Open XAL Project Architecture

Thomas Pelaia II, Ph.D.

Open XAL Meeting
November 14, 2013
Motivation

- Common Core is too large
- Need support for Site Specific Extensions and Plugins
- Need mechanism for assembling and sharing a project
  - Site specific applications, services, extensions and plugins
  - IDE Configuration
- Identity Preservation and Independence
Goals

• New features
  – Support Extensions and Plugins
  – Smaller Common Core
  – Mechanism for assembling and sharing projects

• Maintain
  – Simple Command Line Build
  – Clean layout
# Primary Constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Launched by user, Graphical Interface</td>
</tr>
<tr>
<td>Service</td>
<td>Runs continuously, headless, includes extension</td>
</tr>
<tr>
<td>Core</td>
<td>Common Open XAL library</td>
</tr>
<tr>
<td>Extension</td>
<td>Optional addition to the Core, Core has no dependency</td>
</tr>
<tr>
<td>Plugin</td>
<td>One of each type of plugin is required by Core at runtime</td>
</tr>
</tbody>
</table>
Extension

• May depend on core, extensions and plugins
• Core has no dependency on extensions
• Apps and Services may depend on extensions
• May include libraries, resources and source code
  – Libraries should be completely wrapped (e.g. jmdns)
• Two types
  – Pure
  – Service
Pure Extension

• Placed under top level extensions directory

• Package prefix: xal.extension.<extension-name>

• Several existing packages became extensions
  – All extension package names changed
  – Some packages also got reorganized
  – Applications modified due to package changes
Service Extension

• Associated with a service

• A service’s protocol is an extension

• A service’s other supporting code may be an extension

• Placed at extensions directory under its service’s directory

• Package prefix: xal.service.<service-name>
Plugin

- May depend on core, extensions and plugins
- Core has runtime only dependency on plugin families
- Apps and Services may depend directly on plugins
- May include libraries, resources and source code
  - Libraries should be completely wrapped (e.g. jca, caj)
- Two types
  - Solitary
  - Family Member
Solitary Plugin

• Only one plugin for a given family may be included

• Core references a plugin family class to be implemented by just one plugin (e.g. channel factory)

• Two source code package prefixes to supply
  – xal.<core package tree>
  – xal.plugin.<plugin-name>

• Example JCA Plugin
  – Virtual Accelerator changed to use plugin instead of underlying CAJ calls
Family Member Plugin

• Multiple plugins for a given family may be included

• Core references a plugin family indirectly through configuration files (e.g. database configuration)

• Only one source code package prefix to supply
  – xal.plugin.<plugin-name>

• Database Adaptor Plugins
  – oracle, mysql
  – change database config file for adaptor class matching new package name
Build Phases and Spaces

Core

Extensions
Service Extensions
Plugins

Service
...

Application
...

Managed by UT-Battelle
for the Department of Energy

Thursday, November 14, 13
Project Layout

apps
build
build.xml
config
core
extensions
plugins
scripts
services
test
Project Layout - Core

apps
build
build.xml
config
core
extensions
plugins
scripts
services
test

build.xml
lib
resources
src
test
Project Layout - Extensions

apps
build
build.xml
config
core
extensions
plugins
scripts
services
test

application
bricks
build.xml
extlatgen
fit
orbit
scan
service
solver
widgets

Managed by UT-Battelle for the Department of Energy

Thursday, November 14, 13
Project Layout - Extensions

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test
- application
- bricks
- build.xml
- extlatgen
- fit
- orbit
- scan
- service
- solver
- widgets
- lib
- src

Managed by UT-Battelle for the Department of Energy
Project Layout - Plugins

apps
build
build.xml
config
core
extensions
plugins
scripts
services
test

Readme.html

jca
mysql
oracle

lib
src
xal
ca
plugin

Managed by UT-Battelle for the Department of Energy

Thursday, November 14, 13
Project Layout - Plugins

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test
- Readme.html

- jca
- mysql
- oracle

- lib

- src

- xal

- ca

- plugin

- ChannelFactoryPlugin.java
Project Layout - Services

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test

- build.xml
- common.xml
- pvlogger
- worker
Project Layout - Services

apps
build
build.xml
config
core
extensions
plugins
scripts
services
test

build.xml
common.xml
pvlogger
worker
extension
src

build.xml
Ant Build System Changes

- New *intermediates* build subdirectory
- Build *jars* renamed *products*
- Single shared library combining core, extensions and plugins
- Javadoc - entire shared library
- *jar-resources* target for each build file constructing corresponding *resources.jar* intermediate
- JUnit libraries moved from core/test to top level test
Build Directory Hierarchy

