Oomph Setup JSDT Development Environment

Case study: Bug 488343.

Get the custom .setup

Open Bug 488343; check the related Gerrit change, and download the zip containing the setup file.

Extract the archive, locate the Oomph .setup file, and launch Oomph

Run Oomph

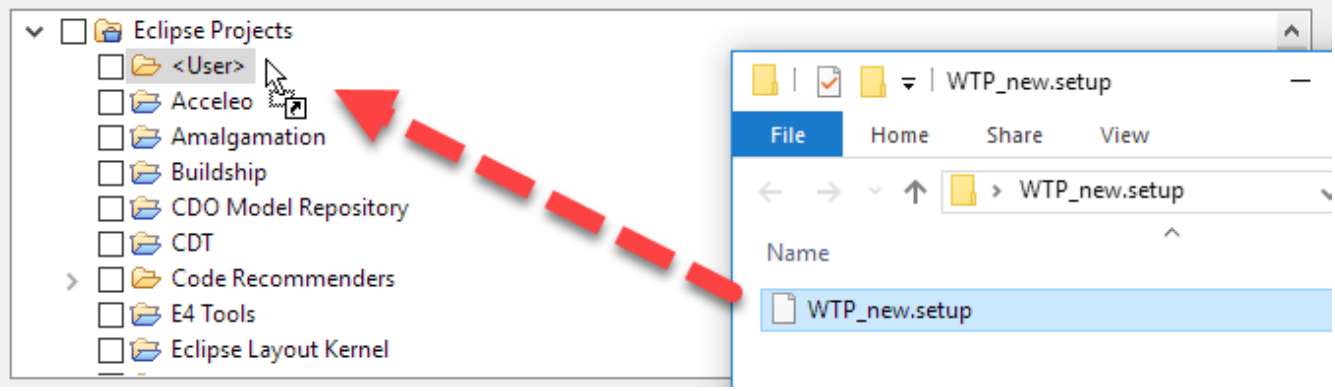
Launch the Eclipse Installer . [1: You should download and install the Oomph installer] , and then switch to advanced mode.

In the next page you can select the base eclipse, to use for development. Choose *Eclipse for RCP and RAP*. Plus, you're suggested to use the 64bit version, to avoid memory limitation issues.

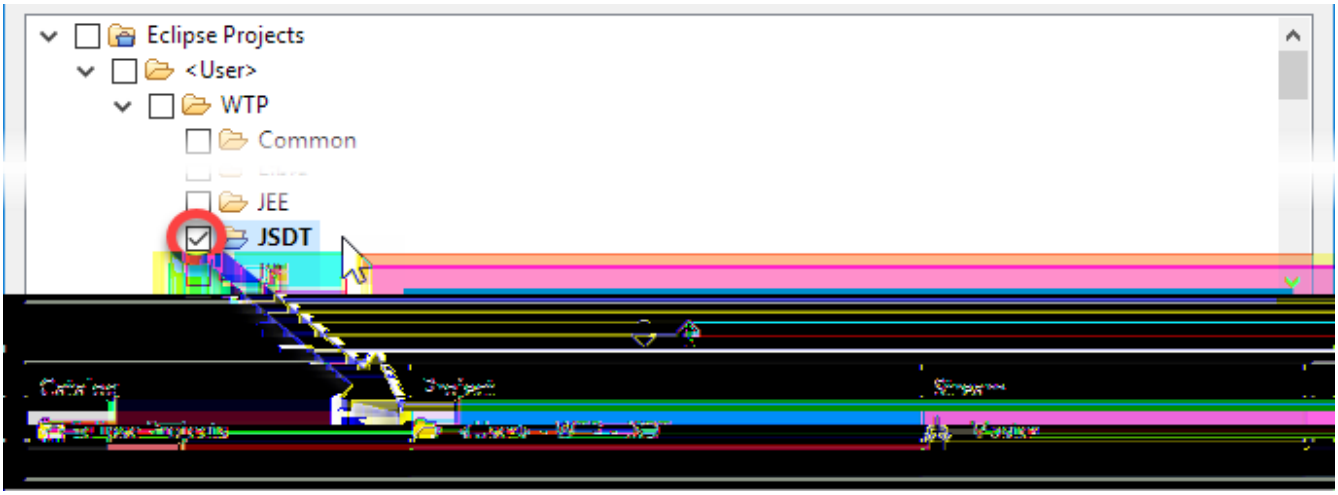
In the Next page you choose which projects you will contribute to.

