

How to Create a New Analysis

- 1 [Introduction](#)
- 2 [Building Analysis](#)
 - 2.1 [New Plug-in Project](#)
 - 2.2 [Add Required Plug-ins](#)
 - 2.3 [New Analysis](#)
 - 2.3.1 [Analysis Description](#)
 - 2.3.2 [Analysis Task](#)
 - 2.3.3 [Analysis Result Type](#)
 - 2.4 [Example Plug-in](#)

Introduction

This tutorial assumes that you have looked over the [Analysis Framework Developer's Guide](#) and have followed the [MAEviz development environment](#) tutorial for setting up a development environment. If not, please look at those two documents because this tutorial assumes you have setup your MAEviz development environment and are ready to create a new analysis plug-in so you can begin extending MAEviz.

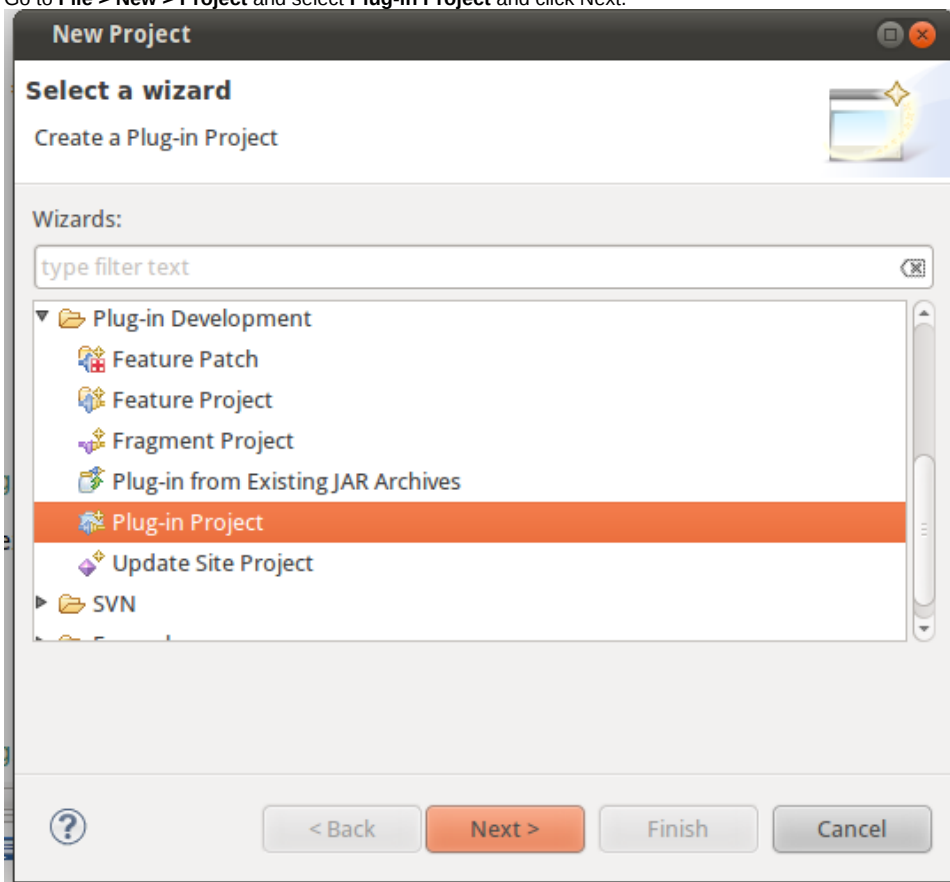
Building Analysis

The analysis we are going to perform is to check if a building is not on soft soil (0) or on soft soil (1). For simplicity, the analysis will randomly generate this number. Below you will find the steps to create this new analysis.

