Hands-On: Using the New Common Builder for Push-Button PDE Builds

Nick Boldt, Red Hat
Andrew Niefer, IBM
Andrew Overholt, Red Hat
Agenda

- Introduction
- PDE Build
  - Exercise: Build a Feature
  - Exercise: Generate a p2 Repository
- Common Build Background
- Dash Athena
  - Exercise: Setup
  - Exercise: Configure, Run, Troubleshoot
  - Exercise: Verify Build
- Meet Hudson
- BYO Build Clinic
Your Hosts

• Nick Boldt
  • JBoss, a division of Red Hat
  • Release Engineering (Modeling, Tools, Dash)

• Andrew Niefer
  • IBM Rational
  • PDE Build Maintainer

• Andrew Overholt
  • Red Hat
  • Linux Tools
Tutorial Exercises & Slides

1. Get a USB drive from presenters & copy its contents to your drive. Also available here: http://www.eclipsecon.org/2009/sessions?id=302

2. Unpack `org.eclipse.dash.common.releng.tutorial_*.zip` somewhere on your drive, e.g., `c:/tmp`. This zip includes both exercises and slides.

3. Open Eclipse 3.5M6, and select 
   File > Import > General > Existing Projects into Workspace

4. Browse to the workspace folder under where you unpacked the zip, e.g.,
   `c:/tmp/org.eclipse.dash.common.releng.tutorial/workspace/`
Tutorial Exercises (cont'd)

5. Import the following 6 projects from the workspace folder:

org.eclipse.dash.common.releng
org.eclipse.gef.releng
org.eclipse.gef.tree
org.eclipse.relenge.basebuilder
XWT.BuildDirectory
XWT.Builder

Remember to check
Copy projects into workspace.

Note that org.eclipse.relenge.basebuilder will not compile due to one or more missing requirements. This is not a problem.
Third Party Code

- On the USB drive, there may also be a downloads folder.

- You can choose to download those files manually or use the provided zips/jars **AS-IS** in order to save time re-downloading them. See **README.txt** for more information.

- You **will** need a copy of ant-contrib 1.0b2, and Eclipse (the rest is optional).

  - Copy ant-contrib.jar into your workspace here: `org.eclipse.dash.common.releng/lib/ant-contrib.jar`

  - If not using local files, you can get ant-contrib 1.0b2 here: `http://downloads.sourceforge.net/ant-contrib/ant-contrib-1.0b2-bin.zip`
Builds ...

- Builds are a TMTOWTDTI* problem
- Standardized = better
- Customization always required...
  ...but that leads away from standardization

* There's more than one way to do it
Like layers of an onion

- Sources in VCS repository
- Packaging details, dependencies, maps
- PDE Build + setup scripts
- X server for running tests
- Web front end for monitoring & auditing
OSGi Bundles

- META-INF/MANIFEST.MF

```manifest
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: %Plugin.name
Bundle-SymbolicName: org.eclipse.gef; singleton:=true
Bundle-Version: 3.4.0.qualifier
Bundle-Activator: org.eclipse.gef.internal.InternalGEFPlugin
Bundle-Vendor: %Plugin.providerName
Bundle-Localization: plugin
Import-Package: com.ibm.icu.text
Export-Package: org.eclipse.gef,
    org.eclipse.gef.commands,
    [...]
    org.eclipse.gef.util
Require-Bundle: org.eclipse.draw2d;visibility:=reexport;bundle-version="[3.2.0,4.0.0)",
    org.eclipse.core.runtime;bundle-version="[3.2.0,4.0.0)",
    org.eclipse.ui.views;resolution:=optional;bundle-version="[3.2.0,4.0.0)",
    org.eclipse.ui.workbench;bundle-version="[3.2.0,4.0.0)",
    org.eclipse.jface;bundle-version="[3.2.0,4.0.0)"
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment: J2SE-1.4
```
Bundle dependencies

- META-INF/MANIFEST.MF

```manifest
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: %Plugin.name
Bundle-SymbolicName: org.eclipse.gef; singleton:=true
Bundle-Version: 3.4.0.qualifier
Bundle-Activator: org.eclipse.gef.internal.InternalGEFPlugin
Bundle-Vendor: %Plugin.providerName
Bundle-Localization: plugin
Import-Package: com.ibm.icu.text
Export-Package: org.eclipse.gef,
  org.eclipse.gef.commands,
  ...
  org.eclipse.gef.util
Require-Bundle: org.eclipse.draw2d;visibility:=reexport;bundle-version="[3.2.0,4.0.0)",
  org.eclipse.core.runtime;bundle-version="[3.2.0,4.0.0)",
  org.eclipse.ui.views;resolution:=optional;bundle-version="[3.2.0,4.0.0)",
  org.eclipse.ui.workbench;bundle-version="[3.2.0,4.0.0)",
  org.eclipse.jface;bundle-version="[3.2.0,4.0.0)
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment: J2SE-1.4
```
Bundle build description

