The Eclipse ICE Developer Menu

Overview

The Eclipse Integrated Computational Environment (ICE) has had a great track record of providing a comprehensive environment for general scientific computing. Tasks such as model input generation, local and remote simulation execution, and post-simulation data analysis and visualization are all very well supported in the application. These tasks take care of the majority of needs for *using* general scientific computing codes, but what about *developing* those applications to begin with? Is there any way ICE can be extended to provide support for the development of science codes?

É Eclipse File Edit Refactor Navigate Search Project Developer Run Window Help

Figure 1: The ICE Developer Menu

The answer is yes! In 2015 ICE was extended to provide support for scientific application *development* through a custom, extensible **Developer** top-level menu. This menu is shown in Figure 1 and provides custom actions that enable efficient scientific application development for both novice and expert users of a given science code.

The Developer Menu is completely customizable through Eclipse Extension Points. Specifically, ICE exposes a new extension point: *org.eclipse.ice.developer.code*. This extension point provides the means to specify details about a scientific code: its name, category (Framework, Nuclear, Other, etc...), and where its repository is hosted, just to name a few. This point also enables the addition of *commands* that point to some custom subclass of *org.eclipse.core.commands.AbstractHandler* that performs some task related to the development of the code.

With these extensions exposed as part of an ICE product execution, ICE handles all the complexity of picking them up and populating the Developer Menu dynamically at runtime. This feature is shown in Figure 2, with an ICE category for ICE development, and another for scientific frameworks. This process is exactly how ICE developers work on ICE itself - by pulling down the ICE binary and leveraging the Developer Menu to clone and build ICE, all within ICE. ICE also provides these hooks for other scientific codes, like the Multiphysics Object Oriented Simulation Environment (MOOSE) for general finite-element simulations. ICE provides a hook for cloning MOOSE and for forking a templated repository for MOOSE Application development.

Developer Run	W	indow Help	;t	Developer	Run Wi	ndow Help			
ICE	•	Clone ICE		ICE	•				
Framework	•	Build ICE		Framewor	k 🕨	MOOSE	•	Fork the Stork	
		ICE Tests						Clone MOOSE	
E		ICE lesis			F 🕵	。 ▽ □ □			

Figure 2: The Developer Menu ICE and MOOSE Actions

Extending the Developer Menu is easy, and relies on simply creating a new plugin and exposing a new extension. Let's see how to do this task in detail:

Extending the ICE Developer Menu

For this tutorial, we are going to create a hook into the Developer menu to clone a scientific code called Fern. Fern is an application that provides an efficient nuclear reaction network solver. It is hosted at https://github.com/jayjaybillings/fern.

To get started, we need to download the ICE plugins to our workspace, which we can actually do through the Developer menu (pretty cool to use the Developer menu to extend the Developer menu!). Click Developer > ICE > Clone ICE to get all of ICE's plugins into the workspace. With ICE cloned to the workspace, we can now begin to extend the Developer menu. First, we will need to create a new plugin.

Create a New Plugin Project

Creating a new plugin for an extension to the ICE Developer menu is simple, just click File > New > Plugin Project (or Other, then select Plugin Project). When the wizard opens, name your new plugin with something similar to Figure 3 and deselect the Generate an activation button. On the next page, simply uncheck the create a plugin from template button and click Finish.

	New Plug-in Project		• •	New Plug-in Project	
Content Enter the data required to	generate the plug-in.		Templates		
Enter the data required to Properties ID: Version: Name: Vendor: Execution Environment: Options Generate an activato Activator: com.myc This plug-in will make Enable API analysis Rich Client Application Would you like to create	eonerate the plug-in. com.mycompany.fam.developer 1.0.0 Developer MYCOMPANY JavaSE-1.8 c Environn r, a Java class that controls the plug-in's life cycle company.fam.developer.Activator e contributions to the UI a rich client application? Yes	No	Create a plug-in using one of Available Templates:	the templates	
?	< Back Next > Cancel	Finish	? < Ba	ck Next > Cancel	Finish

Figure 3: Creating a new Plugin Project

When the MANIFEST.MF file editor opens up, add the *org.eclipse.ice.developer* plugin as a Required Plugin on the Dependencies tab.

Create a New ICE Developer Extension

Now let's create a new Extension to connect ICE and the Developer Menu with a Clone Fern action. To do so, go to the Extensions tab of the plugin MANIFEST file and click add. ICE provides an org.eclipse.ice.developer.code extension point that lets users define various details about their codes.

You will then be presented with the view in Figure 4. Enter a descriptive ID and Name for this extension and click Save.

Teom.mycompany.fern.developer		° 8
🗞 Extensions		0 * * 0
All Extensions Define extensions for this plug-in in the following section. type fitter text o-org.eclipse.ice.developer.code	La Contraction Con	Extension Details Set the properties of the selected extension. Required fields are denoted by ***. ID: com.mycompany.fem Name: Fem Show extension point description Cean extension point checription Cean extension point schema Find declaring extension point

Figure 4: Configure the ID and Name of this Code Extension.

To define your scientific application, right click on the org.eclipse.ice.developer.code extension in the All Extensions section and select New > Code in the context menu. This action will present you with the view in Figure 5 where you can select the code category, code display name, the repository URL, and the branch you would like to work with.

