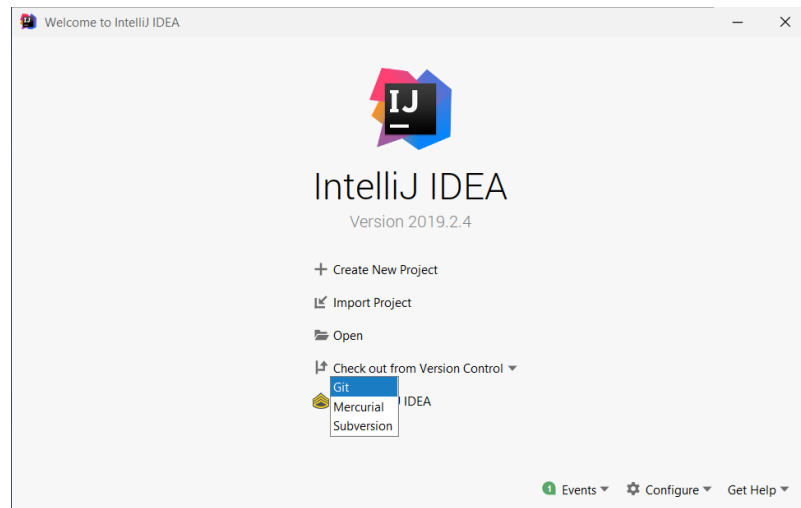


How to build and run projects

Step-by-step instructions

- 1) Create new project using “Check out from Version Control” option and git VCS:

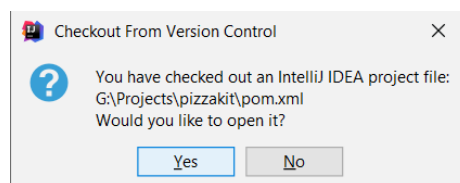


Specify URL of this GitHub repository and the project root (where you want to clone repo):

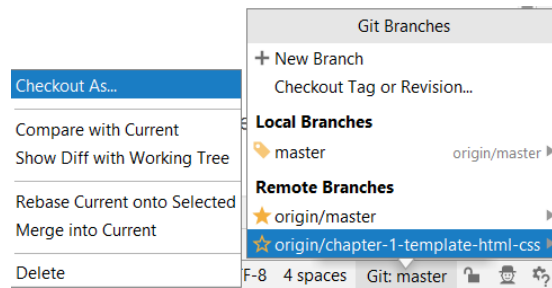


And press “Clone” button. You don’t have to Log in to GitHub.