- build.properties

  bin.includes = about.*,
  eclipse32.png,
  plugin.xml,
  plugin.properties,
  icons/*.gif,
  ./
  META-INF/

  bin.excludes = overview.html, */doc-files/**

  source.. = src/
Agenda

- Introduction

- **PDE Build**
  - Exercise: Build a Feature
  - Exercise: Generate a p2 Repository

- Common Build Background

- Dash Athena
  - Exercise: Setup
  - Exercise: Configure, Run, Troubleshoot
  - Exercise: Verify Build

- Meet Hudson

- BYO Build Clinic
What is PDE Build?

- Ant script generator and build infrastructure
- Manages the compile time classpath based on OSGi dependencies
- The recommended way to build:
  - Plug-ins / Bundles
  - Features
  - RCP apps
- Eclipse itself is built using PDE Build
Overview of the builder

The build uses scripts provided by PDE Build, as well as script generated at build time.

- Fetch source code
- Compile .class files
- Assemble the results into archives.
What's new with PDE Build?

• 3.4
  • Required configuration files are reduced to one file: build.properties
  • Added customizability in the assembly and packaging phases of the build
  • Enhanced feature generation capabilities
  • Generation of individual source bundles
• 3.5
  • p2 integration
Exercise: Build a feature

Goals

- Show how to build features
- Demystify PDE Build
- Introduce key concepts
Relevant Part of PDE Build

Source code → eclipse build script generator

Assembly scripts

Uses plugin.xml and build.properties

Jars

Build scripts

Same scripts than with PDE-UI

Your eclipse based product .zip
Exercise: Build a feature

- Build the e4 XWT feature

- XWT is a collection of declarative bindings for SWT, used to generate SWT widgets from an XML file

- For setup, see slides 4-6. You will need:
  - Eclipse 3.5M6
  - JDK 5.0+
  - 2 XWT projects – XWT.Builder and XWT.BuildDirectory
Step 1: The Build Directory

- The source to be compiled is in the build directory, structured under features and plugins folder like in a normal Eclipse install.

- This is where the build will run. Generated scripts and build results will end up here.

Notes for releng:
- The location is defined by the `${buildDirectory}` ant property
- Make sure build directory and subdirectories are writeable
- This is where build will copy any source it fetches from CVS or other repositories.
Step 2: The Builder

The **builder** directory, aka the build configuration directory, contains the configuration files for the build.

- At a minimum, a build.properties file is required.
- Additional files can be placed here to customize the build.
- Templates of these files are provided in the templates folder of pde build.
- The example includes an extra buildXWT.xml for the convenience of running the build in the workspace.

**Notes for releng:**

- The location is defined by the `${builder}` ant property.
- Customization scripts (allElements.xml, customTargets.xml, customAssembly.xml) placed here will be used instead of the defaults – copy from pde build source and adjust as needed.
Step 3: Setting Some Properties

- The `build.properties` in the builder folder contains all the switches to configure the overall aspect of the build. The 4 properties that must be set for a build to run are:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>topLevelElementId</code></td>
<td>The feature id to be built</td>
</tr>
<tr>
<td><code>baseLocation</code></td>
<td>Folder containing the plugins to build against</td>
</tr>
<tr>
<td><code>buildDirectory</code></td>
<td>Folder containing the source to build (XWT.BuildDirectory)</td>
</tr>
<tr>
<td><code>builder</code></td>
<td>Folder containing the build configuration (XWT.Builder)</td>
</tr>
</tbody>
</table>

- The `builder` property is specified on the command line for the build.

In the example, we are only setting `topLevelElementId = org.eclipse.e4.xwt.feature`

For convenience we are setting the other properties using the buildXWT.xml ant script.

The xwt plugins use 1.5 asserts so we also set “javacSource” and “javacTarget” properties here.
Step 4: The buildXWT.xml Script

The script is very simple, it sets the properties from Step 3 that we did not put in the build.properties file.