Use custom .setup

Currently, the JSDT .setup model is not in the Oomph public catalog. To use the setup, drag and drop the .setup file into the *User* folder.



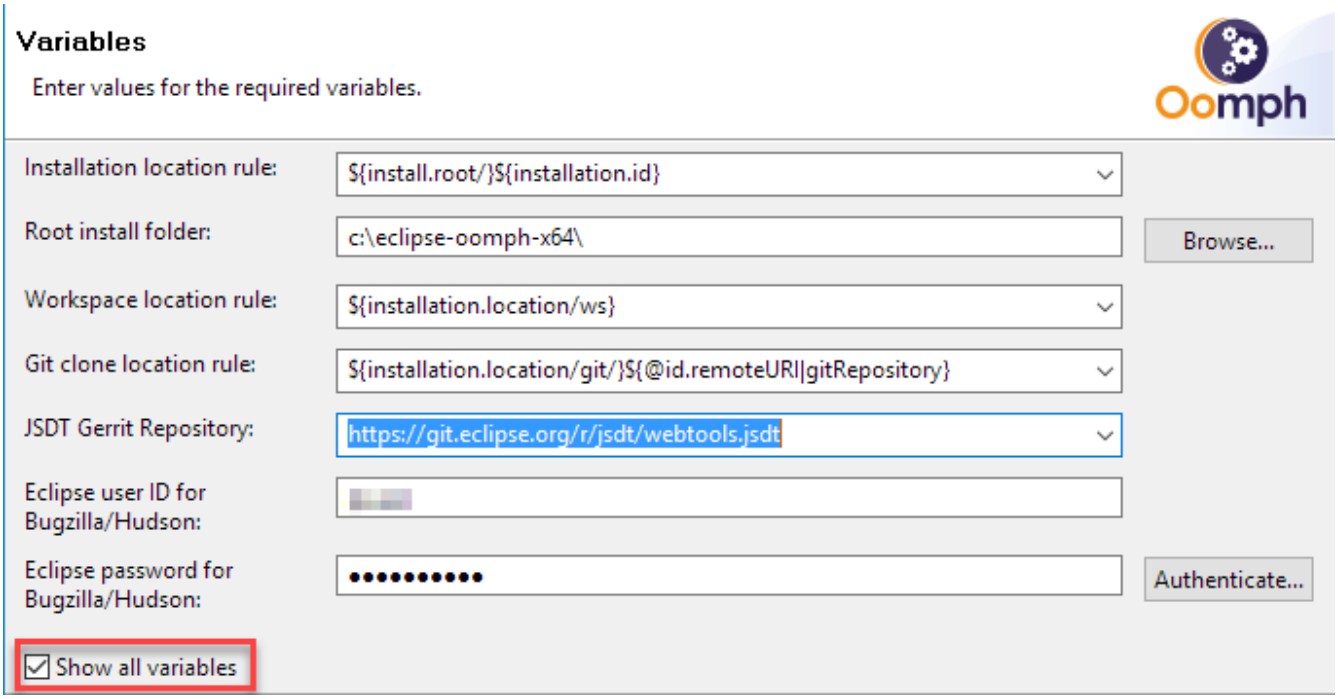
Then, under *User*, you can see the *WTP* subtree. Identify the *JSDT* node; check its checkbox and verify the project is added.



