

Creating a first uAAL application

uAAL consortium

Creating a first uAAL application

uAAL consurtium

Copyright © 2011 universAAL

Table of Contents

1. Introduction	1
Download installer	1
2. Install uAAL Development Environment	2
Description	2
3. AAL Studio overview and installation	5
Role and benefits of this tool	5
Overview of functionality	5
Prerequisites and dependencies	5
Tool installation procedure	5
4. Project and Item Wizards	7
Role and benefits of this tool	7
Overview of functionality	7
Prerequisites and dependencies	10
Tool installation procedure	10
5. Service/application project build	11
Role and benefits of this tool	11
Overview of functionality	11
Prerequisites and dependencies	13
Tool installation procedure	13

List of Figures

2.1. uAAL setup.	2
2.2. Installation options.	3
2.3. Licenses.	3
2.4. Select folder for the installation.	4
2.5. Summary	4
3.1. Add Repository	6
4.1. Wizard:Artifact information	8
4.2. Wizard:Activator package	9
4.3. Create a new item	10
5.1. Build,run,debug a uAAL project	11
5.2. Successfule build	12
5.3. Successfule build	12
5.4. Typing ps command	12
5.5. List of installed and active bundles	12
5.6. Start bundle 6	13

Chapter 1. Introduction

Download installer

The first prototype is available. It is one package with size of approximately 430 MB and supports both 32 and 64 Bit platforms. The instALL is a NSIS installer script. It can be downloaded from:

- 64-bit version [<ftp://ftp.igd.fraunhofer.de/outgoing/cstocklo/instaal-64.exe>]
- 32-bit version [<ftp://ftp.igd.fraunhofer.de/outgoing/cstocklo/instaal-32.exe>]

The above links are valid for 90 days since 07/05/2011.

Chapter 2. Install uAAL Development Environment

Description

The installation process takes a few minutes. It provides a single setup program to automate this step. The installer (NSIS script) is responsible for the download, installation and configuration of the following components:

- JDK
- Maven
- TortoiseSVN / SlikSVN
- Eclipse
- Eclipse Plugins
- MySQL

The following screen dumps illustrate the steps of instALL installation.

Figure 2.1. uAAL setup.