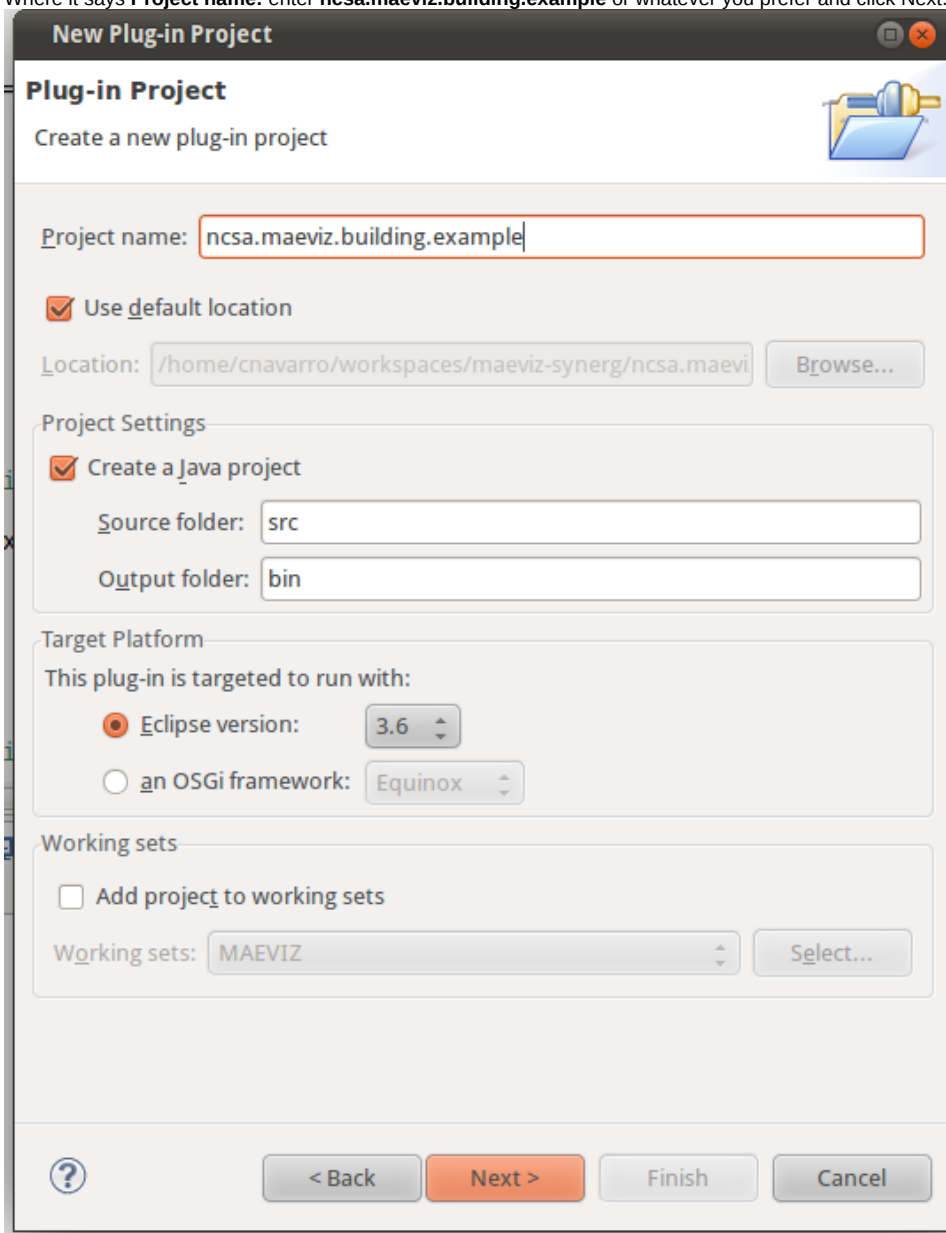
New Plug-in Project

The first step is to create a new plug-in to contain our new analysis. To do this, do the following:

1. Go to **File > New > Project** and select **Plug-in Project** and click Next.



2. Where it says **Project name:** enter **ncsa.maeviz.building.example** or whatever you prefer and click Next.



The screenshot shows the 'New Plug-in Project' dialog box in the Eclipse IDE. The dialog has a title bar 'New Plug-in Project' with standard window controls. Below the title bar is a section 'Plug-in Project' with a sub-header 'Create a new plug-in project' and a folder icon. The 'Project name:' field contains 'ncsa.maeviz.building.example'. The 'Use default location' checkbox is checked. The 'Location:' field shows the path '/home/cnavarro/workspaces/maeviz-synerg/ncsa.maevi' with a 'Browse...' button. The 'Project Settings' section has a checked 'Create a Java project' checkbox, with 'Source folder:' set to 'src' and 'Output folder:' set to 'bin'. The 'Target Platform' section indicates the plug-in is targeted to run with 'Eclipse version: 3.6' (selected with a radio button) and 'an OSGi framework: Equinox'. The 'Working sets' section has an unchecked 'Add project to working sets' checkbox, with 'Working sets:' set to 'MAEVIZ' and a 'Select...' button. At the bottom are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

New Plug-in Project

Plug-in Project
Create a new plug-in project

Project name:

☒ Use default location

Location:

Project Settings

☒ Create a Java project

Source folder:

Output folder:

Target Platform
This plug-in is targeted to run with:

☒ Eclipse version:

☐ an OSGi framework:

Working sets

☐ Add project to working sets

Working sets:

3. Nothing needs to change on this page. Make sure wizard page looks like the one below and click Finish. If the box **This plug-in will make contributions to the UI** is checked, uncheck it.

New Plug-in Project

Content

Enter the data required to generate the plug-in.

Properties

ID:

Version:

Name:

Provider:

Execution Environment:

Options

☒ **Generate an activator**, a Java class that controls the plug-in's life cycle

Activator:

☐ **This plug-in will make contributions to the UI**

☐ **Enable API Analysis**

Rich Client Application

Would you like to create a rich client application? ☐ Yes ☒ No

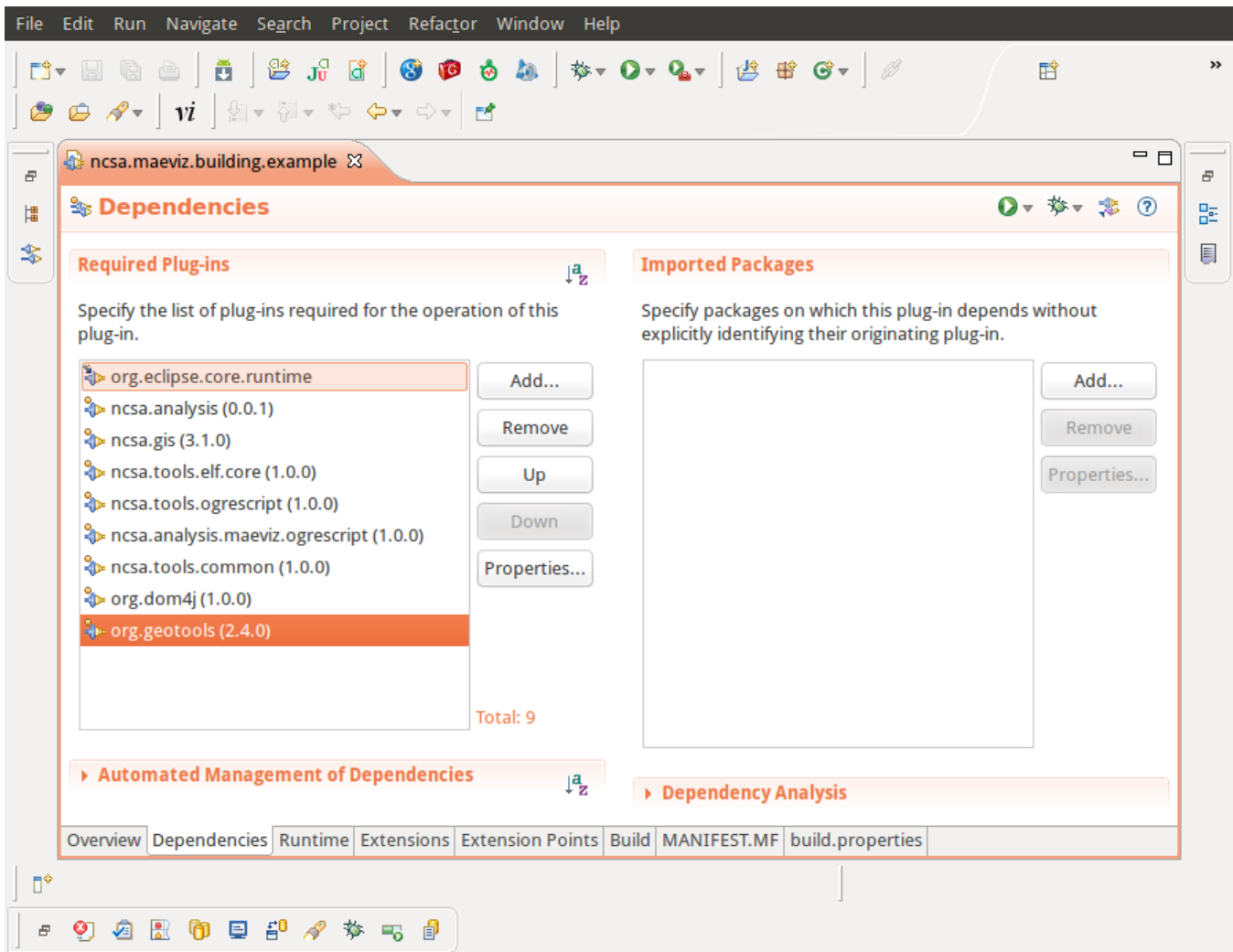
Now that we have a new plug-in, we can move on to adding the required plug-ins for our new project.

