FAQ for building a Cytoscape app

These are questions that are general across a variety of potential applications.

1) What is the structure of a Cytoscape app?
2) How to add an action menu item to the Tools menu
3) How to add your own Swing panel to a CytoPanel in your app

These questions are more specific to this particular purpose of adding a legend, but the functionality may be useful in other contexts as well.

4) How to get the list of visual maps used in a style
5) How to add annotations to the canvas

6) How to find more information

1) What is the structure of a Cytoscape app?

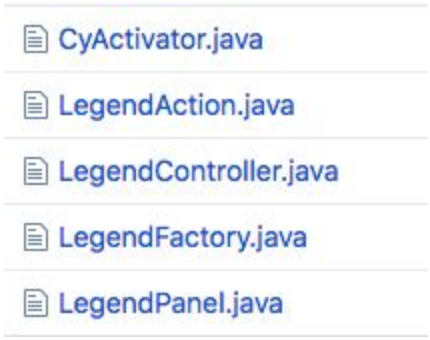
Cytoscape is designed around the OSGi framework, and the imposed constraints of that system provide the big challenge to the learning the structure of this software. It is essential to start programming in Cytoscape by editing an existing program, rather than construct the code de novo. This example may provide a good starting point.

Maven is used to build Cytoscape, specify dependencies, repositories and targets, so you will need to have a POM file for your project.

Your app will be linked into a OSGi Bundle, which is a JAR file with additional information in the manifest. You will have an activator class, conventionally called CyActivator.java, specified in the maven POM file used to build the code.

The contents of your activator class depends on what you're trying to do in your app, but there are few common patterns that can be seen from studying Legend-creator. When your bundle is registered to the system, its start method is called. The start method receives an argument called a BundleContext, which contains pointers you will need in the execution of your code, but won't be available at the time. It's important to store one or more key items from the start() method for later use.

This project contains five source files: an activator, an action, a controller, a factory and a panel.



I always create a Controller class to coordinate my model and view, but the use of the controller in the Java's Model-View-Controller implementation is open to wide interpretation. Many overload the role of the controller in tha panel or elsewhere. Here, using a concrete controller class provides a place to store the CyServiceRegistrar. We will need that reference when we create the legend from the current state of the program, in response to a button we'll put in a our panel.

The activator class extends AbstractCyActivator and provides a start() method that is called when this app is loaded.

```
public class CyActivator extends AbstractCyActivator {
	public CyActivator() { super(); }

	public void start(BundleContext bc) {
		CySwingApplication swingApp = getService(bc,CySwingApplication.class);
```

```
                CyServiceRegistrar reg = getService(bc,CyServiceRegistrar.class);

                LegendController controller = new LegendController(reg);
                LegendPanel legendPanel = new LegendPanel(controller);
                LegendAction legendAction = new LegendAction(swingApp,legendPanel);

                registerAllServices(bc,controller);
                registerService(bc,legendPanel,CytoPanelComponent.class);
                registerService(bc,legendAction,CyAction.class);
        }
}
```

The `LegendFactory` class contains all of the code to create each type of legend item. As its responsibility is to make the actual legend annotations, this class is a good example of how to add an objects to the background canvas in your next app.

The `LegendPanel` class provides the code to create a control panel in the Cytoscape frame. Here, it is a sequence of controls in a Swing container, organized via the BoxLayout. Any of the other Swing layouts (BorderLayout, GridLayout, FlowLayout, etc.) can be used here, but BoxLayout is a good choice here, as it is relatively simple, and less subject to some strange idiosyncrasies found in Swing layout managers. See this tutorial for discussion of visual layouts.

2) How to add an action menu item to the Tools menu

A) Make a class that extends AbstractCyAction.

```
public class LegendAction extends AbstractCyAction
```

B) In the constructor, set the preferred menu field.

```
public LegendAction(CySwingApplication desktop, LegendPanel myPanel){
        super("Legend Panel");
        setPreferredMenu("Tools");
```

3) How to add your own Swing panel to a CytoPanel in your app

It is easy to place your own JPanel at a variety of positions in the CytoscapeDesktop frame. The CytoPanelName interface defines navigational positions. The convention is that controls go into the WEST panel, tables in the SOUTH panel, and results in the EAST, though you can put your panels wherever is appropriate for your context.

A) Define you own class, an extension of Swing's JPanel with your labels , fields, and buttons.

```
public class LegendPanel extends JPanel implements CytoPanelComponent
```

B) Implement the CytoPanelComponent interface to specify title, location, icon, etc.

```
public CytoPanelName getCytoPanelName() { return CytoPanelName.WEST;      }
```

C) Accept the LegendController object in the constructor and cache it.

```
public LegendPanel(LegendController ctrl)
```

D) Set up action listeners for buttons and field pass any actions you catch within the panel to the controller.

```
ActionListener layout = new ActionListener() {
@Override public void actionPerformed(ActionEvent e) { controller.layout(); }
        };
adder.addActionListener(layout);
```

4) How to get the list of visual maps used in a style

A) Get the current visual style assigned to the network.

In LegendController:

```
VisualMappingManager manager =
       (VisualMappingManager) registrar.getService( VisualMappingManager.class);
VisualStyle style = manager.getCurrentVisualStyle();
```

B) Get the mapping functions from the Cytoscape API. A LegendCandidate is a wrapper around the mapping function to facilitate filtering and sorting of the functions.

```
class LegendCandidate implements Comparable<LegendCandidate>

private List<LegendCandidate> candidates  = new ArrayList<LegendCandidate>();


private void findLegendCandidates(VisualStyle style)
{
       Collection<VisualMappingFunction<?, ?>> vizmapFns =
                     style.getAllVisualMappingFunctions();
       for (VisualMappingFunction<?,?> fn : vizmapFns)
       {
              String mappingType = fn.toString();
              if (mappingType.contains("Passthrough")) continue;
              candidates.add(new LegendCandidate(fn));
       }
       Collections.sort(candidates);
}
```

5) How to add annotations to the canvas

A) Use the registrar to get the AnnotationManager in the controller's initialize() method

```
annotationMgr = registrar.getService(AnnotationManager.class);
```

B) Build your own factory class to create your annotation objects

```
factory = new LegendFactory(registrar, networkView);
```

C) You'll set all the attributes (which canvas, z order, formatting) in the factory, and simply add the annotation to the manager without arguments.

```
annotationMgr.addAnnotation(textBox);
```

6) How to find more information

How to browse the Java Docs

I seem to always have trouble finding the javadoc to version 3. Is there a URL more generic than:

http://chianti.ucsd.edu/cytoscape-3.6.1/API/

Make sure you exclude Cytoscape v.2 results from searches, as they seem to come up despite being over 5 years out of date.