

Declarative User Interfaces

presented by Herman Lintvelt

(c) 2010 Polymorph Systems

Agenda

- UI DSLs
- Code Generation
- Declarative UIs

Poll

Who of you are not frustrated with the way you are doing GUI development?

UI DSL

- It is possible to define a DSL for the UI application domain
- Examples:
 - Eclipse XWT (XML)
 - Griffin (Groovy)
 - Swing JavaBuilder (YAML)
 - Glimmer (JRuby)

Let's have a look at the code

```
public class RestaurantAppWindow extends ApplicationWindow {
    private Text text;

    public RestaurantAppWindow() {
        super(null);
    }

    private void doFindByName(String name){
        //TODO this should be delegated to controller...
        System.out.println("Finding Restaurants matching name: "+name);
    }

    protected void configureShell(Shell newShell) {
        super.configureShell(newShell);
        newShell.setText("JRestaurant - your friendly Restaurant Finder");
    }

    protected Point getInitialSize() {
        return new Point(600, 300);
    }

    public static void main(String args[]) {
        try {
            RestaurantAppWindow window = new RestaurantAppWindow();
            window.setBlockOnOpen(true);
            window.open();
            Display.getCurrent().dispose();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    //continue on next slide...
```

```
protected Control createContents(Composite parent) {
    Composite container = new Composite(parent, SWT.NONE);
    container.setLayout(new GridLayout(1, false));

    SashForm sashForm = new SashForm(container, SWT.NONE);
    sashForm.setLayoutData(new GridData(SWT.FILL, SWT.FILL, true, true, 1, 1));

    Composite searchComposite = new Composite(sashForm, SWT.NONE);
    searchComposite.setLayout(new GridLayout(3, false));

    Label searchNameLabel = new Label(searchComposite, SWT.NONE);
    searchNameLabel.setLayoutData(new GridData(SWT.RIGHT, SWT.CENTER, false, false, 1, 1));
    searchNameLabel.setText("Restaurant Name");

    Text text = new Text(searchComposite, SWT.BORDER);
    text.setLayoutData(new GridData(SWT.FILL, SWT.CENTER, true, false, 1, 1));

    Button btnFind = new Button(searchComposite, SWT.NONE);
    btnFind.addSelectionListener(new SelectionAdapter() {
        public void widgetSelected(SelectionEvent e) {
            doFindByName(text.getText());
        }
    });
    btnFind.setText("Find");

    Composite mapComposite = new Composite(sashForm, SWT.NONE);
    mapComposite.setLayout(new GridLayout(1, false));

    Label lblPlaceholderForAwesome = new Label(mapComposite, SWT.NONE);
    lblPlaceholderForAwesome.setText("Placeholder for awesome map interface");
    sashForm.setWeights(new int[] {1, 1});

    return container;
}
```

```
package jrestaurant
```

Griffon

```
application(title: 'JRestaurant',
  size: [600,300],
  locationByPlatform:true) {

  panel(border:emptyBorder(6)) {
    BorderLayout()

    splitPane(dividerLocation:300,
      leftComponent:
        panel(){
          BorderLayout()
          vbox(constraints:WEST){
            label("Restaurant Name:")
            hstrut(5)
            textField(text:bind(target:model, targetProperty:'restaurantName'))
            hstrut(5)
            button("Find", actionPerformed:controller.&doSearchByName)
          }
        },
      rightComponent:
        panel(){
          label("Placeholder for awesome map view")
        }
    })
  }
}
```

```
<Composite xmlns="http://www.eclipse.org/xwt/presentation"
  xmlns:x="http://www.eclipse.org/xwt"
  xmlns:c="clr-namespace:jrestaurant"
  xmlns:j="clr-namespace:java.lang"
  x:Class="jrestaurant.SearchView">
  <Composite.layout>
    <GridLayout numColumns="3" />
  </Composite.layout>
  <Label text="Restaurant Name" x:style="SWT.NONE"></Label>
  <Text text="" x:style="SWT.NONE"></Text>
  <Button text="Find" x:style="SWT.PUSH" selectionEvent="doSearchByName"></Button>
</Composite>
```

```
public class SearchView extends Composite {
  protected SearchModel model = new SearchModel();

  public SearchView(Composite parent, int style) {
    super(parent, style);
  }

  public void doSearchByName(Object object, Event event) {
    System.out.println("Find restaurant by name: "
      + model.getRestaurantName());
  }
}
```

JFrame(name=myFrame,title=JRestaurant Your Friendly Restaurant Finder):

- JLabel(name=searchLabel, text=Restaurant Name)
- JTextField(name=nameField)
- JButton(name=findButton, text=Find, onAction=doSearchByName)

bind:

- nameField.text: model.restaurantName

Applying a UI DSL

- A theoretical UI DSL is not of much use
- It can be applied in two ways:
 - Code Generators
 - Declarative UIs

UI DSL Considerations

- What do we gain by introducing a UI DSL?
- Would it be feasible to adapt a part of your development work to use a UI DSL?
- Need to understand why UI DSL helps before looking at Code Generators and Declarative UIs (or using existing ones)

Code Generators

- UI DSL is used to generate UI code, which then gets compiled and executed
- OR: domain objects are used to create skeleton code for GUI
 - e.g. Grails, Spring-Roo for web
- Desktop Examples:
 - Griffon:
 - Like Grails, but for Swing apps
 - “Convention over Configuration”
- Xtext & Xpand: code generation for custom DSLs

Declarative UIs

- Use UI DSL to build up a (runtime) model of UI that gets translated to UI objects
- Examples:
 - Eclipse e4 (XWT)
 - Groovy SwingBuilder
 - Swing JavaBuilder

Eclipse e4

A service-oriented programming model, based on OSGi, that provides better isolation of software components from their surrounding environment.