To create a new developer action for your code, right click the code element of the extension tree and select New > Command. You will then be presented with the view at the top of Figure 5 where you can input the name of this action and the AbstractHandler subclass that performs the action. ICE provides a default GitCloneHandler that you may select. For this tutorial, select that and click Save. You should now have a view similar to the bottom of Figure 5.

Setup ICE to Run with the New Plugin

To see this new Developer command in action, we need to launch a new instance of ICE. This can be done by opening the Run Configurations Wizard in Run > Run Configurations. Under the Eclipse Applications element in the tree on the left, select the ICE launch configuration for your OS. Open the Plugins tab and add your new developer plugin (see Figure 6) by enabling it. Then, click Run to launch ICE with your developer plugin. With ICE running, navigate to the Developer menu and select Nuclear > Fern > Clone Fern (see Figure 7).

le com.mycompany.fern.developer				- 0
💝 Extensions				0 🎋 🍣 🕐
All Extensions Define extensions for this plug-in in the following section.	↓ª tt E	Extension Elements	t Details s of 'code' Required fields are denoted by '*'.	
type filter text		codeCategory*:	Nuclear	\$
▼	Add	codeName*:	Fern	
[x] (code)	Remove	repoURL:	https://github.com/jayjaybillings/fern	
		branch:	master	
	Up			
	Down			
Overview Dependencies Runtime Extensions Extension Points Bui	Id MANIFEST.MF b	build.properties plugin.>	mi	
Scom.mycompany.fern.developer				- 0
Scom.mycompany.fem.developer 🕸				日 - ① 参 参 ②
Com.mycompany.fern.developer Com.m	18. 20 🕞	Extension Eleme	nt Detailis	□ □ © \$ \$ 0
com.mycompany.fern.developer Extensions All Extensions Define extensions for this plug-in in the following section.	ļªz ≞≓ ⊫	Extension Eleme Set the propertie	nt Detailis s of 'command' Required fields are denoted by '''.	
	tªz ∰ 🖻	Extension Eleme Set the propertie commandName*	nt Details s of 'command' Required fields are denoted by '*'.	 • * * •
	ļª _Z ∰ ⊟	Extension Eleme Set the propertie commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. : Clone Fern org.aciljess.ice.developer.actions.GitCloneHandler	
	Add	Extension Eleme Set the propertie commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. : Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	
com.mycompany.fern.developer ⊠	Iª₂ S □ Add Remove	Extension Eleme Set the propertie commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. : Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	- C
 com.mycompany.fern.developer ⊠ Extensions All Extensions Define extensions for this plug-in in the following section. type filter text Y or grading exclose in the exclose is a section of the extension of the extensio	Add Remove	Extension Eleme Set the propertic commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. : Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	Browse
com.mycompany.fern.developer ⊠	I ² _Z SP ⊂ Add Remove	Extension Eleme Set the propertic commandName* implementation*	nt Details s of 'command' Required fields are denoted by '''. Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	Browse
	I ^A _Z S ^o C Add Remove	Extension Eleme Set the propertic commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	 □ □ ↓ ↓
	I ^A _Z S C Add Remove	Extension Eleme Set the properti commandName implementation*	nt Details s of 'command' Required fields are denoted by '*'. Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	 □ □ ↓ ↓
com.mycompany.fern.developer com.mycompany.fern.developer com.mycompany.fern.developer.code company.fern.fern fype filter text v org.cellpse.ice.developer.code v (code) (x) (command)	ing SI € Add Remove	Extension Eleme Set the propertie commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. : Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	□ □ □ ● * * * ? ? Browse
Com.mycompany.fern.developer 23 Call Extensions Define extensions for this plug-in in the following section. type filter text Compactipes i.co. developer.code Code) Code Command)	ing Si C	Extension Eleme Set the propertie commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. : Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	□ □ □ ● * * * ⑦
 com.mycompany.fern.developer ⊠ Extensions All Extensions Define extensions for this plug-in in the following section. type filter text T ← g.elipse.ice.developer.code T (code) T (command) 	i 2 2 2 C	Extension Eleme Set the propertie commandName* implementation*	nt Details s of 'command' Required fields are denoted by '*'. : [Clone Fern org.eclipse.ice.developer.actions.GitCloneHandler	□ □ □ ● * * * ⑦
Image: Second secon	I Add Remove	Extension Eleme Set the propertie commandName* implementation?	nt Details s of 'command' Required fields are denoted by '*'. : Clone Farn org.eclipse.ice.developer.actions.GitCloneHandler	□ □ □ ● 本 念 ⑦ Browse

Figure 5: Create a new Code description for this extension.

This will kick off the action you specified to clone the Fern repository and pull in any Eclipse projects to the workspace, as shown in Figure



Figure 6: Add your plugin to the run configuration.

t	Developer Run	Window Help		Project Explorer 🛿 🕒 🔄 🏹 🗖 🗖
*	ICE Framework Nuclear	► Fern ►	Clone Fern	 ► Colorect and the second s

Figure 7: Use the Developer Menu to Clone ICE.