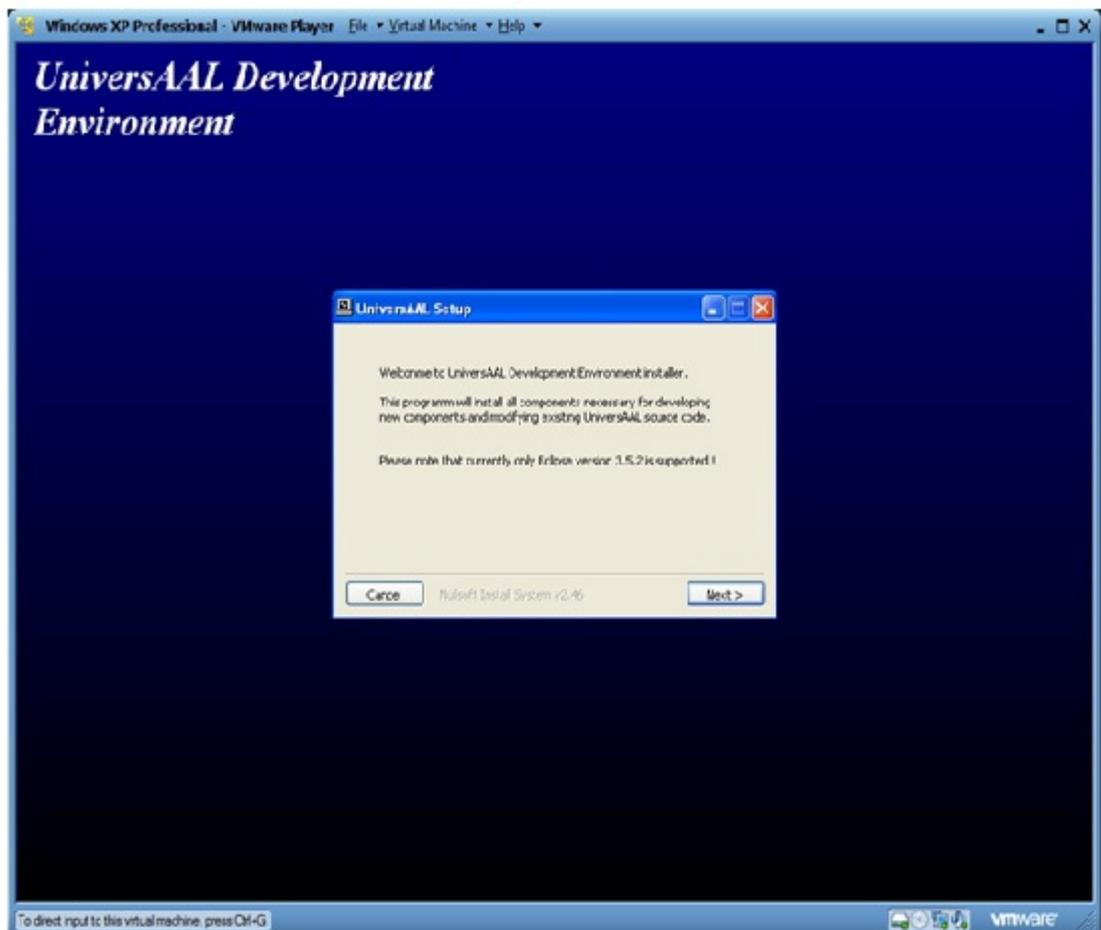


Figure 2.2. Installation options.

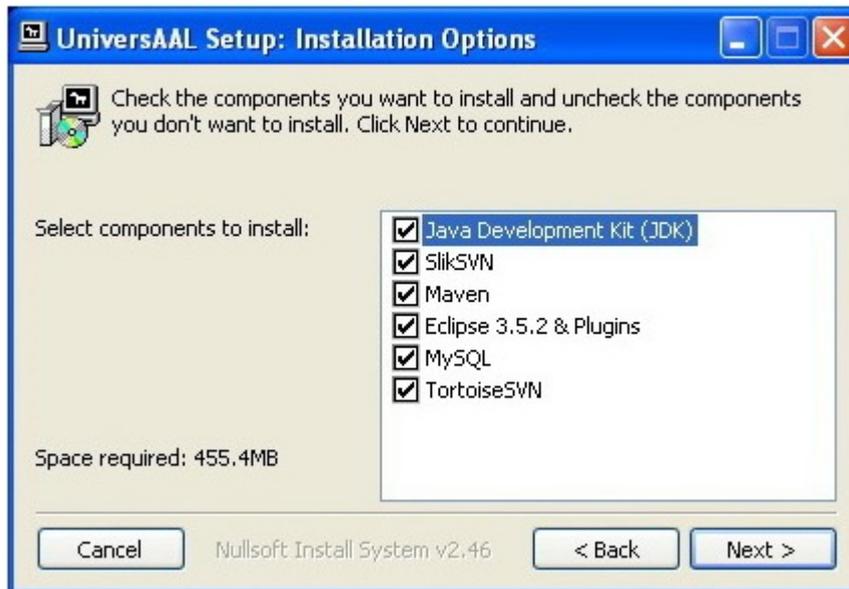


Figure 2.3. Licenses.



Figure 2.4. Select folder for the installation.