- builder: `${basedir}` is the directory containing the script
- buildDirectory: Set to our `XWT.BuildDirectory` project, relative to the builder
- baseLocation: `${eclipse.home}` is the eclipse we are running in
- `${eclipse.pdebuild.scripts}/build.xml` is the main PDE Build script

```xml
<project default="main">
  <target name="main">
    <property name="builder" value="${basedir}"/>
    <property name="buildDirectory" value="${basedir}/../XWT.BuildDirectory"/>
    <property name="baseLocation" value="${eclipse.home}"/>

    <ant antfile="${eclipse.pdebuild.scripts}/build.xml"/>
  </target>
</project>
```

Note: `${eclipse.home}` and `${eclipse.pdebuild.scripts}` are automatically set by Eclipse, see the Ant Runtime Preferences. It is these two properties that make this kind of script a convenient way of running builds.
Step 5: Run The Build

Select the buildXWT.xml script and “Run As → Ant Build...”

On the JRE tab, make sure that “Run in the same JRE as the workspace” is selected.

Once the build is complete, refresh the XWT.BuildDirectory project to see the results.

Look in the new I.TestBuild folder.
Step 5a: Run The Build
(Command line version)

On the command line, you can start eclipse by running “java -jar” on the equinox launcher.

- -application : the org.eclipse.ant.core.antRunner application.
- -buildfile : the main build.xml script provided by PDE Build
- -Dbuilder : the location of your builder directory

The “baseLocation” and “buildDirectory” properties are normally specified in the builder's build.properties file. Though they can also be specified on the command line:

```
java -jar plugins\org.eclipse.equinox.launcher_<ver>.jar
   -application org.eclipse.ant.core.antRunner
   -buildfile plugins\org.eclipse.pde.build_<ver>\scripts\build.xml
   -Dbuilder=<XWT.Builder> -DbuildDirectory=<XWT.BuildDirectory>
   -DbaseLocation=C:\target\eclipse
```
<terminated> buildXWT.xml [Ant Build] /home/nboldt/eclipse

**properties:**

**init:**

**gather.log:**


[copy] Copying 1 file to /home/nboldt/eclipse/workspace-clean2/XWT.BuildDirectory

**properties:**

**init:**

**gather.log:**


[copy] Copying 1 file to /home/nboldt/eclipse/workspace-clean2/XWT.BuildDirectory

**update.feature:**

**all.children:**


[unzip] Expanding: /home/nboldt/eclipse/workspace-clean2/XWT.Builder

BUILD SUCCESSFUL
Total time: 2 seconds
Success

- We have an archive!
- See the resulting build
  XWT.BuildDirectory/I.TestBuild/
  org.eclipse.e4.xwt.feature-TestBuild.zip
- See the automatically generated scripts
  assemble.org.eclipse.e4.xwt.feature.all.xml
  assemble.org.eclipse.e4.xwt.feature.xml
  package.org.eclipse.e4.xwt.feature.all.xml
  package.org.eclipse.e4.xwt.feature.xml
- See the build log in
  XWT.Builder/buildlog.latest.txt
Exercise: Generate a p2 Repository

3.4 version: Add the following properties to the XWT.Builder build.properties file and run the build again to get p2 metadata.

```properties
generate.p2.metadata=true
p2.metadata.repo=file:${buildDirectory}/repo
p2.artifact.repo=file:${buildDirectory}/repo
p2.publish.artifacts=true
```

3.5M6 version: Add the following property

```properties
p2.gathering=true
```

And notice the resulting archive is now a p2 repository.

```plaintext
org.eclipse.e4.xwt.feature-TestBuild-group.group.group.zip
```
Test Installation of Built Feature

- For normal zip, unpack into ~/eclipse/dropins/ and restart Eclipse using File > Restart.
  - cd ~/workspace/XWT.BuildDirectory/I.TestBuild
  - unzip org.eclipse.e4.xwt.feature-TestBuild.zip \\ -d ~/eclipse/dropins/

- Check Help > About > Installation Details > Installed Software. Note the version number, eg., 0.9.0.v200903220728.
Test Installation of Built Repo

- Next, install from the archived p2 repo.

- Help > Install New Software... > Add

- Browse for\(\text{org.eclipse.e4.xwt.feature-TestBuild-group.group.group.zip}\)

- Install the XWT feature (you may need to uncheck the Group items by category checkbox).