Now press *Next*, and proceed to the variables page.

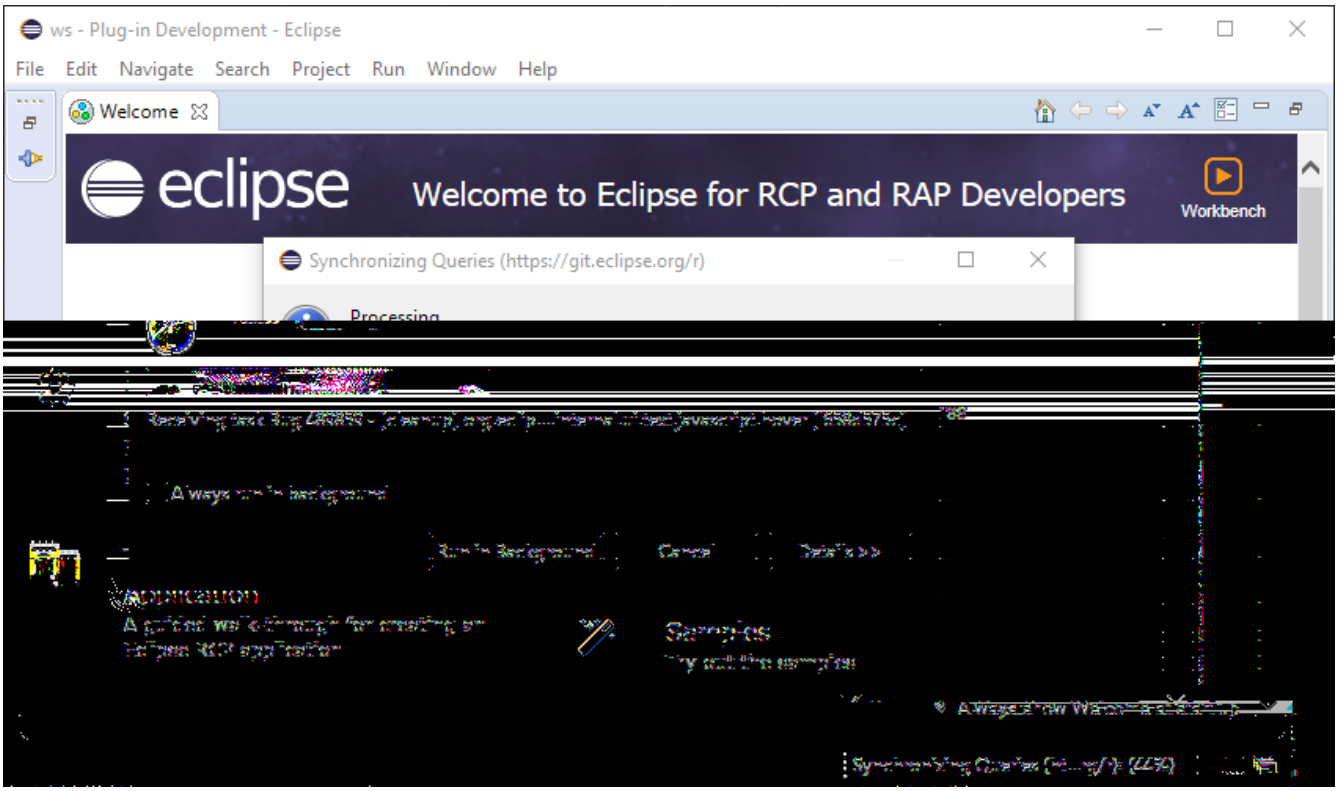
Setup variables and install

Check the *Show all variables* checkbox, so you can check and configure the parameters for your installation, e.g.: *installation location*, *git rules*, *Gerrit repo*, *Eclipse userid* etc..