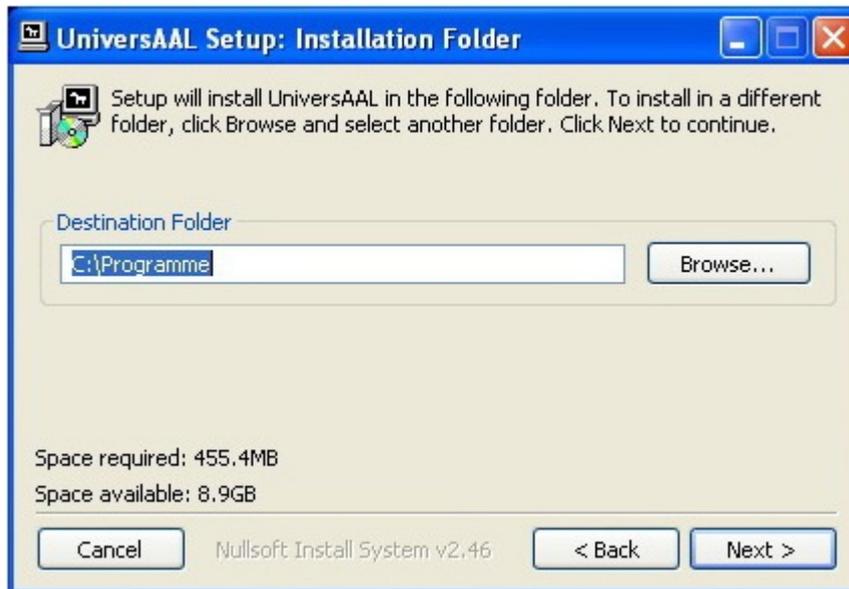
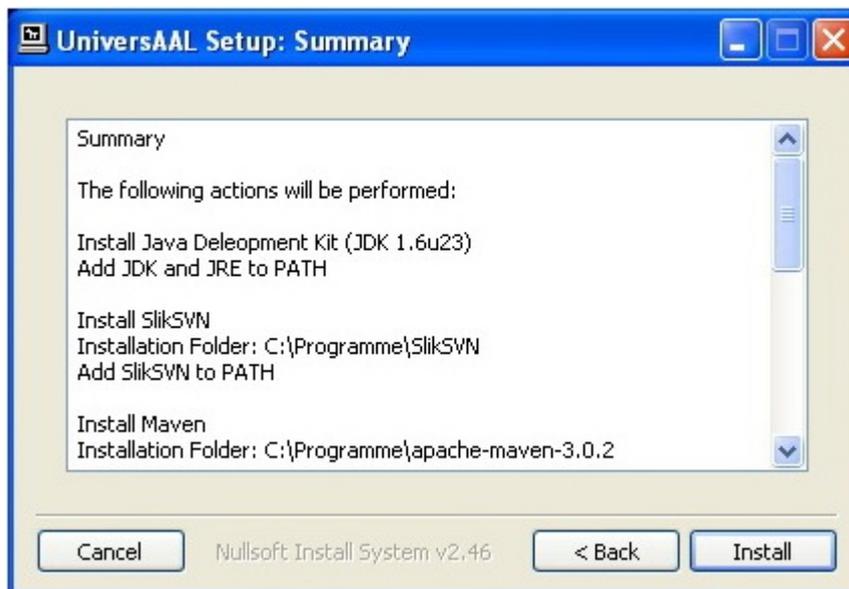


Figure 2.5. Summary



Chapter 3. AAL Studio overview and installation

Role and benefits of this tool

The AAL Studio provides an integrated development environment based on Eclipse for building applications and components using the universAAL execution platform. The AAL Studio will make it easier to get started with the development, and will make some of the development tasks more efficient. Also, it will give easy access to the resource needed by the developer. The client side developer tools created by universAAL are implemented as Eclipse plug-ins and provide integration with other AAL Studio tools. Thus, while development of universAAL compliant applications and components do not require a specific development Java development environment, use of the Eclipse-based AAL Studio is recommended because it gives access to using the provided tools.

Overview of functionality

The functionality of the AAL Studio is provided by the individual tools that are installed within it, including wizards for creating projects, build tools for simplifying building and launching of applications, and modeling and transformation tools for making the development more efficient. For further details of the functionality, see the description on the wiki page for each tool.