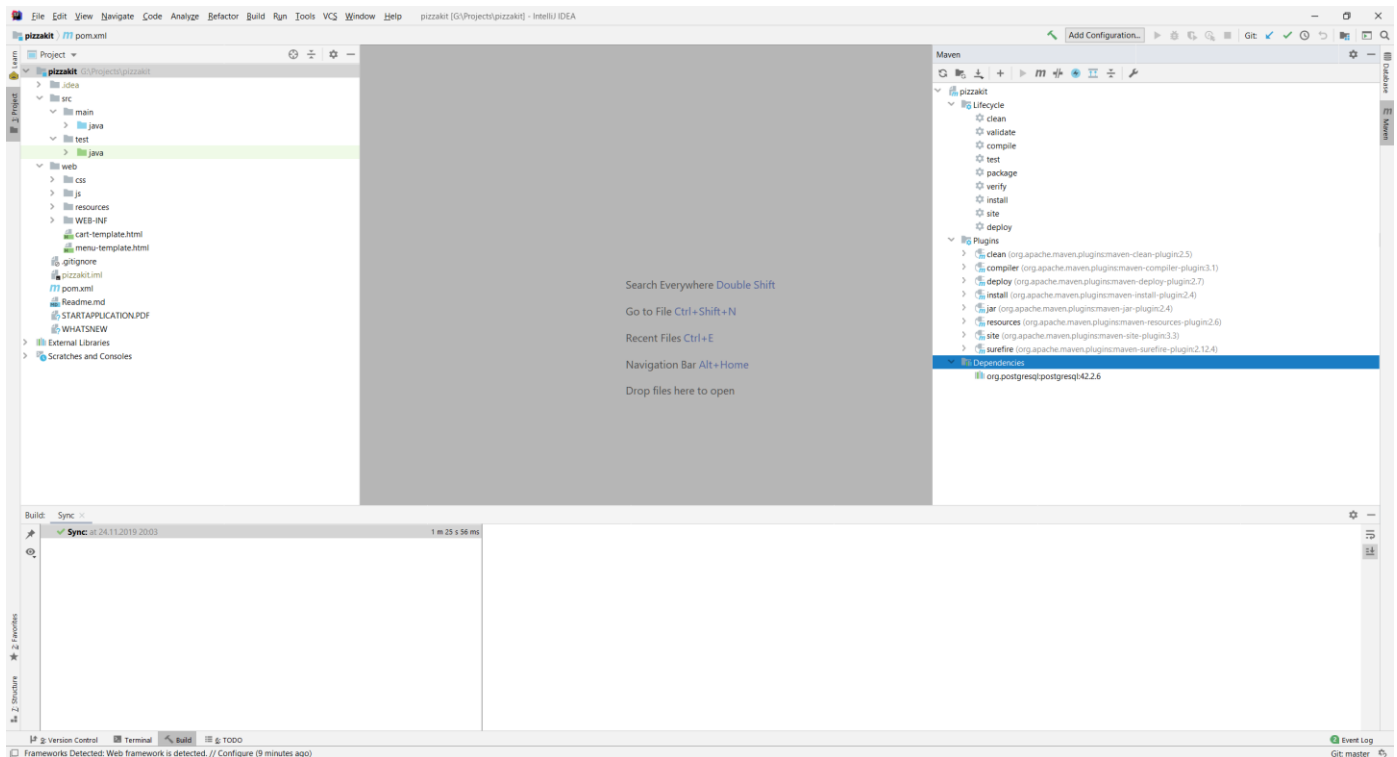
- 2) Thanks to IDE. It knows a lot of technologies, and maven – one of them. It determines that we use maven and asks to open pom.xml to process it. Just agree with.



- 3) After it IDE open the project and synchronizes our maven project (add plugins, verify dependencies and so on). Wait while the process ends.
- 4) Switch to the “chapter-1-template-html-css” branch. You can click on icon to the right bottom corner of IDE, or use menu VCS => Git => Branches. Then choose remote branch **origin/chapter-1-template-html-css** and click on “checkout as” submenu item. And press OK in opened dialog window. This meant that IDE creates a local branch (branch in local repository), connected with remote branch (branch in remote repository).

