Project Beginning

Martin Ledvinka

martin.ledvinka@fel.cvut.cz

Winter Term 2018

Contents

Deploy

WAR

- *Web Archive*
- Format of deployable Java web application artefacts
  - **EAR** for full-blown Java EE artefacts

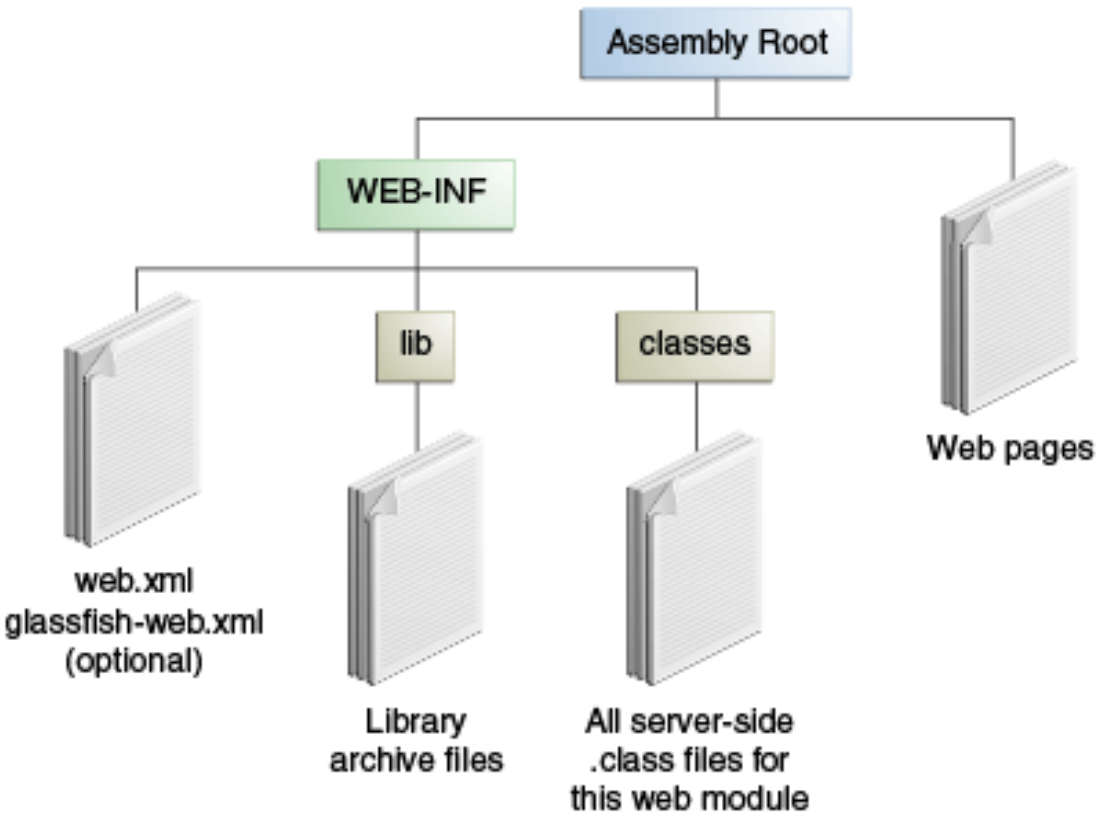

Figure : WAR structure. Source:
`https://docs.oracle.com/javaee/7/tutorial/packaging003.htm`

WAR cont.

- `web.xml` optional since Servlet API 3
  - All configuration can be done in source using Java + annotations
  - We won't be using it in our projects
- `WEB-INF` is not part of the public document tree of the application
  - Not accessible by clients
  - But accessible by servlet code – on classpath
  - Contains application code
- `lib` for required libraries, e.g., Spring, JDBC driver

Deployment

The following applies to Apache Tomcat!

- `webapps` folder for deployed web applications
- Can deploy exploded war (unpacked)
  - Tomcat will otherwise unpack WARs automatically
- Tomcat watches for changes in `webapps`
  - Copy into folder – *deploy*
  - Remove WAR from folder – *undeploy*
  - Application context
    - WAR file name
    - `META-INF/context.xml` in deployed WAR
    - `context.xml` in server configuration

Demo

- Demo of e-shop war deployment to Tomcat
- Demo of e-shop deployment in IntelliJ IDEA via a Run configuration

Maven

Apache Maven

- Software project management and comprehension tool
- Manage project dependencies, build, reporting, documentation
- Repository with libraries
  - Maven central at `maven.org` (web UI at `http://search.maven.org`)
  - Possible to have own repository, see e.g. `http://kbss.felk.cvut.cz/m2repo`
  - Local repository – cache

POM

- *Project Object Model*
- `pom.xml` file
  - Central XML-based configuration of Maven projects
  - Hierarchical project identification
    - `groupId`
    - `artifactId`
    - `version`
  - Manage dependencies – `dependencies` section
  - Manage build process – `build` section – using plugins – `plugins` section

Directory Structure

- `src`
  - `/main`
    - `/java`
    - `/resources`
    - `/webapp`
  - `/test`
    - `/java`
    - `/resources`
- `pom.xml`

Project Build Phases

1. *validate* - validate the project structure and configuration
2. *compile* - compile the source code of the project
3. *test* - test the compiled source code using a suitable testing framework
4. *package* - take the compiled code and package it in its distributable format, such as a JAR
5. *verify* - run any checks on results of integration tests to ensure quality criteria are met
6. *install* - install the package into the local repository
7. *deploy* - copy the final package to the remote repository

Dependency Scopes

- *compile* – default, dependency available on classpath
- *provided* – expected to be provided at runtime – by JDK, application server etc.
- *runtime* – not required for compilation, but is for execution
- *test* – required for test compilation and execution
- *system* – similar to provided except that you have to provide the JAR which contains it explicitly. The artifact is always available and is not looked up in a repository.
- *import* – used when specifying dependencies in parent projects

Gradle

Maven	**Gradle**
XML	Groovy
Maven repo	Maven repo
Plugins	Plugins, direct code
Recompile everything on build	Incremental build

Task

Task – 1 point

1. Clone project `https://gitlab.fel.cvut.cz/ear/b181-eshop`
2. Checkout branch *b181-seminar-02-task*
3. Create a Maven project using the `HelloWorld.java` and `HelloWorldTest.java` files
   - `HelloWorld` is a servlet which should be accessible to the client
   - `HelloWorldTest` is a simple JUnit-based test of `HelloWorld`
   - You may use the `sample-pom.xml` as a reference, but your project has to have unique groupId and artifactId and no excess dependencies
4. Configure the project and demonstrate to the teacher that:
   - It can be packaged using Maven
   - Tests are run during build
   - When you deploy it on Tomcat, the servlet is accessible through a web browser

The End

The End

Thank You

Resources

- `http://maven.apache.org/guides/`
- `https://docs.oracle.com/javaee/7/tutorial/packaging003.htm`