Prerequisites and dependencies

The AAL Studio is based on Eclipse, so the primary prerequisite for installing it is to have a compatible Eclipse version installed. The version we currently recommend is Eclipse 3.5 Modeling Tools, which can be downloaded from:

<http://www.eclipse.org/downloads/packages/release/galileo/sr2>

After installing Eclipse, you will also need some prerequisite Eclipse plugins before installing the universAAL tools. These are listed in detail under each of the tools. A quick summary of what is needed to install the full set of tools:

<http://m2eclipse.sonatype.org/sites/m2e/0.10.2.20100623-1649/>

- m2Eclipse maven support from:

This tool is required by both the wizard and build tools of the AAL Studio

<http://www.ops4j.org/pax/eclipse/update>

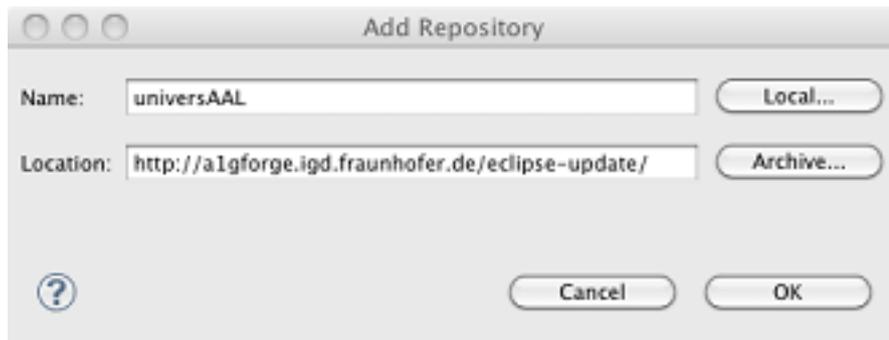
- Pax Runner for Eclipse:

This tool is required by the build tool of the AAL Studio in order to support launching of the application.

Tool installation procedure

The Eclipse Update Site for universAAL packages the AAL Studio tools for easy downloading and installation from Eclipse. The update site can be accessed from the built-in functionality of Eclipse. The update site is available at: <http://a1gforge.igd.fraunhofer.de/eclipse-update/> Note that we are intending to move to Eclipse 3.6 Modeling Tools at a later stage. To install the universAAL tools, add the address of the update site to the list of update sites in Eclipse. In version 3.6 of Eclipse, this can be done by first selecting "Install New Software..." under the "Help" menu. In the dialog box that appears, select "Add..." at the top right, and fill in the dialog box that appears. Installation for Eclipse 3.5 is similar.

Figure 3.1. Add Repository



Chapter 4. Project and Item Wizards

Role and benefits of this tool

This AAL Studio tool is intended to be used by developers of services and platform components. It makes it easy to create new universAAL-compliant projects by providing a skeleton project with all the files you need and initial content to make the project work in universAAL. The item wizards generate new files required or optional to a universAAL project with the proper formatting and template or initial content. It reduces the time of development since without this tool a developer would need to give the project or files the appropriate format to be universAAL-compliant, with the risk of missing some requirement. By using the wizards it is assured to have well-formatted files and project structures. The usage of this tool is not mandatory but recommended. The wizard currently provide setup and configuration options which are useful for typical development projects. Future versions may add further options that also simplify more advanced setup and configuration.

Overview of functionality

The main access to Project and Item Wizards is through the File/New command in Eclipse. Then the proper wizard must be selected. An Eclipse command is available to start the wizards from any menu or other plugin.

This plug-in provides wizards for creating universAAL resources in the Eclipse workspace. They can be either items or new projects. With “items” we refer to new files or other resources inside an existing project in the workspace, such a new XML file that could be necessary for a universAAL project. These new files would be generated with some initial content in order to be properly formatted, as a template, or configured as defined in the parameters passed in the wizard.

On the other hand, the Project wizard would generate a blank new universAAL compliant project in the workspace, ready to develop universAAL applications or components upon. It could be initially configured as described during the wizard, that is, preset configuration parameters, storage location, initial files... The wizard will take care of generating the project hierarchy and structure, and all the mandatory files for the project to be universAAL compliant, as well as any additional file that could be defined by the user in the wizard. This "New Project" wizard does always the same: creating a new project in the workspace, based on a very simple set of configurable parameters (Maven information and initial files).