- Check Help > About > Installation Details > Installed Software. Note the version number will have increased, eg., \textbf{0.9.0.v200903220742}. 
Available Software

Check the items that you wish to install

Work with: /l.TestBuild/org.eclipse.e4.xwt.feature-TestBuild-group.group.group.zip!

Go to the Available Software Sites preferences

type filter text

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse e4 XWT</td>
<td>0.9.0.200903220742</td>
</tr>
</tbody>
</table>

Details

Eclipse e4 XWT declarative SWT

Show only the latest versions of available software  Hide items that are already installed
Group items by category
Contact all update sites during install to find required software
Flow of the build

- `build.xml`
- `Script Generator`
- `com.acme.main.feature`
- `com.acme.ui`

**Generate**
- Discover all the plugins and analyze their dependencies.
- Read `feature.xml`
- Read `build.properties`
- Read `build.properties`
- Generate `assemble scripts`
- Generate `package scripts`
- Generate `acme.feature build.xml`
- Generate `acme.ui build.xml`

**Process**
- Assemble main target
- Invoke `build.xml`
- Invoke `build.xml`
- Invoke `build.xml`
- Invoke `build.xml`
- Invoke `build.xml`

**Generate**
- `assemble scripts`
- `package scripts`
- `acme.feature build.xml`
- `acme.ui build.xml`

**Invoke build**
- `build.xml`
- `build.xml`
- `build.xml`
- `build.xml`

---

Source code in this presentation is made available under the EPL, v1.0
The remainder of the presentation is licensed under Creative Commons Att. Nc Nd 2.5 license
Digging Deeper

- Explore the generated feature & plugin build.xml
  - all.plugins, build.jars
  - @dot, gather.bin.parts

- Import org.eclipse.pde.build into workspace
  - File > Import > Plug-in Development > Plug-ins and Fragments > org.eclipse.pde.build
  - Look in scripts/build.xml: main
  - Look in assemble*.xml and package*.xml
  - Look in allElements.xml

- PDE Build Templates
  - Look in templates/headless-build/customTargets.xml
Digging Deeper (cont'd)

- XWT.Builder/build.properties
  - RunPackager
  - allowBinaryCycles
  - flattenDependencies, parallelCompilation
    - New in Eclipse 3.5M6!
    - No longer need to worry about the order of subfeatures in a given feature.xml
  - outputUpdateJars, generateJnlp
  - forceContextQualifier = v20090323-0800
  - generateFeatureVersionSuffix = hash of contents
  - PluginPath
  - BREEs, bootclasspaths, javacSource, javacTarget
What else can PDE Build do?

- Download zip to use as a target
- Fetch map files and sources
- Hook in custom steps
- Generate source bundles/features
- Build products
Exercise Review

- Review structure of PDE Build process
- Review contents of build.properties, what they do, etc.
- Review important locations
- Locations and how they fit with Athena
Agenda

- Introduction
- PDE Build
  - Exercise: Build a Feature
  - Exercise: Generate a p2 Repository
- Common Build Background
- Dash Athena
  - Exercise: Setup
  - Exercise: Configure, Run, Troubleshoot
  - Exercise: Verify Build
- Meet Hudson
- BYO Build Clinic
Common Build Background

- Basebuilder and Releng Projects
- Common Build History
- Build Workshops
- Dash Athena Project
Basebuilder and Releng Projects

- **Basebuilder**
  - Subset of the Eclipse SDK (CVS, JDT, PDE) required to run PDE Build
  - Includes org.eclipse.build.tools and custom ant tasks

- **Releng Project**
  - Uses org.eclipse.releng.basebuilder
  - Defines build information
    - Project metadata (name, version)
    - Source locations & branches/tags (map file(s))
    - Optional extra packaging steps
    - Defines 3rd-party requirements
      - eg., Linux Tools requires CDT
Common Build History

• Started with one build, OCL, Nov 2005

• By July 2006, there were 10 divergent builds
  • 3rd party jars
  • multiple upstream dependencies (eg., Eclipse, EMF, UML2, OCL)
  • JDK™ 1.4 & 5.0

• Now, with Xtext, there are more than 20!
  • Some with as many as 10 upstream required builds

• Dash Athena Builder under development
  • 2 projects so far (Linux Tools, GEF)
Build Workshops

  - Lots of release engineers from Eclipse projects
  - More talk than walk

  - Nick, Bjorn, Denis, et al
  - Athena project started with first 3 committers
  - GEF build is ported to run headless on build.eclipse.org
Build Workshops (cont'd)

