C L O S I N G   A N D   R E T U R N I N G   C O N N E C T I O
0925.104959.787#main I SystemValues|    System startup on Sep 25 2019, 10:49:59EDT
0925.104959.787#main I     Migrate|    =====
 $\begin{array}{c} / \swarrow \searrow \\ \backslash \end{array}$   $\begin{array}{c} / \searrow \\ \backslash \end{array}$ 
The database was automatically migrated to match the Application's data model.
=====
```



CONCLUSION

READY TO PLAY

- At this point, you have the infrastructure in place to do your development with Tilda
 - We have used eclipse here, but any java IDE with the ability to run command-like tools can work.
 - We have done a Root/Jar/Web configuration of projects, but other configurations and approaches are entirely doable as well: Tilda doesn't dictate any of this.
 - Some people use notepad++ and the command line for their work
 - You can use whatever DB tool you like to work against the database.
- There are two tutorials which build on this “getting started” guide in the Wiki.
 - <https://github.com/CapsicoHealth/Tilda/wiki/First-Tutorial-Part-1%3A-Benefits-Of-A-Transparent-Iterative-Model-Driven-Approach>
 - <https://github.com/CapsicoHealth/Tilda/wiki/WebDev-Tutorial-Part-1%3A-Easy-Servlets>