Figure 4.1. Wizard:Artifact information

Artifact Information

Type a valid Group identifier

Group Id:

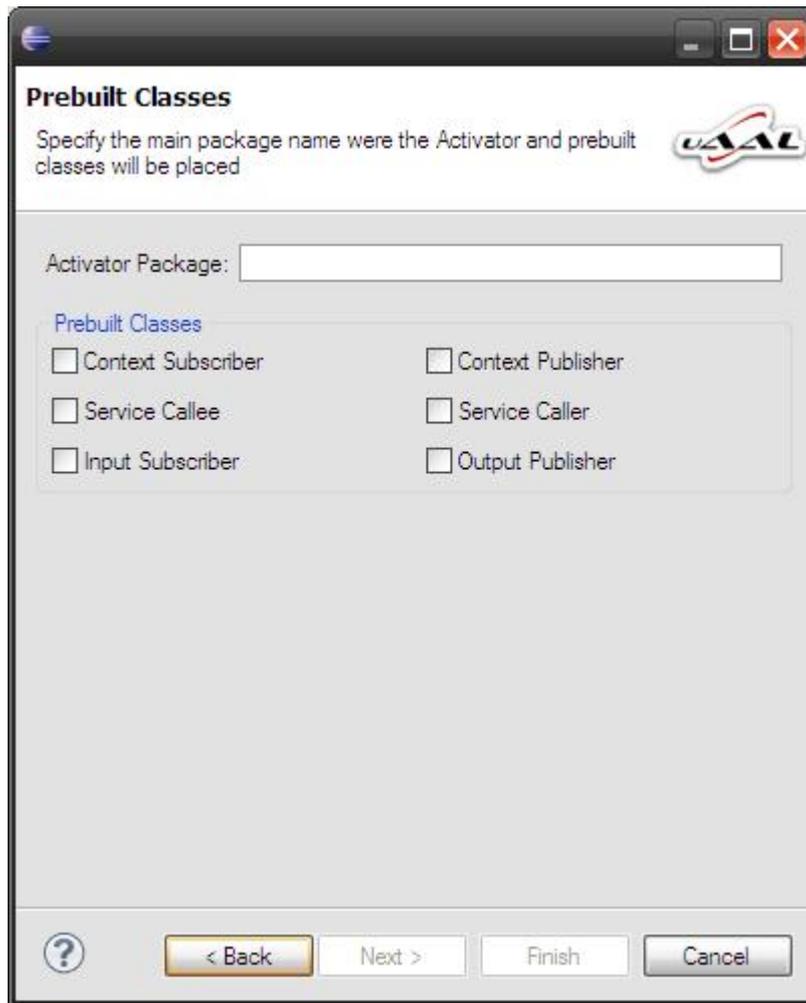
Artifact Id:

Version:

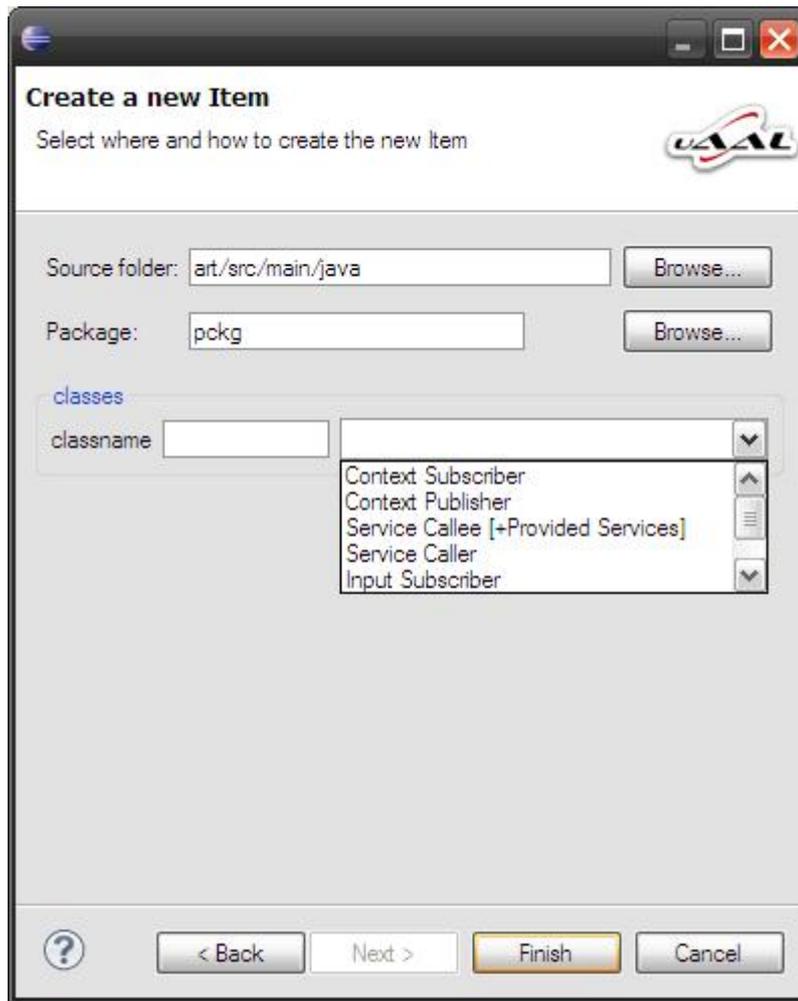
Name:

Description:

? < Back Next > Finish Cancel

Figure 4.2. Wizard:Activator package

The "New Item" wizard creates specific items on existing projects and therefore their commands could also be placed in specific contextual menus. Currently it can be accessed either from universAAL commands, the File/New menu, or by right clicking a project/folder and selecting New... The Items that can be generated in the current version are the same template files that can be generated when creating a new project.

Figure 4.3. Create a new item

Prerequisites and dependencies

No dependencies on other universAAL plugins. However it has dependencies on Maven plugin for Eclipse, which is properly notified, requested and installed during installation procedure. However if it is the plug-in source code what is being imported into Eclipse Plug-in Development, instead of the working plug-in itself, the maven plug-in will have to be manually installed to make it available for the code to compile.

Tool installation procedure

To install the tool, install it as usual with the "Install new software..." menu for plugins of Eclipse. Only required extra intallation is Maven plugin, as stated above, which is automatic. To import and compile the source code, the Maven plugin must be downloaded manually. To do so, go to "Install new software..." and introduce the Maven download site: <http://m2eclipse.sonatype.org/sites/m2e>. Then select to download the Maven Integration for Eclipse (at least version 0.9.0)