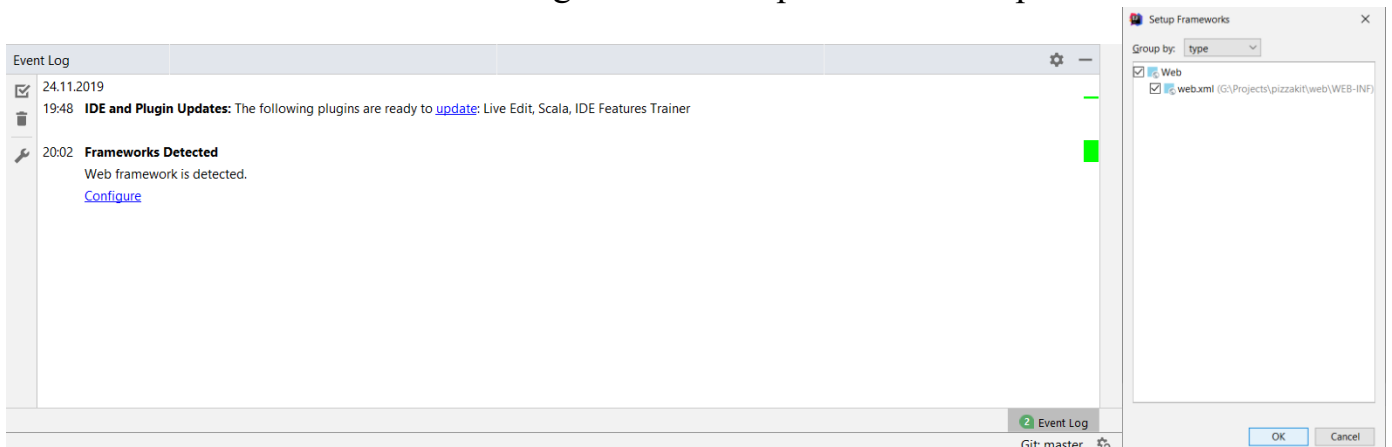



- 5) And don't forget to import changes (maven), when IDEA asks you about.
- 6) At "Maven" tab you can see your dependencies (now it's Postgres JDBC drivers – for future), and different "commands" you can use to build artifacts, deploy them to app server and etc. At "Project" tab you'll see "src" directory (for our classes) and "web" directory for web module.



But the module is not added to project. We have to add it.

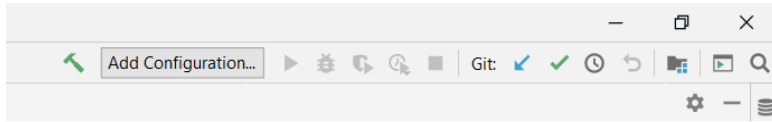
- 7) There are several ways to do this. You can push at "Event log" icon at the right bottom corner of IDEA window. And then in opened tab you'll see IDEA's message that it has detected web framework. You can click on "Configure" link and press "OK" in opened window.



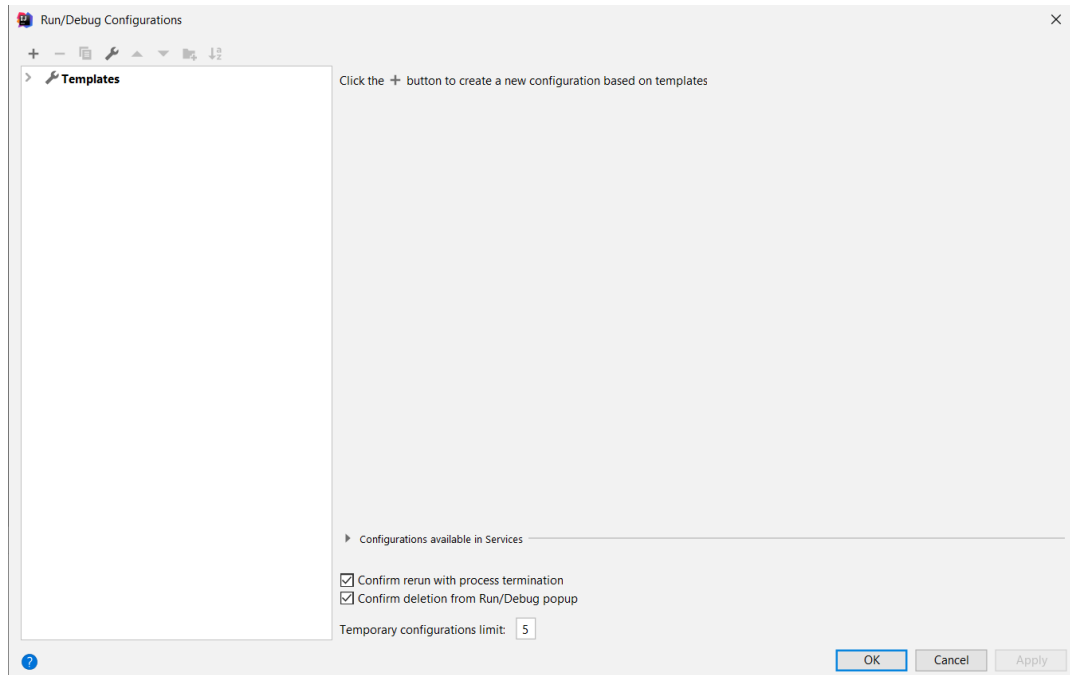
Web root will change it's icon to:  web .

Other way is to add module on “Project Settings” window (“Modules” tab).

