

# Extending SmartTranslator

## Extending Smart Translator via Property Editors

I've been working with Hivemind for a couple of months and I'm very happy with it, so I would like to contribute my 2 cents. I came across a situation where I was using a service that has getters/setters that were of type Class and every time it would exception out on trying to get/set the values from these methods. So I took it upon myself in open source fashion and dug into the code. I found where the exception was coming from and I knew a way to fix it in the source code, but was this the most correct way? NO!

---

I'd rather add startup code to register a proper Property Editor with the Java Beans framework. There's an extension point for this purpose (hivemind.Startup). Once the Property Editor is defined, it should "just work".

– Howard Lewis Ship\_

---

Back to the drawing board. With the hint from Howard Lewis Ship, the light clicked on and I was back in business. I knew the Smart Translator uses Property Editors to work with types, but only primitive types in java are supported without having to create an editor. I needed one for type "java.lang.Class", so I created one.

## Property Editor (ClassEditor.java)

```
{{{public class ClassEditor extends PropertyEditorSupport
{
    public void setAsText( String text ) throws java.lang.IllegalArgumentException
    {
        try
        {
            Thread thread = Thread.currentThread();
            ClassLoader loader = thread.getContextClassLoader();
            setValue( loader.loadClass(text) );
        }
        catch( ClassNotFoundException cnfe )
        {
            throw new java.lang.IllegalArgumentException(cnfe.getMessage());
        }
    }

    public Object getValue(){
        return super.getValue();
    }

    public String getAsText(){
        return getValue().toString();
    }
} }}}
```

Since this editor needed to be registered with the PropertyEditorManager, I created a service to handle the task.

## PropertyEditorLoader Service

### Service Interface (PropertyEditorLoader.java)

```
public interface PropertyEditorLoader
{
    void setPropertyEditors( List editors );
    void register();
}
```

### Service Implementation (PropertyEditorLoaderImpl.java)

```

public class PropertyEditorLoaderImpl implements PropertyEditorLoader
{
    private List _editors;
    private ClassResolver _resolver;

    public void register()
    {
        for( int i=0; i < _editors.size(); i++ )
        {
            PropertyEditorParameters editors = (PropertyEditorParameters) _editors.get(i);

            Class targetType = _resolver.findClass( editors.getTargetType() );
            Class editorClass = _resolver.findClass( editors.getEditorClass() );

            PropertyEditorManager.registerEditor( targetType, editorClass );
        }
    }

    public void setPropertyEditors(List editors)
    {
        _editors = editors;
    }

    public void setClassResolver( ClassResolver resolver )
    {
        _resolver = resolver;
    }
}

```

## Supporting Class (PropertyEditorParameters.java)

```

public class PropertyEditorParameters
{
    private String _targetType;
    private String _editorClass;

    public String getTargetType()
    {
        return _targetType;
    }

    public void setTargetType(String targetType)
    {
        _targetType = targetType;
    }

    public String getEditorClass()
    {
        return _editorClass;
    }

    public void setEditorClass(String editorClass)
    {
        _editorClass = editorClass;
    }
}

```

## Wiring Hivemind

hivemodule.sdl

```

module (id=my.stuff version="1.0.0")
{
  ...
  configuration-point ( id=PropertyEditors )
  {
    schema
    {
      element (name=editor)
      {
        attribute (name=target-type required=true)
        {
          "The class name of a custom data type"
        }

        attribute (name=editor-class required=true)
        {
          "The class name of and editor for a custom data type"
        }

        conversion (class=propertyeditor.PropertyEditorParameters)
        {
          map (attribute=target-type property=targetType)
          map (attribute=editor-class property=editorClass)
        }
      }
    }
  }

  service-point ( id=PropertyEditorLoader interface=propertyeditor.PropertyEditorLoader)
  {
    "Register an editor class to be used to edit values of a given target class"

    invoke-factory( service-id=hivemind.BuilderFactory )
    {
      construct ( class=propertyeditor.PropertyEditorLoaderImpl initialize-method=register
                  class-resolver=property=classResolver )
      {
        set-configuration( property=propertyEditors configuration-id=PropertyEditors )
      }
    }
  }

  contribution (configuration-id=PropertyEditors)
  {
    editor ( target-type=java.lang.Class editor-class=propertyeditor.ClassEditor)
  }

  contribution (configuration-id=hivemind.EagerLoad)
  {
    load ( service-id=PropertyEditorLoader )
  }
  ...
}

```

Using a configuration point for the PropertyEditors gives me the capability of implementing any number of PropertyEditors I want.

## Conclusion

The Property Editor is created and registered with hivemind and the Smart Translator is now aware of type "java.lang.Class" and my problems are solved. *"It just works"*

---

[HowardLewisShip](#): This is very good, however I have one criticism. The PropertyEditorLoader service should implement java.lang Runnable and be contributed into the hivemind.Startup configuration. Even as coded here, the methods setPropertyEditors() and register() can be part of the service implementation *but not the service interface*, because it doesn't make sense for other code to call these methods.

---

[KurtHoehn](#): Thank You, I added an alternative configuration below.

---

## Alternative `PropertyEditorLoader` Service

### `PropertyEditorLoader` Implementation (`PropertyEditorLoaderImpl.java`)

```
public class PropertyEditorLoaderImpl implements Runnable
{
    private List _editors;
    private ClassResolver _resolver;

    public void run()
    {
        for( int i=0; i < _editors.size(); i++ )
        {
            PropertyEditorParameters editors = (PropertyEditorParameters) _editors.get(i);

            Class targetType = _resolver.findClass( editors.getTargetType() );
            Class editorClass = _resolver.findClass( editors.getEditorClass() );

            PropertyEditorManager.registerEditor( targetType, editorClass );
        }
    }

    public void setPropertyEditors(List editors)
    {
        _editors = editors;
    }

    public void setClassResolver( ClassResolver resolver )
    {
        _resolver = resolver;
    }
}
```

### `hivemodule.sdl`

```
...
service-point ( id=PropertyEditorLoader interface=java.lang.Runnable)
{
    "Register an editor class to be used to edit values of a given target class"

    invoke-factory( service-id=hivemind.BuilderFactory )
    {
        construct ( class=propertyeditor.PropertyEditorLoaderImpl class-resolver-property=classResolver )
        {
            set-configuration( property=propertyEditors configuration-id=PropertyEditors )
        }
    }
}
...
contribution (configuration-id=hivemind.Startup)
{
    service ( service-id=PropertyEditorLoader )
}
...
```