*Spring* Boot is not intended to replace Spring, but to make working with it faster and easier. As a result, most of the changes needed for migrating an application are related to configuration. For the most part, our OAPI controllers and other service components will remain the same.

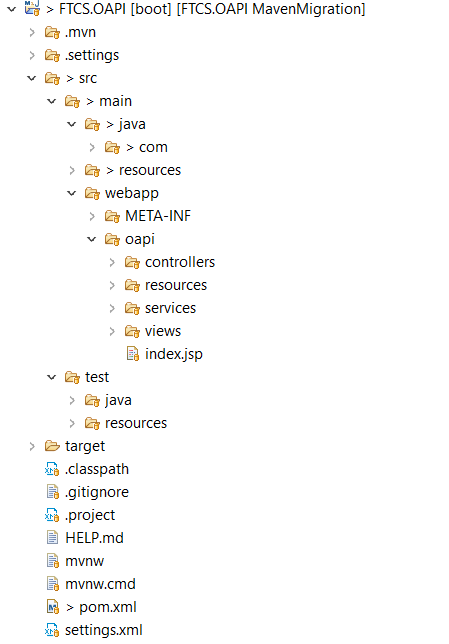
**Developing with Spring Boot brings several advantages:**

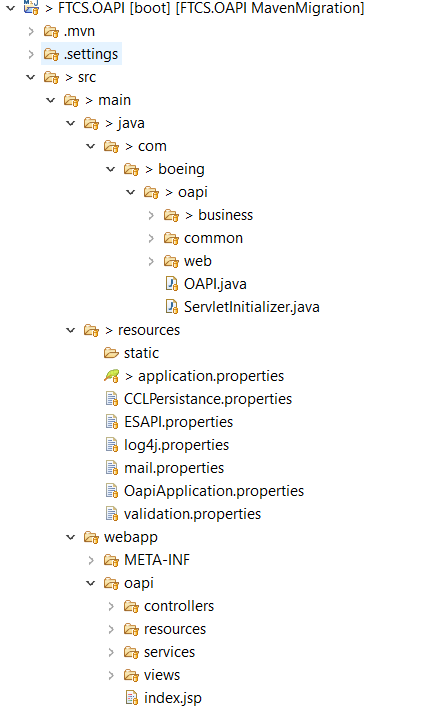
* simpler dependency management
* default auto-configuration
* embedded web server
* application metrics and health checks
* advanced externalized configuration
* Unit & Integration Tests

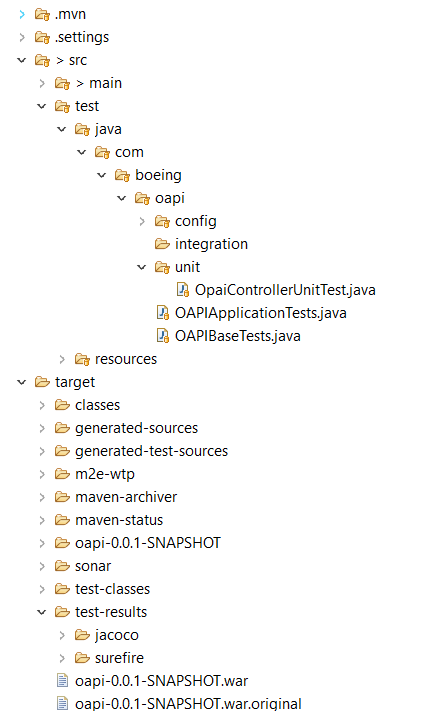
**OAPI Spring Boot migration**

1. Created a Spring Boot project and copied src, webapp from existing OAPI project.
2. Removed all xml's including web.xml and added Annotations.
3. Deleted Log.java file.
4. Excluded CCL Logger for now.
5. Added all dependencies to pom.xml
6. Created Spring Boot main app (OAPIApp).
7. Created ServletInitializer class and extended with SpringBootServletInitializer and overridden its configure method
8. Added application.properties in resource folder and added a property spring.datasource.jndi-name in it.
9. Added context.xml under webapp/META-INF and configured JNDI resource.
10. Created local Tomcat server and configured datatsource in server.xml.
11. Configured Jacoco in pom.xml for code coverage
12. Created JPATestConfig and application-test.properties for JUnits and created sample Unit tests
13. Created build pipeline in Azure.
14. Deploy in local Tomcat and test it.

**Folder Structure:**







**Pending Tasks:**

* Add as many JUnits for Codecoverage
* Log Management – ELK
* Create Release pipelines for Dnew/Qnew/Bdev/PCF Cloud
* Create Pipelines for UI & Soap UI automation testing post release and prior to UAT/Prod.

**GIT Check-In History:**