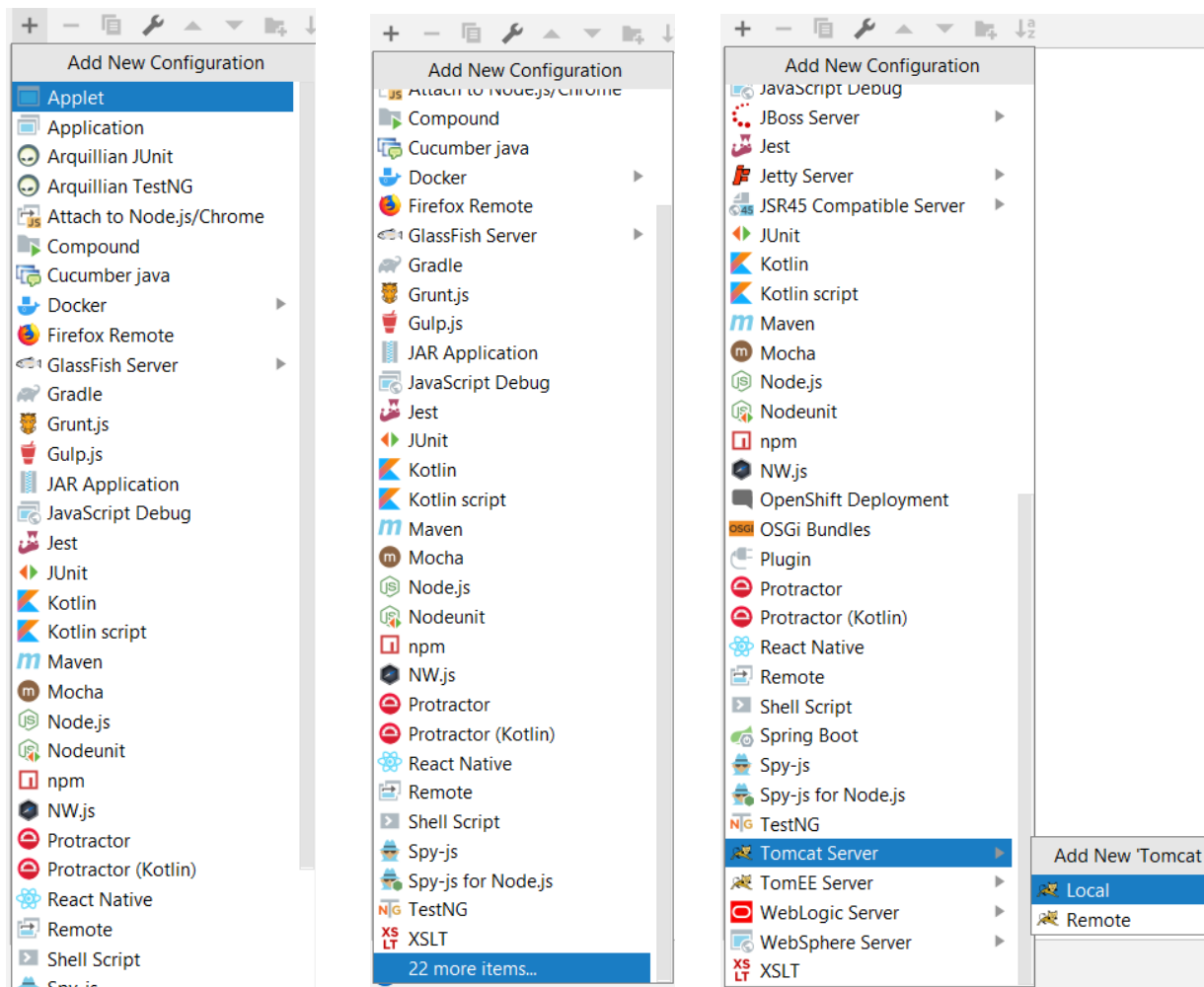
8) Now we have to configure build and deploy options. Look at the top right corner