Chapter 5. Service/application project build

Role and benefits of this tool

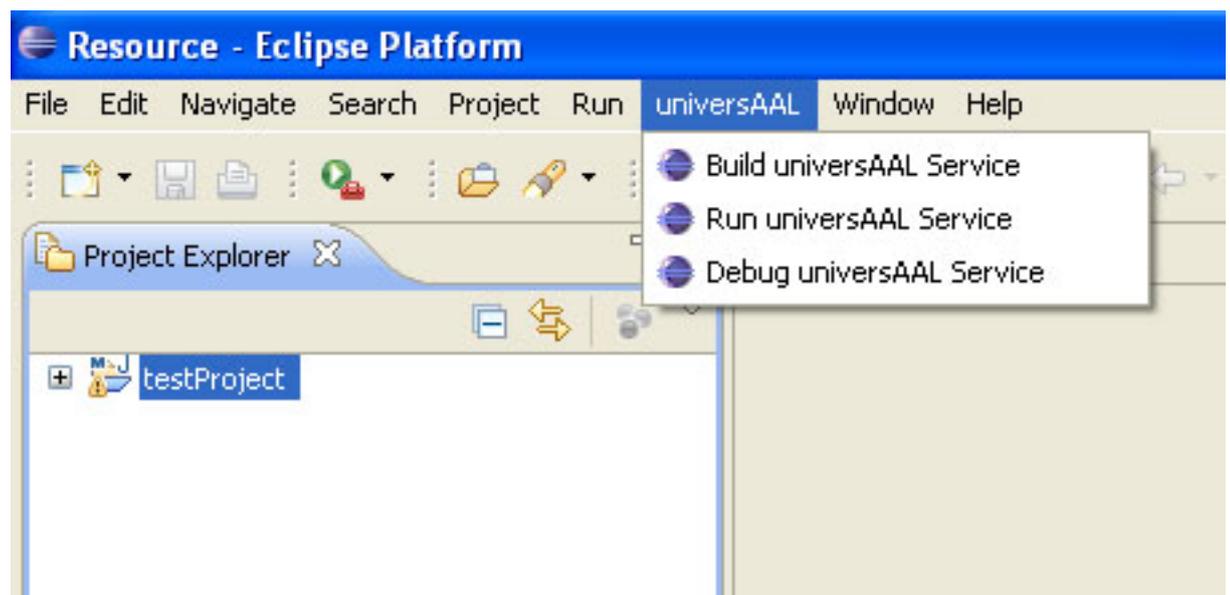
This AAL Studio tool is intended for universAAL service developers. It automates the process of building a compatible universAAL project, in order to be integrated to uStore, by utilizing Maven artifacts and Maven build mechanism. Project dependencies are resolved automatically in order to reduce incompatibility issues, since projects are build using universAAL compatible libraries. It also provides automatic uploading of implemented artifacts to local and remote repositories. This tool is highly recommended, since manual build and upload can lead to incompatible to universAAL services.

Overview of functionality

This plugin provides a safe way for building and running a universAAL service/application within Eclipse workspace. Having based on the prototype skeleton projects by the Project and Items Wizards Tool, the developer implements his application according to universAAL standards. This tool is responsible for building the whole developed Eclipse project using the appropriate libraries and dependencies for making a new universAAL compatible service/application executable. By building such projects, the resulting jar files are deployed to the local or universAAL Maven repositories, in order to be published or reused by other projects/providers. Uploading to remote repositories (uStore) is not implemented yet. Finally, this tool also has the capability to run or debug a new developed universAAL project. This is done by automatically creating an Eclipse launch configuration file that starts up all the universAAL middleware bundles that needed in order to run the new service/application.

Service/application developer should select the project that wants to build in the Eclipse workspace in order that the universAAL build, run and debug options become accessible, as in the following figure:

Figure 5.1. Build,run,debug a uAAL project



Then, the following output should appear after a successful build in the Eclipse console:

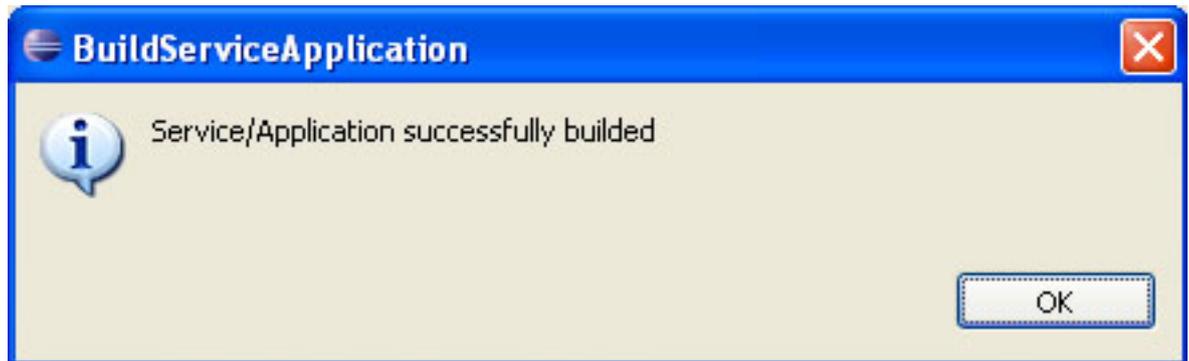
Figure 5.2. Successful build

```

[INFO]
[INFO] --- maven-bundle-plugin:2.1.0:bundle (default-bundle) @ testProject ---
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.578s
[INFO] Finished at: Tue Dec 21 15:55:36 EET 2010
[INFO] Final Memory: 22M/104M
[INFO] -----

```

along with the following pop-up window.

