Eclipse Server Installation

Eclipse for the People

We maintain an Eclipse installation that is ready to use for developers working at our research group. The installation is usually quite up-to-date, consisting of the most recent Eclipse release with a then-current selection of features already installed. Ideally, this installation can be used to work on every part of KIELER without the need to install additional features.

More importantly, this installation serves as the basis for the target platform KIELER is developed against. We found that a centralized and clearly defined target platform is necessary to avoid any version issues.

This documentation is not valid for the new Eclipse installation central to the institute.

Using the Installation

To use the installation, simply start Eclipse with the following command line:

```
/home/java/eclipse/eclipse
```

Administering the Installation

The information in this section is only relevant for those of us that actually administer the shared Eclipse installation. If you're just using it, this is not for you.

To-Do List for Installing a New Version

When installing a new version, follow this to-do list:

1. Install a basic Eclipse distribution in a properly named folder, e.g. `eclipse_4.2.1` or `eclipse_4.3_modeling`.
2. Install additional features as described in Getting Eclipse.
3. Install the delta pack necessary to build products for other platforms.
4. Update `/home/java/eclipse` to point to the new installation.
5. Copy the installation into a new directory and copy all plugins and features of the delta pack into its `plugins` and `features` folders.
   Produce a P2 reference repository for the automatic Maven build to use. To do that, execute the following script with reasonable parameters

   ```
   #!/bin/bash
   if ([ $# -lt 3 ] || [ $# -gt 4 ])
   then
     echo "Usage: $0 [ECLIPSE_INSTALLATION] [TARGET_DIR] [REPOSITORY_NAME] [SOURCE_DIR]?"
   else
     ECLIPSE_DIR=$1    # e.g. /home/java/eclipse-modeling-4.4.1/
     TARGET_DIR=$2     # e.g. /home/java/public_html/repository/
     REPO_NAME=$3      # e.g. luna441
     if [ $# -eq 3 ]
     then
       SOURCE_DIR=$ECLIPSE_DIR
     else
       SOURCE_DIR=$4
     fi
     CMD="java -jar $ECLIPSE_DIR/plugins/org.eclipse.equinox.launcher_*.jar \
          -application org.eclipse.equinox.p2.publisher.FeaturesAndBundlesPublisher \
          -metadataRepository file:$TARGET_DIR/$REPO_NAME \
          -artifactRepository file:$TARGET_DIR/$REPO_NAME \
          -source $SOURCE_DIR \
          -configs any.any.any \
          -compress \
          -publishArtifacts"
     echo $CMD
     eval $CMD
   fi
   ```

6. Move the repository to our `kieler` user's `public_html/repository/` directory.
7. Download the respective eclipse delta pack and create a P2 Repository for it as in 5.
8. Move the delta pack repository to `public_html/repository/<eclipse_release>_delta/`

9. Update the reference repository location in the parent POM files of the source code repositories. Also update the target platform definition files in the `config` repository.

Note on Windows

In case you want to setup the P2 repository on a Windows system, adapt the file paths as follows and take care to not append a **file:** after the `-source` argument.

```
java [...] -metadataRepository file:\\E:\juno42rep -artifactRepository file:\\E:\juno42rep -source E:\juno42rep
-publishArtifacts
```

Creating a Target Definition

To use the reference installation with private eclipse installations (e.g. on personal laptops) an eclipse *target definition* is required. As eclipse's tooling to create such a `.target` file is pretty much unusable, you are welcome to use the following script.

```
#!/bin/bash
#
# This script can be used to automatically create an eclipse target
# platform definition from the contents of a p2 repository.
#
# $NAME - The name of the created file
# $SEQ_N - A sequence number within the target definition. This should
#          be higher than the numbers used before as eclipse
#          uses it to cache states internally
# $TARGET_PLAT_* - Urls to the used p2 repositories
#

NAME="pragmatics_luna441.target"
SEQ_N=40
TARGET_PLAT="http://rtsys.informatik.uni-kiel.de/~kieler/repository/luna441/"
TARGET_PLAT_DELTA="http://rtsys.informatik.uni-kiel.de/~kieler/repository/luna441_delta/"

#
# Implementation below
#

# Ignore any platform specific fragments for the target definition.
# They will be resolved automatically.
IGNORE_FRAGMENTS="linux\|win\|cocoa\|mac\|solaris\|aix\|hpux"


# function to download the contents.jar from an p2 repository
# and extract all the installable units
# $1 - url of the p2 repository
function parseContent {

  TMP="tmp"
  mkdir $TMP
  wget -P $TMP "$1/content.jar" > /dev/null 2>&1
  unzip "$TMP/content.jar" -d $TMP  > /dev/null 2>&1

  CMD="cat $TMP/content.xml | grep -e '<unit .*' | grep -v -e '$IGNORE_FRAGMENTS' | sed \"s/singleton='false'
//g\" | sed \"s/>/\/>/g\""
  UNITS=$(eval $CMD)

  rm -r $TMP

  echo -e "$UNITS"
}

# function assembling a location
# $1 - the url of the p2 repository
# $2 - a list of units that should be part of the target definition
function location {
  LOCATION='    <location includeAllPlatforms="false" includeConfigurePhase="false" includeMode="slicer"
```

```
includeSource="true" type="InstallableUnit">
        '$2'
      <repository location="'$1'"/>
    </location>
  '
  echo -e "$LOCATION"
}



HEAD='<?xml version="1.0" encoding="UTF-8" standalone="no"?> \n
      <?pde version="3.8"?> \n
    <target name="'$NAME'" sequenceNumber="'$SEQ_N'"> \n
        <locations> \n
     '

FOOT='
  </locations> \n
</target>'


# print the header of the target definition
echo -e $HEAD > $NAME

# parse the provided content.xml file
UNITS=$(parseContent "$TARGET_PLAT")
echo -e "$(location "$TARGET_PLAT" "$UNITS")" >> $NAME

UNITS=$(parseContent "$TARGET_PLAT_DELTA")
echo -e "$(location "$TARGET_PLAT_DELTA" "$UNITS")" >> $NAME


# finish the target definition
echo -e $FOOT >> $NAME
```