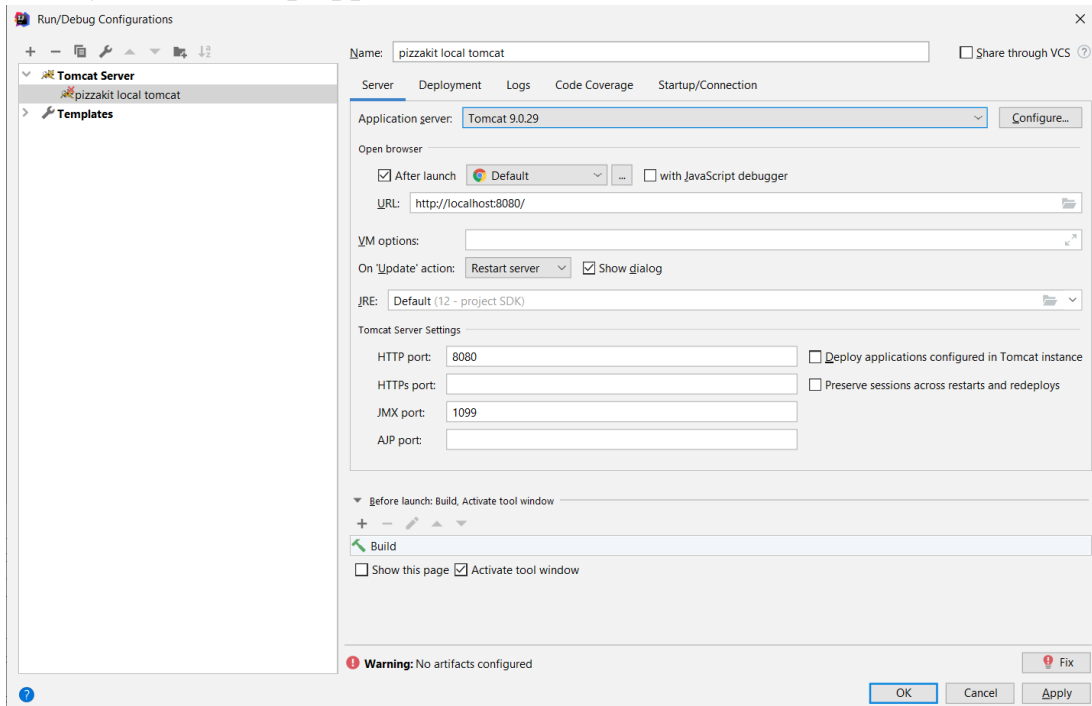
Buttons: run and debug buttons are not active. Moreover, there is no build configuration. Click on “Add configuration” to add it and corresponding window opens:



Click on “+” button. You need to choose “Tomcat server” option. If there is no such option, scroll down and click on “more items”. Choose “Local” server.

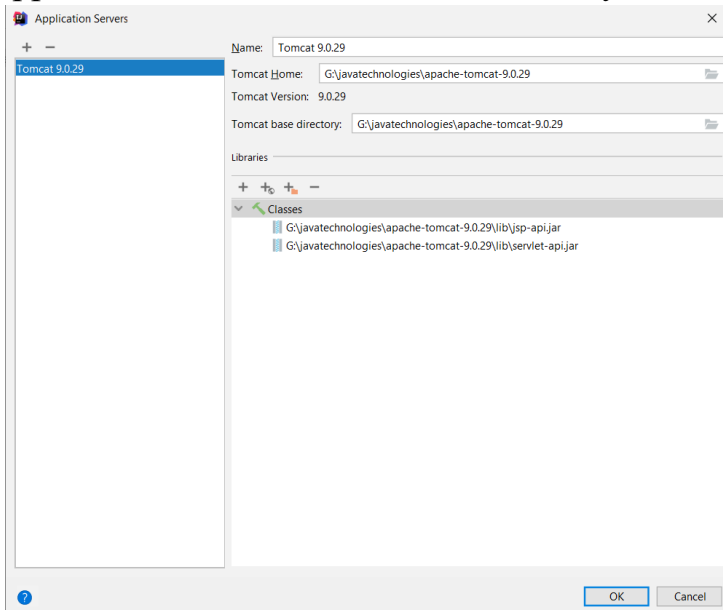


9) Now you can set-up app server.