  - Nick, Andrew O, Andrew N, Kim, Pascal, et al
  - Athena grows to 6 committers
  - Releng project greatly simplified

  - Nick, Andrew O, Bjorn
  - Support for building in Eclipse on Linux, Mac and Windows
  - Support for building, testing, and jar signing via Hudson

- **Ongoing**
Dash Athena Project

- **What it is**
  - A work in progress!
  - A wrapper for PDE Build using org.eclipse.releneg.basebuilder
  - Ant tasks backed by some shell/exec calls (less over time)

- **What it does**
  - Standardizes paths and variables
  - Simplifies default offering, while allowing for customizations
  - Uses Ant Contrib, Ant4Eclipse, and PDE SVN Plugin

- **Where it runs**
  - In Eclipse (Windows, Mac, Linux 32/64)
  - In Hudson & via commandline (Linux 32/64)
Agenda

- Introduction
- PDE Build
  - Exercise: Build a Feature
  - Exercise: Generate a p2 Repository
- Common Build Background
- Dash Athena
  - Exercise: Setup
  - Exercise: Configure, Run, Troubleshoot
  - Exercise: Verify Build
- Meet Hudson
- BYO Build Clinic
Exercise: Setup
(Network-based config; skip if projects already in workspace)

1. Download Team Project Set File (PSF):
   http://dev.eclipse.org/viewcvs/index.cgi/org.eclipse.dash/athena/
   org.eclipse.dash.commonbuilder/org.eclipse.dash.common.releng.tutorial/
   psf/gef.psf?revision=1.1.2.1&root=Technology_Project

2. Import PSF using File > Import > Team > Team Project Set

3. You should now have these projects:
   - org.eclipse.releng.basebuilder
   - org.eclipse.dash.common.releng
   - org.eclipse.gef.releng

4. Download ant-contrib-1.0b2 (not 1.0b3!):
   http://downloads.sourceforge.net/ant-contrib/ant-contrib-1.0b2-bin.zip

5. Unpack it, and put the renamed jar here:
   - org.eclipse.dash.common.releng/lib/ant-contrib.jar
Building In Eclipse Using Athena

● Modify properties for your machine
  ● Paths for Java, Ant, workspace dir, build dir

● Use local source checkout if you prefer
  ● Subsequent builds are faster, consume less bandwidth

● Right-click releng project's build.xml
  ● Run As > Ant Build

● Console log stored in releng project
  ● buildlog.latest.txt
Exercise: Configure & Run

- You should already have the example project
  org.eclipse.gef.releng checked out from
  Athena repository (see previous exercise)

- Open build.properties

- Right-click releng project's build.xml
  Run As > Ant Build

- Check console log output in
  buildlog.latest.txt
Athena build.properties

- What & how to build:
  projectid=tools.gef
  zipPrefix=GEF
  version=3.5.0
  buildType=N
  mainFeatureToBuildID=org.eclipse.gef.all
  testFeatureToBuildID=org.eclipse.gef.test
  
  JAVA_HOME=/path/to/java
  JAVA14_HOME=/path/to/java
  JAVADOC14_HOME=/path/to/java/bin
  
  dependencyURLs=http://download.eclipse.org/
    eclipse/downloads/drops/
    S-3.5M5-200902021535/
    eclipse-SDK-3.5M5-linux-gtk.tar.gz
Athena build.properties (cont'd)

- Where to get sources:
  - If building from CVS or SVN, use map file(s)

- If building locally, use:
  ```bash
  localSourceCheckoutDir=\C:/workspace/org.eclipse.gef.tree
  ```

- To get local cache, import using `gef-tree.psf`

- Or via command line:
  ```bash
  cd ~/workspace; cvs -d :pserver:anonymous@dev.eclipse.org:~/cvsroot/tools -q co -d org.eclipse.gef.tree \org.eclipse.gef
  ```
Troubleshooting

- Missing classpath or bootclasspath entries
  - ant-contrib.jar (wrong version!)
  - org.eclipse.pde.build.svn_*.jar
  - dt.jar, rt.jar, classes.jar

- Map file problems
  - Missing file? Missing entries in file?
  - Wrong feature or plugin id to path mapping?
  - Invalid cvs/svn credentials?