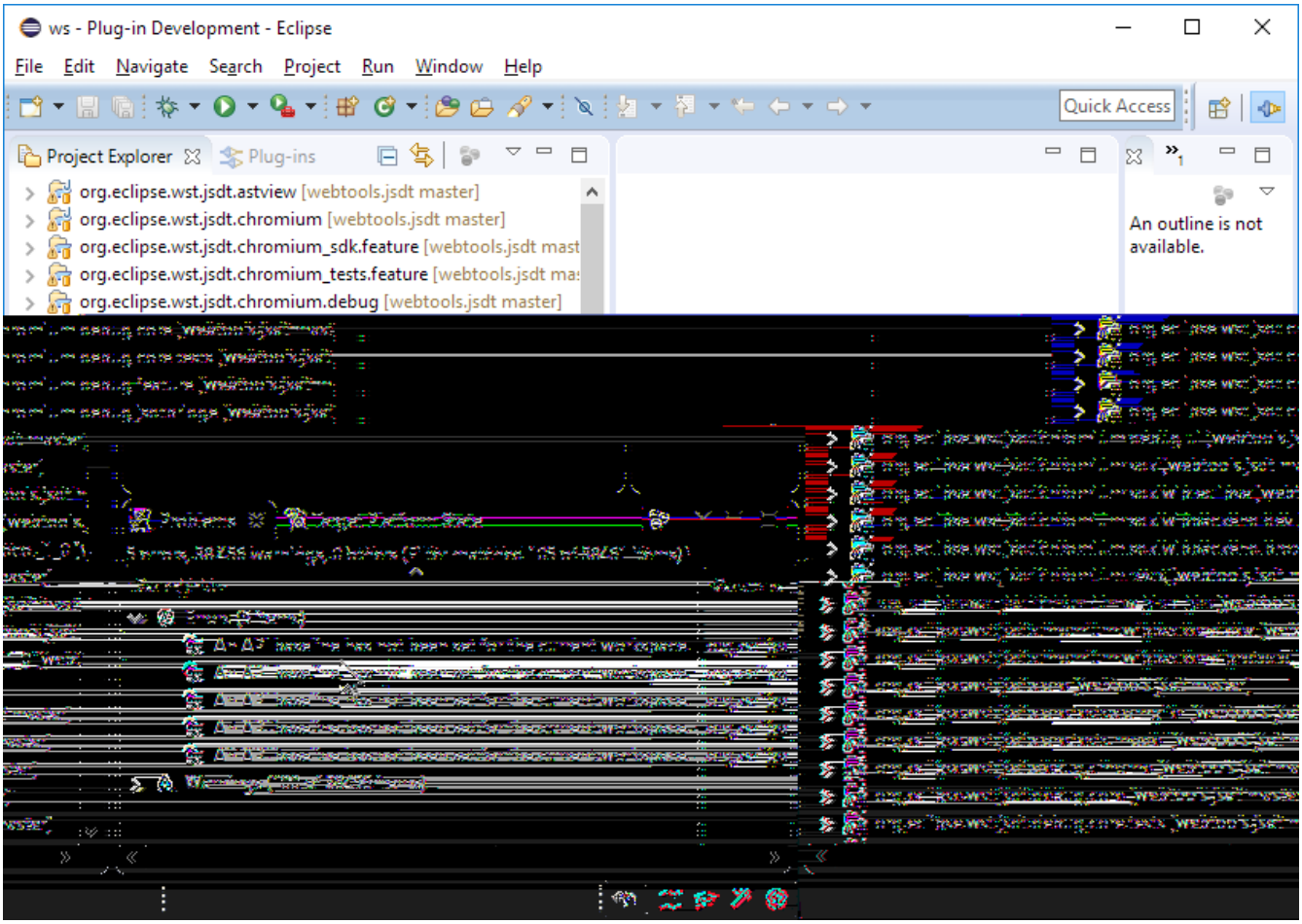
In the next page press *Finish* to start downloading and installing the bundles. The download is fast, as with the bundle pool, Oomph never downloads the same bundle twice.