Look at params. Name – you can input name of your configuration. Tab Server (other tabs left untouched):

- Application server – here you can choose app server. Click the Configure button to add app server and choose it's home directory:

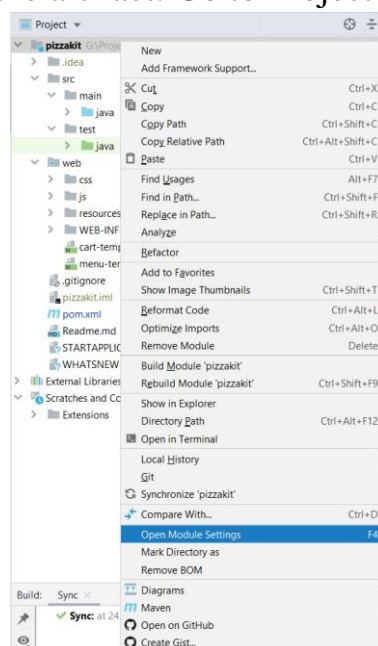


- Open Browser – here you can choose browser to open after deploy action
- URL – url of web-server. You will access your web-app by this url
- JRE – JRE, used to launch app server
- HTTP port – you can specify port of web-server

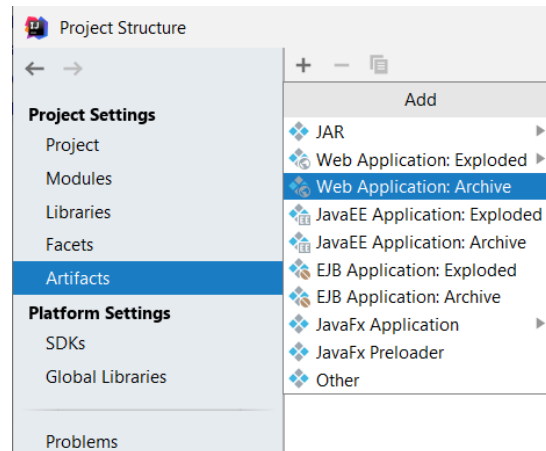
You can ignore IDE's warning about artifacts for now. We are dealing with it in further steps.

Main run process's stages of every JavaEE applications are: 1) build artefact; 2) start\restart app server; 3) deploy artefact to app server. There are some options with 2d and 3d stages depends on application, artifact type (exploded or archive, war, ear ...), app server itself. We will restart server every time we deploy something to it. Let's configure it, using IDEA instruments.

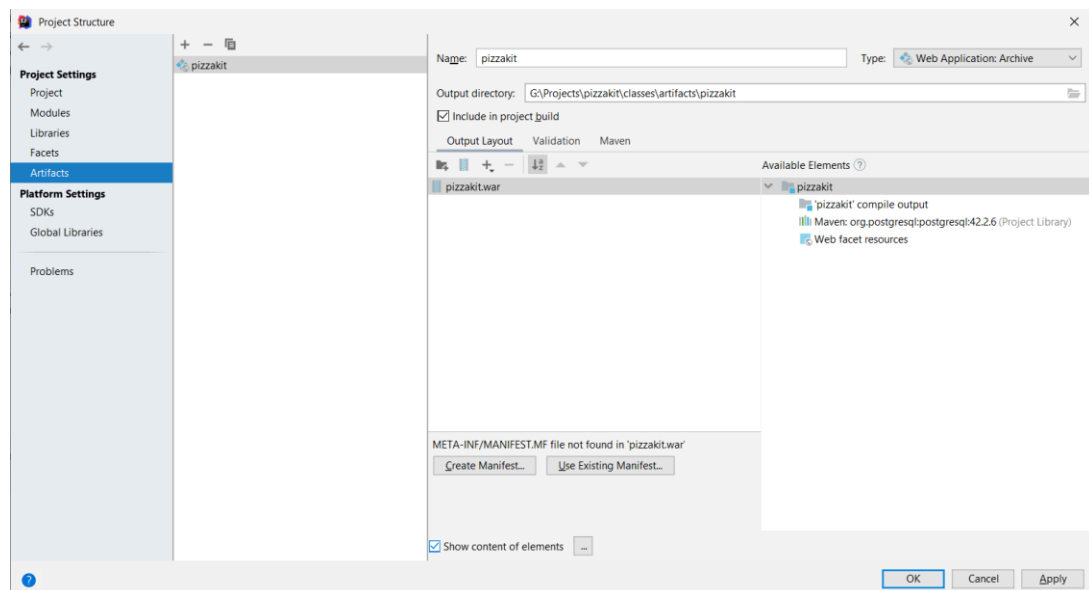
- 10) First, we need to describe the artifact. Go to Project setting.