Add Required Plug-ins

We will need to add a few required plug-ins. To do this, do the following:

1. Open up the **MANIFEST.MF** file associated with our plug-in and click on the **Dependencies** tab
2. Where it says **Required Plug-ins**, click the **Add...** button and add the following plug-ins
 - ncsa.analysis
 - ncsa.gis
 - ncsa.tools.elf.core
 - ncsa.tools.ogrescript
 - ncsa.analysis.maeviz.ogrescript
 - ncsa.tools.common
 - org.dom4j
 - org.geotools

Your Dependency list should look like the one in the screenshot below:



The next steps are to add a New Analysis, a Task, and a result schema that tells the analysis how to create our result.

New Analysis

Analysis Description

To create a new analysis, open the **Manifest.MF** if it isn't already open and do the following

1. Click on the **Extensions** tab and click the **Add...** button

2. In the **New Extension** wizard that opens, search for **ncsa.analysis.newAnalyses**, select it and click Finish.

Extension Point Selection

Create a new New Analyses extension.

Extension Points | Extension Wizards

Extension Point filter:

ncsa.analysis.newAnalyses

☒ Show only extension points from the required plug-ins

Extension Point Description: [New Analyses](#)

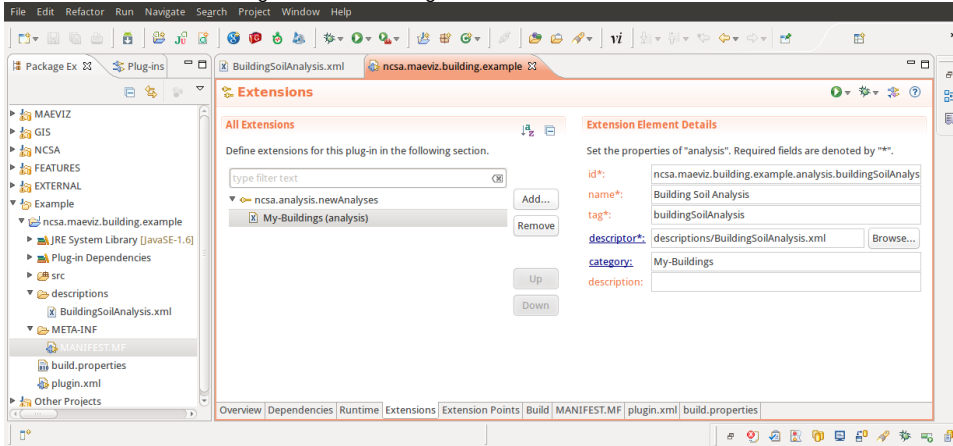
[Enter description of this extension point.]