While Oomph is completing the install, you will see the new instance of Eclipse popping up, and completing the setup tasks, including getting bundles, and fetching source code from Git/Gerrit.

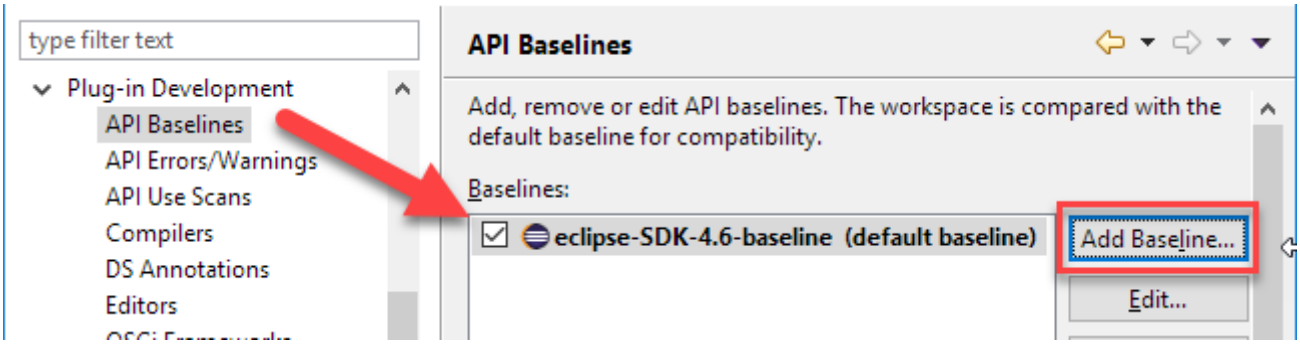


First run of your new IDE

When the setup is complete, close the welcome window. In the workspace you will see the list of JSDT projects, cloned from Git/Gerrit.



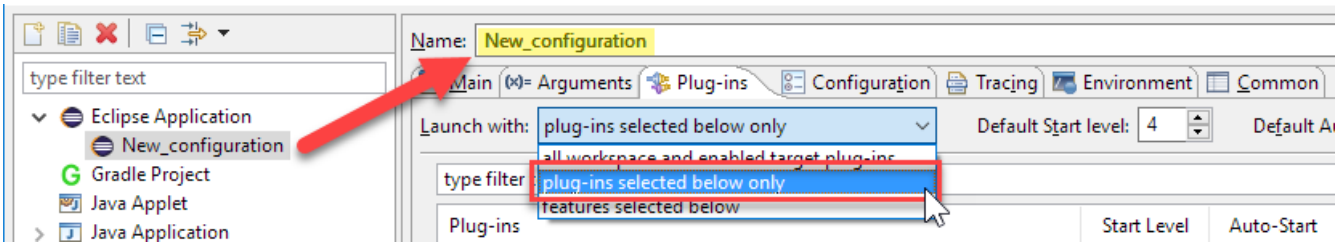
Look in the error log, where you can see errors related to a missing API Baseline. Fix this in *Preferences* > *Plug-in Development* > *API Baseline*, by adding an existing Eclipse Neon installation directory as target Baseline.