Then go to “Artifacts” tab and add “Web Application: Archive” (You can choose “Exploded” the difference is that Archive is WAR-file and Exploded is unarchived Archive).



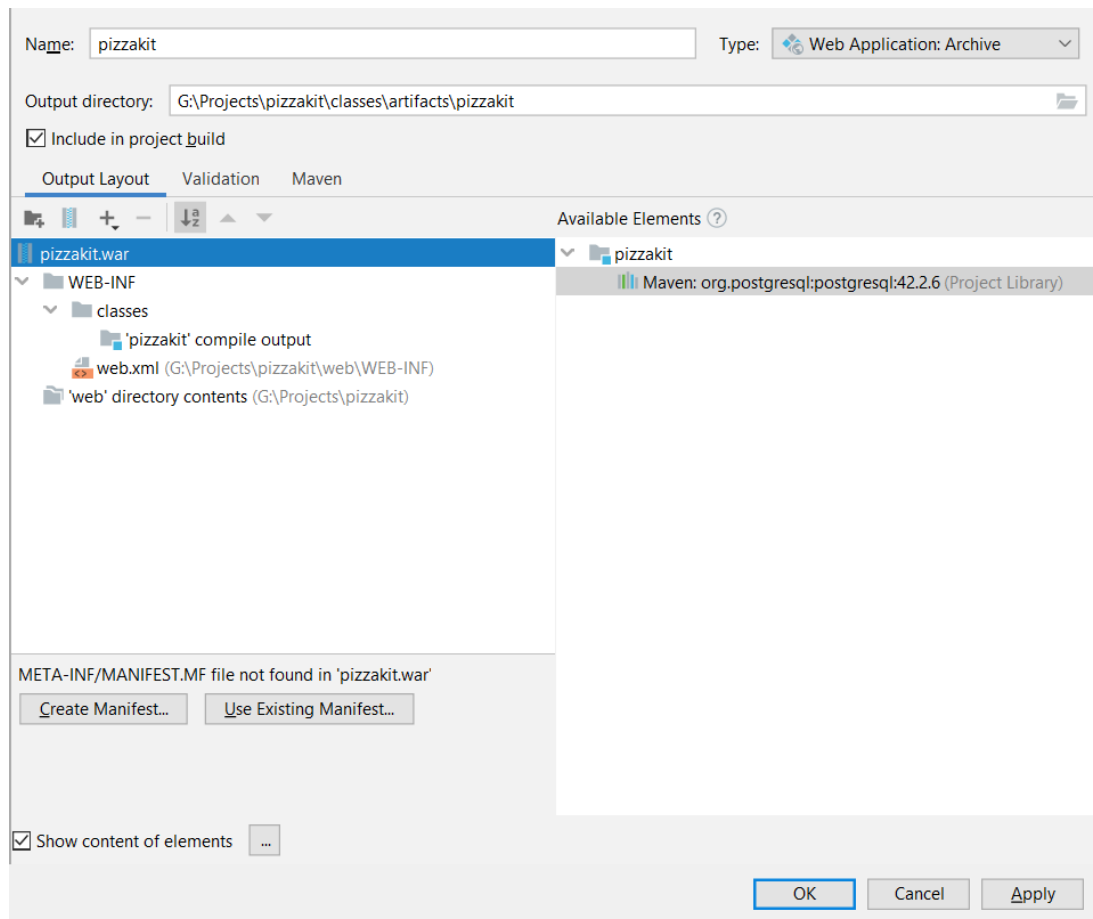
What you need to understand in opened window:



Name – name should describe what you are going to build.