- Testing problems
  - Need Xvnc or Xvfb to run headless tests
  - Missing org.eclipse.test or org.eclipse.ant.optional.junit?
  - Note: tests do not currently run headless on Mac or Windows
Troubleshooting (cont'd)

- Building from SVN sources
  - Need an SVN client installed; PDE SVN does not include its own SVN implementation

- Platform-specific configuration tips

  http://wiki.eclipse.org/Common_Bulid_Infrasturcture/Virtual_Server_Setup/Fedora
  http://wiki.eclipse.org/Common_Bulid_Infrasturcture/Virtual_Server_Setup/MacOSX
  http://wiki.eclipse.org/Common_Bulid_Infrasturcture/Virtual_Server_Setup/WinXP
thirdPartyLicenseAcceptance="I accept"

projectId=tools.gif
sipPrefix=33F
incubation=
version=3.5.0
buildType=N
mainFeatureToBuildId=org.eclipse.gif.all
testFeatureToBuildId=org.eclipse.gif.test
build.steps=buildJavaTests, buildTests, buildUpdate, generateDigests, publish, cleanup

localSourceCheckoutDir=c:/workspace/org.eclipse.gif.tree

dependencyURLs=http://download.eclipse.org/eclipse/downloads/drops/S-3.5M-2/0

javac HOME="C:/Programs/Java/jdk1.6.0_11"
javac HOME="C:/Programs/Java/jdk1.6.0_11"
javac HOME="C:/Programs/Java/jdk1.6.0_11/bin"

# For windows, must be explicit about paths using correct slashes (/); don't use \.

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}

[javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 2 files to C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 9 files to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Copying 1 file to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting directory C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Run p2 metadata generator using launcher.jar - C:\workspace\org.eclipse.gif.tree
[javac] Generating metadata for C:\tmp\build\N0009302191\eclipse\p2\metadata
[javac] Generation completed with success [1 seconds].
javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] buildAll.java#run :: build.step::generateDigests

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}

[javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 2 files to C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 9 files to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Copying 1 file to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting directory C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Run p2 metadata generator using launcher.jar - C:\workspace\org.eclipse.gif.tree
[javac] Generating metadata for C:\tmp\build\N0009302191\eclipse\p2\metadata
[javac] Generation completed with success [1 seconds].
javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] buildAll.java#run :: build.step::generateDigests

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}

[javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 2 files to C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 9 files to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Copying 1 file to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting directory C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Run p2 metadata generator using launcher.jar - C:\workspace\org.eclipse.gif.tree
[javac] Generating metadata for C:\tmp\build\N0009302191\eclipse\p2\metadata
[javac] Generation completed with success [1 seconds].
javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] buildAll.java#run :: build.step::generateDigests

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}

[javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 2 files to C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 9 files to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Copying 1 file to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting directory C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Run p2 metadata generator using launcher.jar - C:\workspace\org.eclipse.gif.tree
[javac] Generating metadata for C:\tmp\build\N0009302191\eclipse\p2\metadata
[javac] Generation completed with success [1 seconds].
javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] buildAll.java#run :: build.step::generateDigests

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}

[javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 2 files to C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 9 files to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Copying 1 file to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting directory C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Run p2 metadata generator using launcher.jar - C:\workspace\org.eclipse.gif.tree
[javac] Generating metadata for C:\tmp\build\N0009302191\eclipse\p2\metadata
[javac] Generation completed with success [1 seconds].
javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] buildAll.java#run :: build.step::generateDigests

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}

[javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 2 files to C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 9 files to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Copying 1 file to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting directory C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Run p2 metadata generator using launcher.jar - C:\workspace\org.eclipse.gif.tree
[javac] Generating metadata for C:\tmp\build\N0009302191\eclipse\p2\metadata
[javac] Generation completed with success [1 seconds].
javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] buildAll.java#run :: build.step::generateDigests

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}

[javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 2 files to C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Copying 9 files to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Copying 1 file to C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] Deleting directory C:\tmp\build\N0009302191\eclipse\test.assembly
[javac] Run p2 metadata generator using launcher.jar - C:\workspace\org.eclipse.gif.tree
[javac] Generating metadata for C:\tmp\build\N0009302191\eclipse\p2\metadata
[javac] Generation completed with success [1 seconds].
javac] Building sip: C:\tmp\build\N0009302191\eclipse\test.assembly\test
[javac] buildAll.java#run :: build.step::generateDigests

build.properties

buildJavaTests:
buildCopy:
buildJava:
buildUpdate:
buildSite:
build}
java - org.eclipse.gef.releng/build.properties - Eclipse SDK - /Users/nickboldt/Documents/workspace

