Extending Appian - Custom Smart Services & Expressions

Introduction

The Appian Smart Service and Expression functionality is extensive but may not be complete for specific use cases. In such an event, you have the ability to extend this functionality by defining your own Smart Services and Expression, making use of Appian's API.

To do this, follow the guide below to install the Appian Plug-in creator and create your first custom Smart Service or Expression.

Pros

- Extend/create new functionality
- Provides flexibility

Cons

- Appian will not support your plugin
- Plug-ins require maintenance
- With increased complexity comes a higher likelihood of bugs

Installing the Appian Plug-in Creator for Eclipse

First download and install Eclipse (Tested on 4.5.1 Mars). Ensure that you install the Eclipse IDE for Java Developers.

- Click **Help > Install New Software**.
- Add a new update site using the following URL.

https://forum.appian.com/extras/developer_edition/site.xml

Creating an Appian Plug-in Project

To create a new project, open the Java perspective.

- Right click the project explorer in Eclipse.
- Click **New > Appian Plug-in Project**.
- If you do not see the `Plug-in Project` option, click **New > Other**. Select the **Plug-in Project** from the **Appian** folder.

Plug-in projects are configured simply by specifying a **Plug-in Key** and **Display Name**.

- The **Plugin-key** is used to uniquely identify your plug-in and should follow the format `com.appiancorp.mypluginname`.
- The **Display Name** can be any text string. Both are required.

The new project appears similar to the following image.

The following folders are created.

Project > src > main > java

Use this source folder to store all Java source files. The New Expression Function wizard adds Java source files to this folder.

Project > src > main > resource

Use this source folder to store all properties (internationalization) files. The New Expression Function wizard adds properties files to this folder.

Project > lib

Use this source folder to store all JAR files that your classes need in order to compile and run. Any JAR file included in this folder is automatically included in the plug-in bundle.

Project > lib-compile

This folder contains JAR files that are used to compile the plug-in bundle, but are not packaged with it. Typically JAR files that are included with

the base product (and therefore accessible on the default classpath) are stored here. The JAR files that ship with the Appian Plug-In Creator are from Appian 6.6.1. They should be replaced with JARS files from the Appian version for which the plugin is being developed.

Project > appian-plugin.xml

This file defines all components (currently only expression functions and categories) that are included in the bundle. The New Expression Function wizard updates the contents of this file.

After the project is created, ensure that the **Build Automatically** checkbox is selected.

- If not, from the main menu, click **Project > Build Automatically**.

Creating a New Java Smart Service

- Right click the **Appian Plug-in Project** (created previously).
- Select **Appian > New Smart Service**.

- Type the name of the new smart service.
- Type the palette name.
    - Use an existing palette name, unless you create a new one.
- Type the palette category.
    - Use an existing palette category name, unless you create a new one.

- Add or remove the default node inputs and outputs for the smart service. To see custom data types, import the XSD using the `Import Appian Datatypes` wizard.

- Define the Java class that provides the runtime behaviour of this smart service.

- Provide your logic for the smart service execution.
- Populate the node outputs.

- An activity class definition in XML is no longer needed.
- The class is annotated with references to inputs and outputs.
- A new properties file and two new icons are created and referenced in the package.
    - Use the properties file to generate internationalized error messages in the smart service.
    - Use the icons as the default palette and canvas icons for these activities.
    - We recommend creating your own GIF images to help identify your smart service in the palette and on the canvas.

Creating a New Java Expression

Importing Appian Custom Data Types

Deploying Plug-ins

Plug-ins are located in *`APPIAN_INSTALL/_admin/plugins/`*. Deploy a plug-in by copying it to this directory.

Plug-in deployment supports hot deployment. Changes in the Plug-ins directory are read at the polling interval *`conf.plugins.poll-interval`* specified in the *`custom.properties`* file. If this interval is set to 0 (zero), hot deployment is disabled.

Export your project as a JAR file

1. Right-click your project and click **Export...**
2. Select the **JAR file** option as the **Export destination**.
3. On the **Resources to export** dialog, clear the **.classpath** and **.project** selections as these files are used exclusively by Eclipse.
4. Select the **_admin/plugins** folder of your installation directory for your export destination.

5. This directory is created during application server startup.

Done!

Appian API

https://forum.appian.com/suite/wiki/75/api/overview-summary.html

Common Classes, Functions & Annotations

Getting your Plug-in Approved