# Town Wizard DB Project Layout

The following directories are available under the application root directory:

* **pom.xml** Maven configuration file (I think POM stands for Project Object Model). It declares all application dependencies managed by Maven and has some other instruction for maven
* **src** Root source code directory which contains Java files, and some configuration files
* **src/main/java** Java source files. Java files are organized in packages, and the directories repeate the structure of the packages. There are two groups of packages.  
    
  The packages located under *com.townwizard.db* contain code specific to user services and data. The subpackages are:
  + **application** Java classes to run the project in a standalone server
  + **configuration** Configuration service classes
  + **constants** Constants
  + **dao** Data access classes and interfaces (they use hibernate to talk to DB)
  + **logger** Centralized logger class
  + **model** Java classes which represent business objects (User, Address, etc). They have hibernate mapping logic in them.
  + **resources** Classes, which actually handle the web service requests
  + **services** Classes, which represent “mid-tier” and transactional layer between resources and dao objects (that is the tiers in the application are: Resources -> Services -> DAOs
  + **util** Utility (helper) classes  
      
    The second group of packages resides located under *com.townwizard.globaldata* and contains code related to serving global content such as events and places.
  + **connector** Classes handling low-level code communicating with providers such as Google, Facebook, Yellow Pages
  + **dao** Related DAO classes
  + **model** and **model.directory** Related model classes
  + **service** and **service.geo** Service level (mid-tier) classes and interfaces
  + **service.provider** Service level classes responsible of handling providers’ data

* **src/main/resources** Contains configuration files
  + **application.properties** Containes properties like DB connection parameters. For now it’s just one file, and thre properties are hardcoded. In the future, we will probably have different config files for different environments. The current file will suffice though if we run Tomcat on the same host with the PHP application
  + **application.xml**  This is Spring configuration file. Mainly, it creates an object responsible for managing DB connections, and also it tells Spring to scan application directories in order to search and instantiate all Resources, Services, and DAOs. The last but not least, it creates declaratively a transaction manager object which handles application transactions
* **src/main/webapp** Contains standard Java web application WEB-INF/web.xml file necessary to deploy it on any application server like Tomcat. This file is parsed by Tomcat and it instantiate the servlet responsible for instantiating all Spring objects
* **src/main/test** Root directory for unit test Java classes and configuration files
* **sql** SQL scripts broken down by migrations. Also contains sample data sql scripts.
* **tomcat-model** Contains examples of what needs to be done in order to deploy the application under Tomcat for reference. For now, there is only one file in there, but the directory structure under this directory mimics the Tomcat’s one
* **target** Contains Maven build artifact. This directory is added to the git ignore, so any changes in it will not be reported by git
* **.classpath, .project, .settings** These are eclipse project files
* **doc** Document directory
* **jmeter** JMeter project files