Available templates for new analyses:

< Back Next > Finish Cancel

3. After adding the extension point, if Eclipse did not add blank new analysis element, right click on the extension point and select **New > analysis**
4. The next step is to fill in the required parts for the new analysis. Enter the following information
- **id:** `ncsa.maeviz.building.example.analysis.buildingSoilAnalysis`
 - **name:** Building Soil Analysis
 - **tag:** buildingSoilAnalysis
 - **category:** My-Buildings

5. The final requirement is a descriptor. I recommend creating a new folder inside your plug-in called descriptions and adding a new file called **BuildingSoilAnalysis.xml** that we will fill in later. After creating this file, you will need to specify it in the **descriptor** field of your new analysis. You should now have a something similar to the image below:



Your **BuildingSoilAnalysis.xml** file should contain the following xml statements:

```
<analysis-description id="nca.maeviz.building.example.analysis.buildingSoilAnalysis">
  <analysis-type type="simpleIteration">
    <property name="iteratingDatasetKey" value="buildings">
    </property>
  </analysis-type>
  <groups>
    <group-name>Required</group-name>
    <group-name>Advanced</group-name>
  </groups>

  <!-- Analysis Inputs -->
  <!-- the result name must be prefixed with the tag of the analysis, in this case buildingSoilAnalysis -->
  <parameter format="resultName" phylum="string" cardinality="single" key="buildingSoilAnalysis.resultName"
friendly-name="Result Name"/>

  <parameter phylum="dataset" cardinality="single" key="buildings" friendly-name="Buildings">
    <types>
      <type>buildingv4</type>
      <type>buildingv5</type>
    </types>
  </parameter>

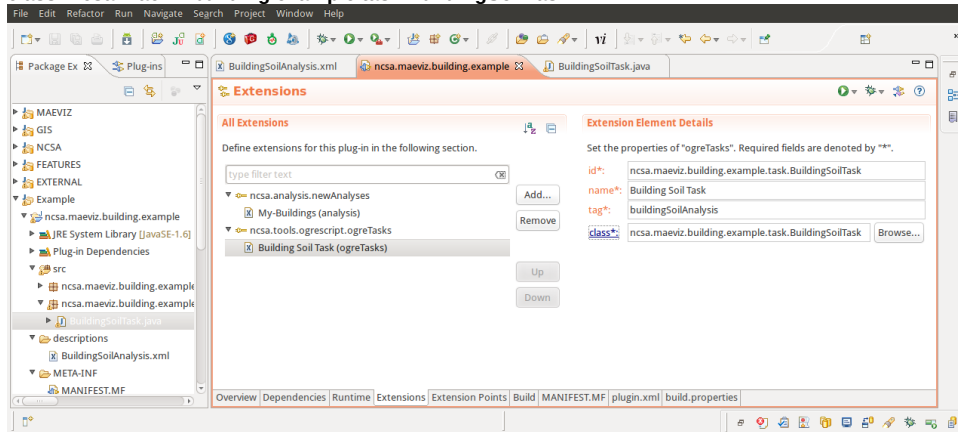
  <!-- Analysis Outputs -->
  <output friendly-name="Building Soil Results" key="buildingSoilAnalysis" phylum="dataset" format="
shapefile" geom="buildings" guids="buildings">
    <property name="buildings" type="base-dataset-key" value="buildings"/>
    <property name="schema" type="schema" value="nca.maeviz.building.example.schemas.buildingSoilResults.
v1.0"/>
  </output>
</analysis-description>
```

Analysis Task

The next step is to create a new analysis task. This is the Java code that will be executed by the new analysis to produce our building soil analysis output. To create a new task, do the following:

1. Click on the **Extensions** tab and click the **Add...** button
2. In the **New Extension** wizard that opens, search for **nca.tools.ogrescript.ogreTasks**, select it and click **Finish**.
3. After adding the extension point, if it did not add a blank task element, right click on the extension point and select **New > ogreTasks**.
4. For the new analysis task, enter the following details
 - **id:** **nca.maeviz.building.example.task.BuildingSoilTask**
 - **name:** **Building Soil Task**
 - **tag:** **buildingSoilAnalysis**