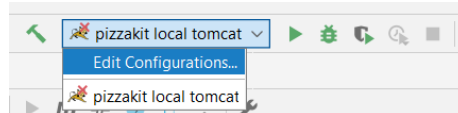
Output directory – generated by IDE and depends on project output, compiled classes and included resources will be placed there (don’t touch it)

Output layout describes the content of artifact. Left click on “Web facet resource” (this is the content of our web module directory – web) and “pizzakit’ compile output” (the compiled classes that we are going to create in **src\main\java** will be placed in **web-inf\classes** directory). You can place libraries to your artifact. But there is a better way – place them manually to **<tomcatdir>/lib** directory. Result:



And don't forget to press "OK" button =)))

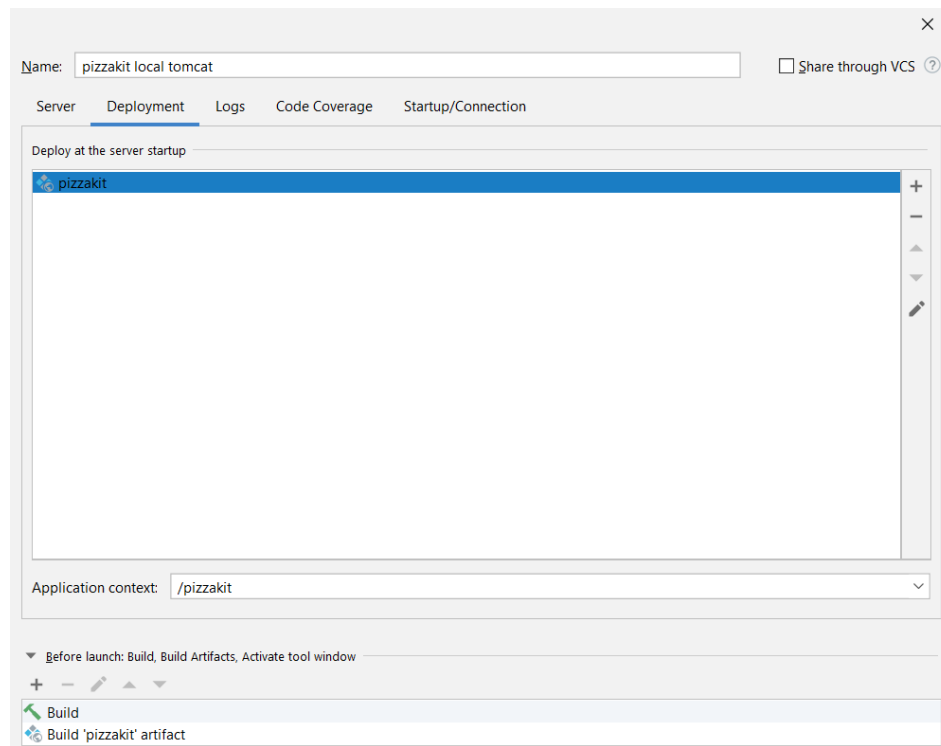
- 11) Now we need to configure a deployment of the artifact process. Let's edit our build configuration once again (select "Edit configuration" option)



Go to "Deployment" tab and add artifact



Couse we have only one artifact; it is chosen automatically. Also, you can specify an Application context root. It determines the URL of your application. If you specify **/pizzakit** (as on the next image) you have to use **localhost:8080/pizzakit** url. If you use only / as context root, you have to use **localhost:8080/** url.



12) Now You are ready to start our first “static” application, consisted of 2 html files (plus stiles, plus images). Press the usual green triangle button “Run” (Alt+shift+F10 on Win platform). You will see the catalina (servlets container’s name) output:



And after all process have finished (building artefact, starting app server, deploying artefact) – you can open browser (if it hasn’t started automatically) and input url, depends on your configuration (step 11). In our example it is **http://localhost:8080/pizzakit/**