Building CS-Studio with Maven

Will Rogers

Diamond Light Source

What do we want to do?

- Probably, create a custom product
- Include some features, exclude others
- Set up local configuration

- Alternatively, be able to work on the CS-Studio code

Some background: Maven

- Maven is a *Good Idea:*
  - Write pom.xml
  - Maven:
    - downloads the tools it needs
    - downloads the libraries you ask for
    - compiles the code
    - runs the tests
    - creates a product

Some background: Tycho

- Tycho is Maven for Eclipse builds:
  - It understands:
    - MANIFEST.MF
    - plugin.xml
    - feature.xml
    - cs-studio.product

Some background: OSGi

- OSGi is used by Eclipse:
  - The bundle system helps with modularity
  - However, it means that your libraries must be OSGi bundles

The repos

- diirt
- maven-osgi-bundles
- cs-studio-thirdparty
- cs-studio
- org.csstudio.product

Let’s start easy

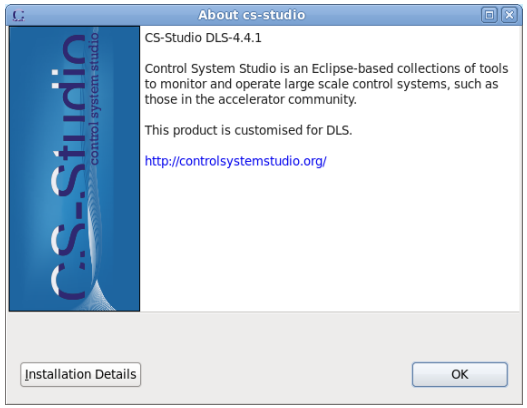
- ~~diirt~~
- ~~maven-osgi-bundles~~
- ~~cs-studio-thirdparty~~
- ~~cs-studio~~
- org.csstudio.product

The product

- You can define your own product fairly easily
  - Copy org.csstudio.product
    - `git clone git@github.com:ControlSystemStudio/org.csstudio.product`
    - `cd org.csstudio.product`
    - `mvn clean verify   # this took 6 minutes for me just now`
- Now customise:
  - Edit repository/cs-studio.product
    - choose the features you want
  - Change various branding things
  - If you have your own plugins, you can add them here
  - mvn clean verify
- Then maybe spend several hours fixing the plumbing but hopefully not

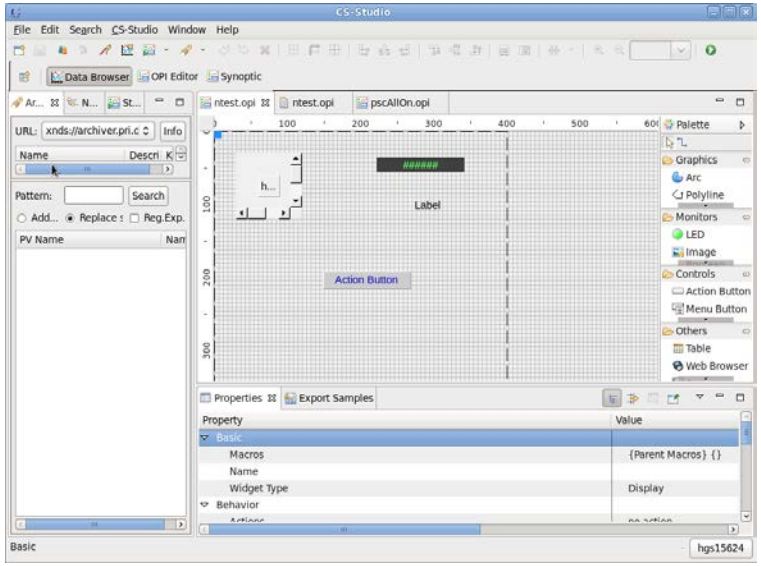
So what can you do?