- **class: ncsa.maeviz.building.example.task.BuildingSoilTask**



It is critical that the **tag** for the Task be identical to the **tag** for the analysis since this is how MAE Viz determines which task goes with which analysis. After entering the text for the **class** field, if you click on the **class**, a source file will be generated for you. Where it says **Superclass** locate the class called SimpleFeatureTask and click Finish. You should see a wizard similar to the one below:

Java Class

Create a new Java class.



Source folder:	ncsa.maeviz.building.example/src	Browse...
Package:	ncsa.maeviz.building.example.task	Browse...
☐ Enclosing type:		Browse...

Name:	BuildingSoilTask	
Modifiers:	☒ public ☐ default ☐ private ☐ protected ☐ abstract ☐ final ☐ static	
Superclass:	ncsa.analysis.ogrescript.tasks.core.SimpleFeatureTask	Browse...
Interfaces:		Add...

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☐ Inherited abstract methods

Now that you have your class file, add the missing lines of code from the example below:

BuildingSoilTask extends SimpleFeatureTask

```
@Override
protected void handleFeature( IProgressMonitor monitor ) throws ScriptExecutionException{
    resultMap.put( "soiltype", isSoftSoil() );
}

/**
 * Generate a number between 0 and 5 (exclusive)
 * 0 - Soil Type A
 * 1 - Soil Type B
 * 2 - Soil Type C
 * 3 - Soil Type D
 * 4 - Soil Type E
 * @return Soil Type
 */
private int isSoftSoil() {
    Random rand = new Random();
    return rand.nextInt( 5 );
}
```

Analysis Result Type

The next step is to create a result type schema to specify the new fields created by our analysis. This tutorial will not go into detail about the schema file since it is behind the scope of this tutorial.

1. Click on the **Extensions** tab again and click the **Add...** button
2. In the **New Extension** wizard that opens, search for **ncsa.gis.gisSchemas**, select it and click Finish.
3. After adding the extension point, if it did not add a blank schema element, right click on the extension point and select **New > gisSchema**.
4. For the new result type, enter the following details
 - **id:** **ncsa.maeviz.building.example.schemas.buildingSoilResults.v1.0**
 - **name:** **Building Soil Results**
 - **version:** **1.0**
 - **type:** **buildingSoilAnalysis**
 - ***description:** *
 - **file:** **gisSchemas/buildingSoilResults_1.0.xsd**
 - **format:** **shapefile**
 - **requiredFields:** **soiltype**
 - **mapLayer:** **10**

For the schema file **buildingSoilResults_1.0.xsd**, please enter the following information:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:gml="http://www.opengis.net/gml" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ionicsoft.com/wfs" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:iwfs="
http://www.ionicsoft.com/wfs" targetNamespaceOptionnal="true" xmlns="http://www.ionicsoft.com/wfs"
elementFormDefault="qualified">
    <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/2.1.2
/feature.xsd"/>
    <xsd:element name="building-soil-analysis" substitutionGroup="gml:_Feature" type="iwfs:building-soil-
analysis"/>
    <xsd:complexType name="building-soil-analysis">
        <xsd:complexContent>
            <xsd:extension base="gml:AbstractFeatureType">
                <xsd:sequence>
                    <xsd:element name="maeviz.soiltype" minOccurs="0" nillable="true" type="xsd:integer"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

Example Plug-in

If you have setup your MAEviz development environment, then you should have connected to our Subversion repository. You can find this example plug-in by going to the repository at svn://subversion.ncsa.uiuc.edu/ncsa-plugins/ and doing the following:

1. Expand **trunk**
2. Find the plug-in `ncsa.maeviz.building.example`, right click on it and select **Checkout**.

This will download the plug-in to your workspace. You might need to download this to a new workspace if you already have a plug-in with that project name. Do not overwrite yours unless that is your intent.