Build.properties

# TODO: could be relative, absolute (local), or http paths to archives
#dependencyURLs=http://download.eclipse.org/eclipse/downloads/drops/S-3.5M-200902011535/eclipse-SDK-3.5M-win32.tar.gz
#dependencyURLs=http://download.eclipse.org/eclipse/downloads/drops/S-3.5M-200902011535/eclipse-SDK-3.5M-linux-gtk.tar.gz
#dependencyURLs=http://download.eclipse.org/eclipse/downloads/drops/S-3.5M-200902011535/eclipse-SDK-3.5M-linux-gtk-x86_64.tar.gz
#dependencyURLs=http://download.eclipse.org/eclipse/downloads/drops/S-3.5M-200902011535/eclipse-SDK-3.5M-linux-gtk-ppc.tar.gz
#dependencyURLs=http://download.eclipse.org/eclipse/downloads/drops/S-3.5M-200902011535/eclipse-SDK-3.5M-macosx-carbon.tar.gz

#projRelengRoot=pserver.anonymous@dev.eclipse.org/cvsroot/tools
#projRelengRoot=pserver.anonymous@dev.eclipse.org/cvsroot/technology
projRelengRoot=org.eclipse.dash/athena/org.eclipse.dash.commonbuilder/org.eclipse.gef.releng

basebuilderBranch=R35_M5

# build with which JDK? see e.e.d.commonbuilder.releng/server.properties for defined paths if not explicitly set here (eg., to build
# If only building in Eclipse, may want to set JAVA_HOME=$[java.home] (use Eclipse's default JVM as defined in Ant)
# If building via commandline or in Hudson, better to set absolute paths, eg., JAVA_HOME=/usr/lib/jvm/java
JAVA_HOME=/System/Library/Frameworks/JavaVM.framework/Home
JAVA14_HOME=/System/Library/Frameworks/JavaVM.framework/Home
# used by doc.jsr builders; not all builds will need this
JAVADOCC_HOME=/System/Library/Frameworks/JavaVM.framework/Home/bin
compilerArg=-enableJavadoc -encoding ISO-8859-1

# If building on Windows, may want explicit short-paths
JAVA_HOME=C:/Program-1/Java/jdk1.6.0_11
JAVA14_HOME=C:/Program-1/Java/jdk1.6.0_11
JAVADOCC_HOME=C:/Program-1/Java/jdk1.6.0_11/bin

# for windows, must be explicit about paths using correct slashes (/); dirs with spaces should be avoided
relengBuilderDir=C:/workspace/org.eclipse.gef.releng
relengBasebuilderDir=C:/workspace/org.eclipse.gef.releng_basebuilder

<terminated build.xml>
Exercise: Verify Build

- Exercise: Install GEF SDK from Archived p2 Repository (an update site with metadata)
  - Help > Install New Software... > Add > Archive...
  - Browse for GEF-Update-*.zip
  - Uncheck 'Group items by category' checkbox
  - Select GEF SDK feature
  - Restart when prompted
  - Verify GEF SDK is installed
    Help > About > Installation Details

- Exercise: Install from SDK zip (Optional)
  - Uninstall previously installed GEF SDK; restart when prompted
  - Verify GEF is no longer installed. Close Eclipse
  - Unpack GEF SDK zip into eclipse/dropins/gef/
  - Restart Eclipse
  - Verify GEF SDK is installed
    Help > About > Installation Details
Exercise:
Install GEF SDK
From Archived p2 Repo
Exercise: Verify Build

- Exercise: Smoke Test (GEF Logic Example)
  - Install GEF SDK as in one of previous exercises
  - File > New > Example... > Logic
  - Select generated project
  - Run > Run As > Eclipse Application

- In second Eclipse instance
  - New > Project... > General > Project
  - New > Example... > GEF > Logic Diagram
  - Select Four-bit Adder Model
  - Review generated diagram
Exercise: Smoke Test
(GEF Logic Example)

Select a wizard
Create a plug-in project that defines a logic editor.