Figure 5.3. Successful build

Then, the created jar is uploaded automatically to the local repository along with a pom file containing all project dependencies. When running or debugging a service/application, the Pax-runner plugin will start all the necessary middleware bundles in order to run/debug the service/application. The developer can view all the started bundles by typing

Figure 5.4. Typing ps command

```

P=

```

in the Eclipse console window. The output of this command should look like:

Figure 5.5. List of installed and active bundles

```

START LEVEL 8
  ID   State      Level  Name
[  0] [Starting] | [  0] System Bundle (1.4.0)
[  1] [Installed] | [  8] Apache Felix Bundle Repository (1.4.2)
[  2] [Installed] | [  6] Unnamed - testProject:testProject:bundle:1.0 (1.0.0)
[  3] [Installed] | [  5] The PERSONA Context Ontology (0.2.0.SNAPSHOT)
[  4] [Installed] | [  4] PERSONA Turtle serializer (0.3.0.SNAPSHOT)
[  5] [Installed] | [  4] PERSONA middleware (0.3.0.SNAPSHOT)
[  6] [Installed] | [  3] OPS4J Pax ConfMan - Properties Loader (0.2.2)
[  7] [Installed] | [  3] ACL UPnP (0.2.0.SNAPSHOT)
[  8] [Installed] | [  3] OPS4J Pax Logging - Service (1.4)
[  9] [Installed] | [  3] Soda-Pop as OSGi Bundle (0.3.0.SNAPSHOT)
[ 10] [Active]   | [  2] OPS4J Pax Logging - API (1.4)
[ 11] [Active]   | [  2] wrap_mvn_jp-go.ipa.jgcl_1.0 (0)
[ 12] [Active]   | [  2] wrap_mvn_java3d_vecmath_1.3.1 (0)
[ 13] [Active]   | [  2] wrap_mvn_org.bouncycastle.jce.jdk13_144 (0)
[ 14] [Active]   | [  2] ACL Interfaces (0.3.0.SNAPSHOT)
[ 15] [Active]   | [  2] wrap_mvn_java3d_j3d-core_1.3.1 (0)
[ 16] [Active]   | [  2] Apache Felix Configuration Admin Service (1.2.8)
[ 17] [Starting] | [  2] Apache Felix UPnP Base Driver (0.8.0)
[ 18] [Resolved] | [  2] wrap_mvn_org.osgi.osgi_R4_compendium_1.0 (0)
[ 19] [Installed] | [  2] Apache Felix Log Service (0.9.0.SNAPSHOT)
[ 20] [Active]   | [  1] Apache Felix Shell Service (1.0.2)
[ 21] [Active]   | [  1] Apache Felix Shell TUI (1.0.2)

```

where the bundle with ID 2 is the bundle that has been just uploaded. In order to start running it, the developer should enter the following command:

Figure 5.6. Start bundle 6



where 6 is the default bundle level for new developed project.

Prerequisites and dependencies

This tools has been developed and tested using Eclipse 3.5.0. Moreover, the m2eclipse plugin is needed in order to run the plugin, and more specifically the 0.10.2.20100623-1649 version (update site: <http://m2eclipse.sonatype.org/sites/m2e/0.10.2.20100623-1649>) For running the middleware bundles, the Pax Runner plugin has been utilized (update site: <http://www.ops4j.org/pax/eclipse/update>)

Tool installation procedure

The installation of this plugin is performed through the "Install new Software" option of the "Help" menu of Eclipse IDE. The universAAL eclipse update site should then be inserted()in order to select the Service/Application build plugin.