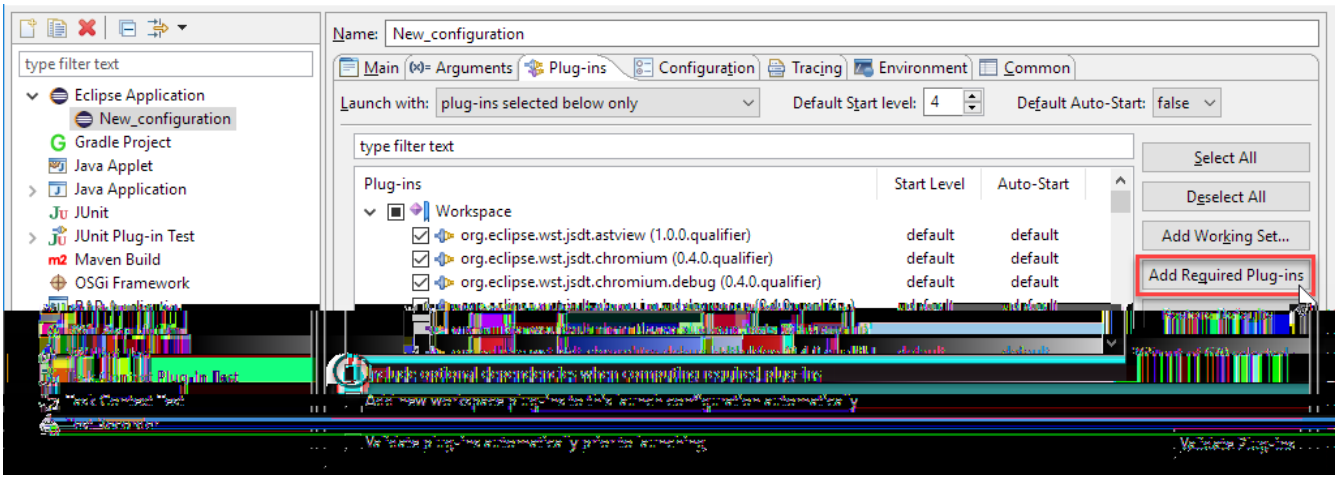
After this you're ready to launch an Eclipse product.

Launch Target Configuration

Open Run.. > Run Configuration and create a new Eclipse Application configuration. Switch to the Plug-ins tab, and select to launch with *Plug-ins selected below only* . [2: this give fine control in adding/removing plug-ins from the configuration.]

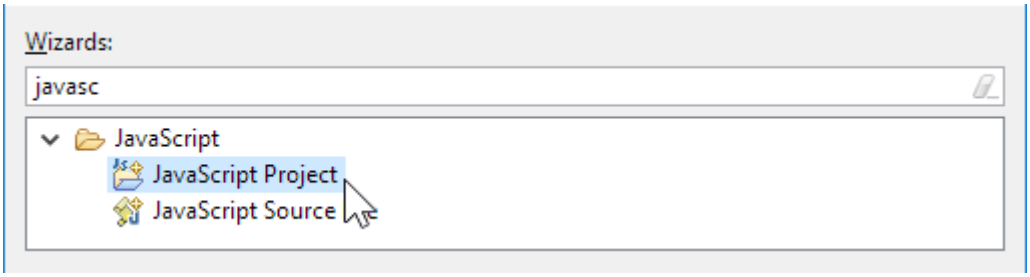


After this, remove the **test** plug-ins in the workspace from the launch configuration; de-select the *Include optional dependencies...* checkbox; click the *Add Required Plug-ins* button and, Finally, *Run* the launch configuration.



Run and Check the Target application

Now the target application is running, we can check that contains all the JSDT features The easy way to check this, is creating a javascript project.



Update : If you see some issue in running the target application, this means there are some adjustments we need to do to the setup file.

Conclusions

Creating an Oomph .setup is initially hard. But if the team manage to have a good setup, this will make easier to new developers to contribute.

Indeed, using an Oomph setup is pretty easy, since the first time you're using it. It just take a little time to get used to concepts like *bundle pools*; *.setup files*; *user variables*; and after a couple of runs, it is actually easier to use it rather than explaining how it works.

Many thanks to *Esteban Dugueperoux*, who contributed the initial setup, and to all the other

Contributors and doc writers, who are making Eclipse better.

Hope you enjoyed this walkthrough. Soon I'm going to make this document available under EPL both on the JSDT website and on my personal blog.

References

Written with AsciiDoc. See ref, also man.