Wizards:
- Class
- Interface
- Java Project
  - Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- CVS
- Java
- Plug-in Development
- User Assistance
- Examples
  - Flow
  - Logic
  - Shapes
- WYSIWYG Document

Source code in this presentation is made available under the EPL, v1.0
The remainder of the presentation is licensed under Creative Commons Att. Nc Nd 2.5 license
Exercise: Smoke Test
(gef Logic Example)
(cont'd)
Exercise: Verify Build

Exercise: Smoke Test (GEF Shapes) (Optional)
- Install GEF SDK as in one of previous exercises
- File > New > Example... > Shapes
- Select generated project
- Run > Run As > Eclipse Application

In second Eclipse instance
- New > Project... > General > Project
- New > Example... > GEF > Shapes Diagram
- Open palette on right side of editor; create a diagram
Exercise: Smoke Test (GEF Shapes Example)
Agenda

- Introduction
- PDE Build
  - Exercise: Build a Feature
  - Exercise: Generate a p2 Repository
- Common Build Background
- Dash Athena
  - Exercise: Setup
  - Exercise: Configure, Run, Troubleshoot
  - Exercise: Verify Build
- Meet Hudson
- BYO Build Clinic
Meet Hudson

- Continuous Integration (build when sources change)
- Change Tracking (CVS, SVN, ...)
- Automated Jar Signing
- Extensible Framework (Plug-ins)
- Parameterized Builds, Chained Builds

https://hudson.dev.java.net/
### Build Groups & Chained Builds

<table>
<thead>
<tr>
<th>S</th>
<th>W</th>
<th>Job</th>
<th>Last Success</th>
<th>Last Failure</th>
<th>Last Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>cbi-admin</td>
<td>15 days (#2)</td>
<td>N/A</td>
<td>0.63 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cbi-gef-3.4.x-nightly</td>
<td>22 days (#61)</td>
<td>23 days (#58)</td>
<td>14 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cbi-gef-3.5.x-nightly</td>
<td>8 hr 8 min (#21)</td>
<td>8 hr 9 min (#20)</td>
<td>13 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cbi-linuxtools-0.2.x-Galileo-nightly</td>
<td>2 days 9 hr (#62)</td>
<td>9 hr 20 min (#63)</td>
<td>25 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cbi-linuxtools-0.2.x-Ganymede-nightly</td>
<td>9 hr 0 min (#59)</td>
<td>2 days 13 hr (#55)</td>
<td>19 min</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>W</th>
<th>Job</th>
<th>Last Success</th>
<th>Last Failure</th>
<th>Last Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>galileo.build</td>
<td>21 hr (#141)</td>
<td>6 hr 12 min (#156)</td>
<td>1 hr 12 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>galileo.fixperm</td>
<td>8 hr 57 min (#110)</td>
<td>N/A</td>
<td>10 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>galileo.generate</td>
<td>6 hr 12 min (#160)</td>
<td>10 hr (#155)</td>
<td>29 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>galileo.promote</td>
<td>14 days (#35)</td>
<td>2 mo 10 days (#15)</td>
<td>3 min 40 sec</td>
</tr>
</tbody>
</table>
Summary

1. [268824] gef-head ahunter 090316 This site has moved. Please update your features.
2. v20090316-1158

Revisiion 21716 by ebaron:
Silence push/popd look for -docs pattern

Revision 21717 by ebaron:
Updated linuxtools.psf. Added shell script for creating local source checkout directory.
Parameterized Builds
Build Duration Trending
Running Hudson

• Download the latest hudson.war or .jar from https://hudson.dev.java.net/

• java -jar hudson.war

• (That's it!)
Agenda

- Introduction
- PDE Build
  - Exercise: Build a Feature
  - Exercise: Generate a p2 Repository
- Common Build Background
- Dash Athena
  - Exercise: Setup
  - Exercise: Configure, Run, Troubleshoot
  - Exercise: Verify Build
- Meet Hudson
- BYO Build Clinic
BYO Build Clinic / Q&A

- Bring us your checked out sources, we'll try to get you running w/ an Athena build

- Any questions?

- Want more? For extra credit, see the next two slides.
Exercise: Linux Tools (Optional)

- Try building the Linux Tools project!
- Get org.eclipse.linuxtools.releng project
  - See psf/linuxtools.psf file in tutorial zip
- Source tree can be fetched from
- Configure releng/build.properties to include required dependencies: JDK 6.0, Eclipse, EMF, GEF, CDT & BIRT.
- Run build.xml!
Exercise: Build in Hudson (Optional)

- Try building with Hudson, either locally or on a virtual server. See http://wiki.eclipse.org/Common_Build_Infrastructure/Virtual_Server_Setup
- Start Hudson. Open http://localhost:8080
- Create a new job
- Launch a GEF or Linux Tools build using
  - org.eclipse.dash.common.releng/tools/scripts/start.sh
  - (Run script without options for examples and summary of commandline flags)