- Build
  - dist
    - doc
      - intermediates
      - products
Build Directory Hierarchy - Products

- Build
  - dist
    - doc
    - intermediates
    - products
  - apps
    - lib
    - scripts
    - services
Build Directory Hierarchy - Intermediates

- Build
  - dist
  - doc
  - intermediates
  - products
  - apps
    - core
    - extensions
    - services
    - shared
Code Sharing Goals

• Share common core, applications, services, extensions and plugins

• Pick and choose among other components

• Add site specific components

• Maintain reference base for certifying code and quick start
Code Sharing Proposal

• Git Subtree seems well suited
  – Each component may reside in its own branch and repository
    • Better history management
    • Easier task distribution
  – Easily push and pull components

• Complete projects including IDE support can also be maintained
Git Subtree Mechanism

• Normal Git Repository (nothing special)

• Commands push/pull specified subdirectory files and commits to/from another repository and branch

• Branch commits truncated appropriate to subtree
Common Git Subtree Commands

Add files and commits to the specified path from the specified repository branch:

```bash
git subtree add -P <path> <repository> <branch>
```

Pull files and commits to the specified path from the specified repository branch:

```bash
git subtree pull -P <path> <repository> <branch>
```

Push files and commits at the specified path to the specified repository branch:

```bash
git subtree push -P <path> <repository> <branch>
```

Reference:

Generic Open XAL Git Subtrees

- **Project**
  - **Base**
    - **Extensions**
      - Extension A
      - Extension B
    - **Plugins**
      - Plugin A
      - Plugin B
    - **Apps**
      - App A
      - App B
    - **Services**
      - Service A
      - Service B
# Current Repository Modules

<table>
<thead>
<tr>
<th>Type</th>
<th>Repository Prefix</th>
<th>Examples</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>project</td>
<td>xcode</td>
<td>IDE Specific</td>
</tr>
<tr>
<td>Base</td>
<td></td>
<td>openxal</td>
<td>Ant based</td>
</tr>
<tr>
<td>Core</td>
<td></td>
<td>core</td>
<td>Stable, Common</td>
</tr>
<tr>
<td>Extension</td>
<td>extension</td>
<td>application, bricks, extlatgen, fit, orbit, scan, service, solver, widgets</td>
<td>Site specific additions</td>
</tr>
<tr>
<td>Plugin</td>
<td>plugin</td>
<td>jca, mysql, oracle</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>app</td>
<td>bricks, ddbrowser, extlatgenerator, knobs, launcher, mtv, opticseditor, opticsswitcher, orbitcorrect, pvhistogram, pvlogger, scan1d, scan2d, scope, virtualaccelerator, xyzcorrelator</td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>service</td>
<td>pvlogger, worker</td>
<td></td>
</tr>
</tbody>
</table>
Code Sharing Workflow Example
Clone Existing Project

# clone the existing project with the default components
```
git clone ssh://YourID@git.code.sf.net/p/xaldev/project.xcode openxal-xcode
```

# work on a new branch
```
git checkout -b site.sns.master
```

Directory Structure:

- Open XAL.xcodeproj
- openxal
- utils

- Open XAL Base
- Supporting Xcode Scripts
Code Sharing Workflow Example
Add Energy Manager Application

# define remote to application for convenience
```
git remote add sf.energymanager https://YourID@git.code.sf.net/p/xaldev/app.energymanager
```

# define remote to Open XAL base
```
git remote add sf.openxal https://YourID@git.code.sf.net/p/xaldev/openxal
```

# add the Energy Manager application from the remote to our project
```
git subtree add -P openxal/apps/energymanager sf.energymanager master
```

# Push our new branch back to the project repository
```
git push origin site.sns.master
```

# Push the base back to remote repository with a new branch: site.sns.master
```
git subtree push -P openxal sf.openxal site.sns.master
```
Code Sharing Workflow Example
Pushing Energy Manager Changes Back

After making site specific changes to the Energy Manager application:

```
# Push changes back to the project repository
git push origin site.sns.master

# Push the base back to remote repository with a new branch: site.sns.master
git subtree push -P openxal sf.openxal site.sns.master

# Push the application back to the repository with new branch: site.sns.master
git subtree push -P openxal/apps/energymanager sf.energymanager site.sns.master
```

Now there are three remote repositories with site specific branches:
- project.xcode
- openxal
- app.energymanager
Future Related Tasks

• Complete support for unit tests beyond core
• Add a top level directory for samples (scripts and Java source code)