[How to Write and Use Plug-ins for Catalog Forms](http://cawiki.ca.com/display/SolEng/How%2Bto%2BWrite%2Band%2BUse%2BPlug-ins%2Bfor%2BCatalog%2BForms)

Plug-ins are useful to populate "Select" or "Table" fields dynamically.

Inside these plugins data can be retrieved via webservice, database etc and processed before returning to the form.

To write and use a Select plug-in for use in a form, follow this process

1. Download Sample Select plug-in from Catalog.

You can find it under Tools->Administration->Plug-ins->Sample Select Plug-in.

This will be helpful to understand the structure as well as to get require JAR files.

1. Create a JAVA project and add commons-logging-api.jar and plugins.api.jar to it.

You can find these JAR files from downloaded sample Select plug-in.

1. Create a package with some meaningful name e.g. "com.ca.itcmservice.plugins.getitcmagentpackages.select".
2. Create a Class that implements \* interface FDSelectDataProvider
3. Open the class and you will find following interface methods:
4. Once you are done with above mentioned steps, export the package as JAR file.
	1. setInputs - This method is invoked by the plug-in container to provide inputs provided by the caller.

This method is used to read the input arguments sent via the Catalog form.

Input to this method is a Map that contains key and value pairs sent from caller form.

You can access the values in this way : inputs.get("propertyname")

* 1. setContext - This method provides the plug-in with services to access data provided by the container.
	2. getOptions - This method is invoked by single select, multi-select and dual list fields in a form designer form to populate their options.

In this method a LIST of FDOption is created and all the options are added to this list and returned to the caller form.

In this method we can call webservices, query database to get the options and add them to the list to be returned.

E.g. List<FDOption> options = new LinkedList<FDOption>();
        FDOption option = new FDOption("MyKey","MyValue");
        options.add(option);

1. Select an id for your plugin.
2. It should be unique and preferably meaningful e.g. "ca.catalog.itcmservice.agentpackages.select-plugin".
3. Create a folder with above id and place the JAR file in this folder.

Copy the plugin.properties from Sample select plug-in and paste in above created folder.

Copy this folder on the Catalog machine under $Catalog\_Home$/filestore/plugins/ folder

* 1. Change various attributes of property file with current plugin values. E.g. plugin\_class, id, name, description etc.
	2. Remember value of id should be same as the id of plugin.
	3. classloader.type should be set as private,
	4. if you need some JARs which are not part of default Catalog JRE else you can set classloader.type as shared.

Login to Catalog and navigate to Administration->Tools->Plug-ins and click on Reload Plugins button.

You should be able to see your newly created plugin here.

Now your plugin is ready to be used.

* 1. 

To use this plug-in in your form for the Select field, you need to provide id of the plugin in the "Report/Plug-in Id" property of the select field.

If your plug-in requires some input arguments, you can pass them via the "Report/Plug-in Variables" property. E.g. $({'dmhostname':ca\_fdGetTextFieldValue(ca\_fd.formId,'dmhost'),'dmwsurl':ca\_fdGetTextFieldValue(ca\_fd.formId,'dmwsurl')})

* 1. 

You should be able to see the Select field being populated with the options returned via the plugin!

You can use the steps mentioned above to create Table plugin.

Only difference being you need to implement FDTableDataProvider interface.

Instead of getOptions here we need to implement getTableRows method.