"M-VC" Project Proposal
XML language for rich SWT/JFace UIs

along with data-binding, validation, and other goodies

Eclipse Summit Europe 2007

11 October 2007
Presentation outline

- Introduction to M-VC
- Of M-VC Forms
- Everything is an endpoint
- Enriching your domain classes
- Think of the RCP programmer
- From our standpoint
How it started

90% of the code in large-scale projects is User Interface-related

SWT/JFace make sure this saying stays true

We devised a way to work around that without losing too much (!)
How it started (2)

SWT and JFace are great considering:

• SWT is powerful, fast, consistent with underlying platform, friendly to Win32 programmers, etc.

• JFace adds a lightweight yet useful stack for model-based programming
How it started (3)

SWT/JFace programming has a drawback:

rich interfaces require too many LOCs (lines of code)

Out of programmers’ laziness was born M-VC
What it has become

M-VC has been rewritten 3 times until today. It is now

- A complete **functional** language (M-VC Forms) with back-bindings to Java and forward-bindings to Java6 scripting
- A rich model metadata infrastructure that closely maps the model to the UI
- A set of services, tools and helpers for the RCP developer
Of M-VC Forms
Imagine a button in SWT

```java
helloButton = new Button(parentComposite, SWT.NONE);
helloButton.setText("Say Hello");
helloButton.setLayoutData(new GridData(GridData.FILL_BOTH));
helloButton.addSelectionListener(new SelectionAdapter()
{
    @Override
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Hello World!");
    }
});
```
In M-VC this would be

```xml
<element type="button" label="Say Hello">
  <observe endpoint="click">
    <call endpoint="::systemOut"
      parameter="Hello World!/">
    </call>
  </observe>
</element>
```
Another example

The following send-email form...

```xml
<layout numOfColumns="2" grabHorizontal="true">
  <element type="text" id="from" label="From" widthHint="200" />
  <element type="text" id="to" label="To" widthHint="200" />
  <element type="text" id="server" label="SMTP Server" widthHint="200" />
  <element type="text" id="subject" label="Subject" grabHorizontal="true" />
  <element type="text" variant="memo" id="text" label="Message"
      grabHorizontal="true" heightHint="M" />
  <element type="button" label="Send" horizontalSpan="2"
      horizontalAlignment="right">
    <observe endpoint="click">
      <call class="com.ergobyte.test.mvc.SendEmailEndpoint" />
    </observe>
  </element>
</layout>
```
Another example: result
Mapping the UI to the model

Provided the following domain class

```java
public class LoginCredentials {
    private String username;
    private String password;

    @Label("Your login name")
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    @Label("Your login password")
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```
Mapping the UI to the model (2)

You can connect it with the UI simply as that

\[
\begin{align*}
\text{<element type="text" connect=":instance.$username" />}
\text{<element type="text" connect=":instance.$password" />}
\end{align*}
\]

You will get the label for free too.

For a surprise, try to put the first element twice in your form.
Features of M-VC forms

- Themable, actually 2 themes (SWT and JFace Forms)
- Translatable (*.properties files)
- All layouts available
- Dynamic - can change while form is open
- Global actions support
- Hot-replace: modify a form, save it and reopen it from within the target program
- Can be used everywhere SWT is acceptable. Right now in Views, Editors, PreferencePages, Dialogs, Wizards
- Easy tables and trees, with cell editing
- Easy drag and drop, with automatic POJO to native DnD object and reverse transformations
Everything is an endpoint
Endpoints?

Within M-VC, controls are called Elements. An Element is a special form of Namespace. Every Namespace contains any number of Endpoints.

So…

M-VC elements contain endpoints. Ex "text" element has "value", "readOnly", "alignment" endpoints.
Endpoints ???

An endpoints is something a namespace provides you **by name** (ex: "readOnly")

IEndpoint interface is blank!
But normally endpoints implement one or more of the derived interfaces:

- readable, writable, callable, observable
- and others (quite a few actually)
Endpoints - finally an example

When it comes the "readOnly" endpoint of the "text" element, you could say:

```xml
<read endpoint="readOnly" /> <!-- get the value -->
<write endpoint="readOnly" value="true" /> <!-- set the value -->
<observe endpoint="readOnly">
    <!-- do something when the value changes -->
</observe>
<call endpoint="readOnly">
    <!-- trigger the value-change logic -->
</call>
<connect endpoint1="form.$password.disable" endpoint2="readOnly" />
```
Endpoints are everywhere

the label of a form,
the color of a button,
the layout in which controls are,
the library functions of M-VC,
the services (described later),
the model meta-data, etc
Enriching your domain classes
M-VC needs your help

M-VC goes a long way to make things happen auto-magically
Labels, formatters, tooltips, validation, lookups, icons, I18N, etc are all made for you

But you have to annotate your domain to get maximum benefit
Domain classes additions

Annotations: @MvcType, @Property, @Label, @Validator, ...

Interfaces: Labeled, Editable, Contextual, Parametric, Validatable, TreeElement, ...

When used correctly you diminish even more the effort to code the UI
Think of the RCP programmer
Bindings to the RC Platform

A view in <1min: FormViewPart
An multi-page editor in <1min: FormEditorPart
A wizard in <1min: FormWizard
A preference page in <1min: FormPreferencePage
A dialog in <1min: FormDialog

You get the gist. Try also

&view id="test" name="A View"
   class="com.ergobyte.mvc.forms.FormViewFactory:test:testForm.xml" /

Global action handlers

M-VC handles common global actions like Copy, Paste, Select All, Refresh etc

The "actions" endpoint is available to all M-VC elements, allowing them to enable/disable them.

Then a simple <observe endpoint="copy">... is needed
Services

Services are classes that get initialized on-demand and provide global facilities to all M-VC projects.

IFormManager, IModelManager, IServiceManager, IActionManager, IObjectRegistry, IPreferenceManager

are all provided by default by M-VC.
Object registry service

Keeps an efficient hash-map of named objects

MvcPlugin.getObjectRegistry().getPartition(Dog.class).get("snoopy");
Preference manager service

Stores and retrieves preferences (key-value pairs).

Caches values, uses various backends, actually JFace and Database but more can be implemented.
From our standpoint
We use M-VC everyday

On top of M-VC we created the "Unified Business Application Framework", which combines

**M-VC + Hibernate + BIRT + RCP**

to create a platform where any kind of business application can be developed in a fraction of the time

Our clients love it. Our developers too!

We believe M-VC is mature enough to become a successful community project.
Eclipse project proposal

We want to open-source M-VC under the EPL.

Ideally, a number of persons will express their interest into M-VC, and help prepare the proposal

If accepted, Ergobyte Informatics will contribute the initial code, plus commiters to the project
Thank you for your attention

Georgios Nikolaidis – gnikolaidis@ergobyte.gr