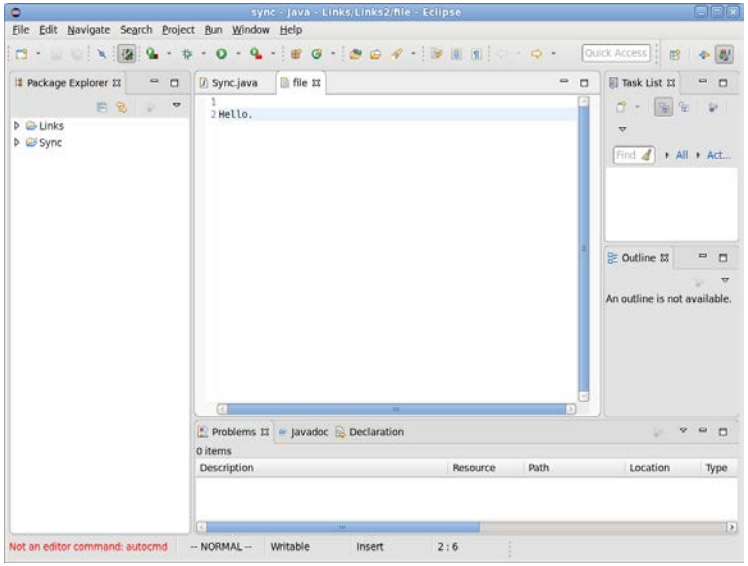
- Set default preferences
  - plugin_customization.ini
- Change the branding
- Choose your features
  - BOY, Databrowser, Archiver Appliance plugin
- Remove the stuff you don't want
  - BEAUTY, BEAST
- Add your own plugins
  - DLS elog



What can’t you do?

- Eclipse is Eclipse …





Some lessons

- `mvn clean verify`
  - this cleans, builds, tests and packages. I use this
- `mvn clean install`
  - cleans, builds, tests, packages and installs into the local repository. Not often useful
- `-DskipTests=true`
  - not running tests can save time
- `mvn -X`
  - very verbose logging, occasionally useful

Some more lessons

- `~/.m2/settings`
  - Maven settings
- `~/.m2/repository`
  - Maven local repo. This is where all the things that are downloaded are downloaded to. Sometimes it's a good idea to delete this
- p2 repos and m2 repos are not the same!
  - The collaboration hosts p2 repositories with all the jars you need
- Jar files aren't good enough
  - You need OSGi bundles, which are jars but have extra metadata. Often you can get these from the Eclipse repositories. Just because it's on Maven Central doesn't mean you can use it.

Diving in

- ~~diirt~~
- ~~maven-osgi-bundles~~
- ~~cs-studio-thirdparty~~
- cs-studio
- org.csstudio.product

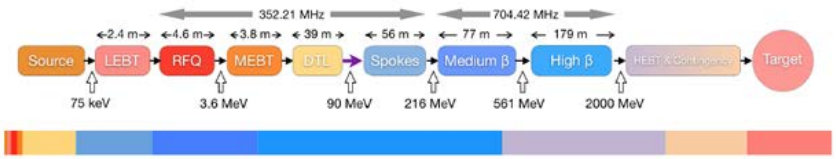
Development

1. Get Eclipse (Neon will do)
2. Build a product (see above)
3. Set your target platform to that product*
4. Clone cs-studio
5. Import the plugins you want to work with
   - Import as Existing Maven Projects!
6. Add test plugins** from Orbit***
7. Set up a run configuration****
   - Select org.csstudio.product.product
8. Press the green play button
9. Cross your fingers
10. Your changes should show up in the new application

* These bits require some Googling

Multi-stage build

- Now we need to build the product **against** the cs-studio (p2) repo
  - `cd /path/to/cs-studio/core`
  - `mvn clean verify`
  - `cd ../applications`
  - `mvn clean verify`
  - `cd /path/to/org.csstudio.product`
  - `mvn clean verify –Dcsstudio.composite.repo=/path/to/cs-studio/p2repo`
- Phew!
  - And we haven't covered `diirt`, `maven-osgi-bundles` and `cs-studio-thirdparty`



Summary

- Building CS-Studio is not trivial
- It is possible
- Ask for help!