- 1 The GUI is represented as a uniform model that can be generically queried, manipulated, tooled, and extended, allowing for rapid design and customization of the user interface with little or no coding effort.**
- 2 Use of web styling technology (CSS), allows the presentation of user interface elements to be infinitely tweaked and reconfigured without any modification of application code.**
- 3 Bringing Eclipse runtime technology into the JavaScript world, and enabling software written in JavaScript to be executed in the Eclipse runtime.**
- 4 A framework for defining the design and structure of Standard Widget Toolkit (SWT) applications declaratively. This eliminates writing of repetitive boilerplate SWT code, thus reducing development cost, improving UI consistency, and enabling customized application rendering.**
- 5 A new port of SWT, dubbed "browser edition", that allows existing SWT applications to be executed on web platforms such as ActionScript/Flash.**
- 6 In the development tools space, a more flexible resource model that provides better support for complex project layouts.

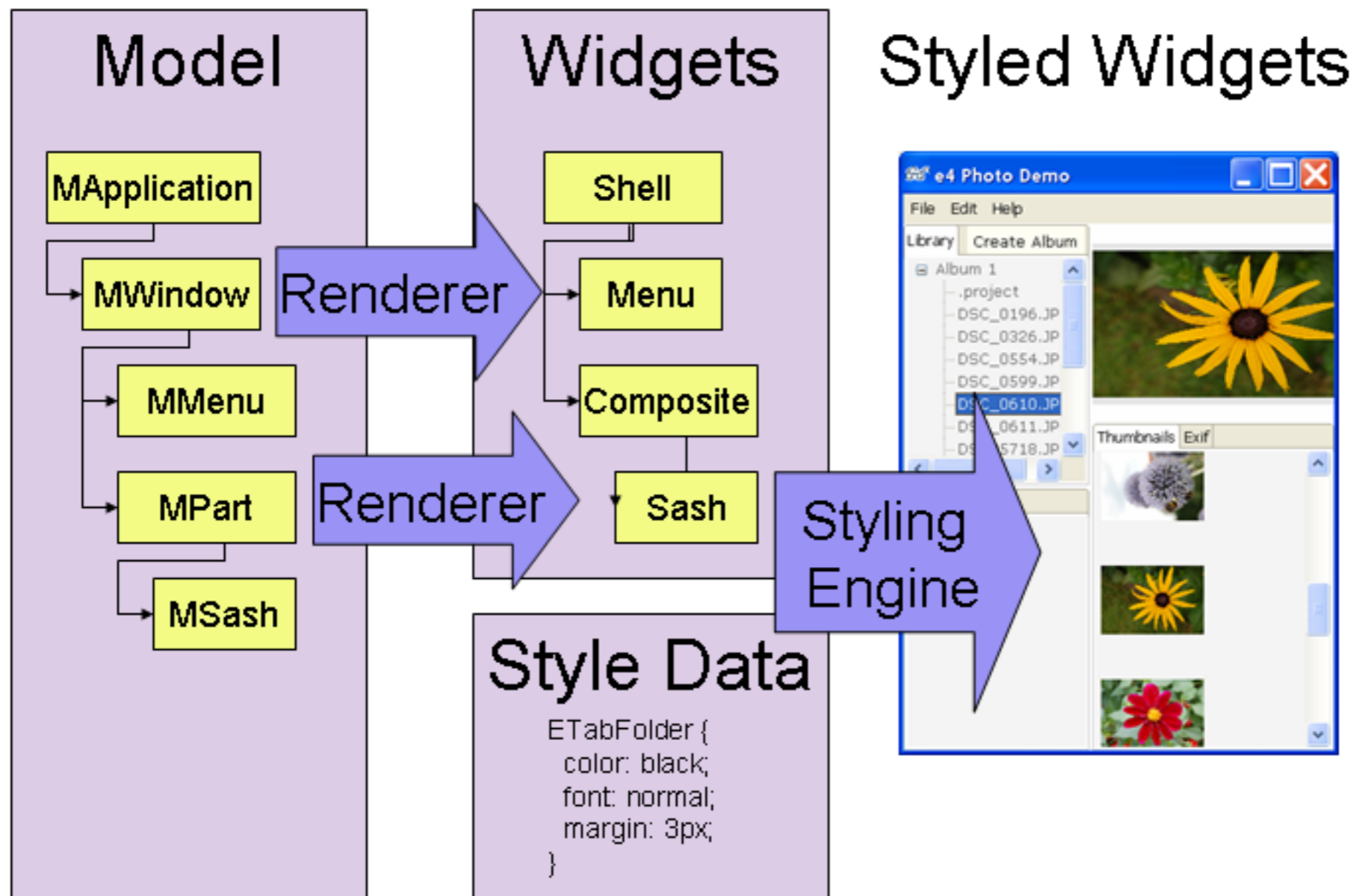


Figure 4 – Render and style dataflow

I. Uniform Model

The GUI is represented as a

- **uniform model** that can be generically
 - queried,
 - manipulated,
 - tooled,
 - and extended,
- allowing for rapid design and customization of the user interface with **little** or no **coding effort**.

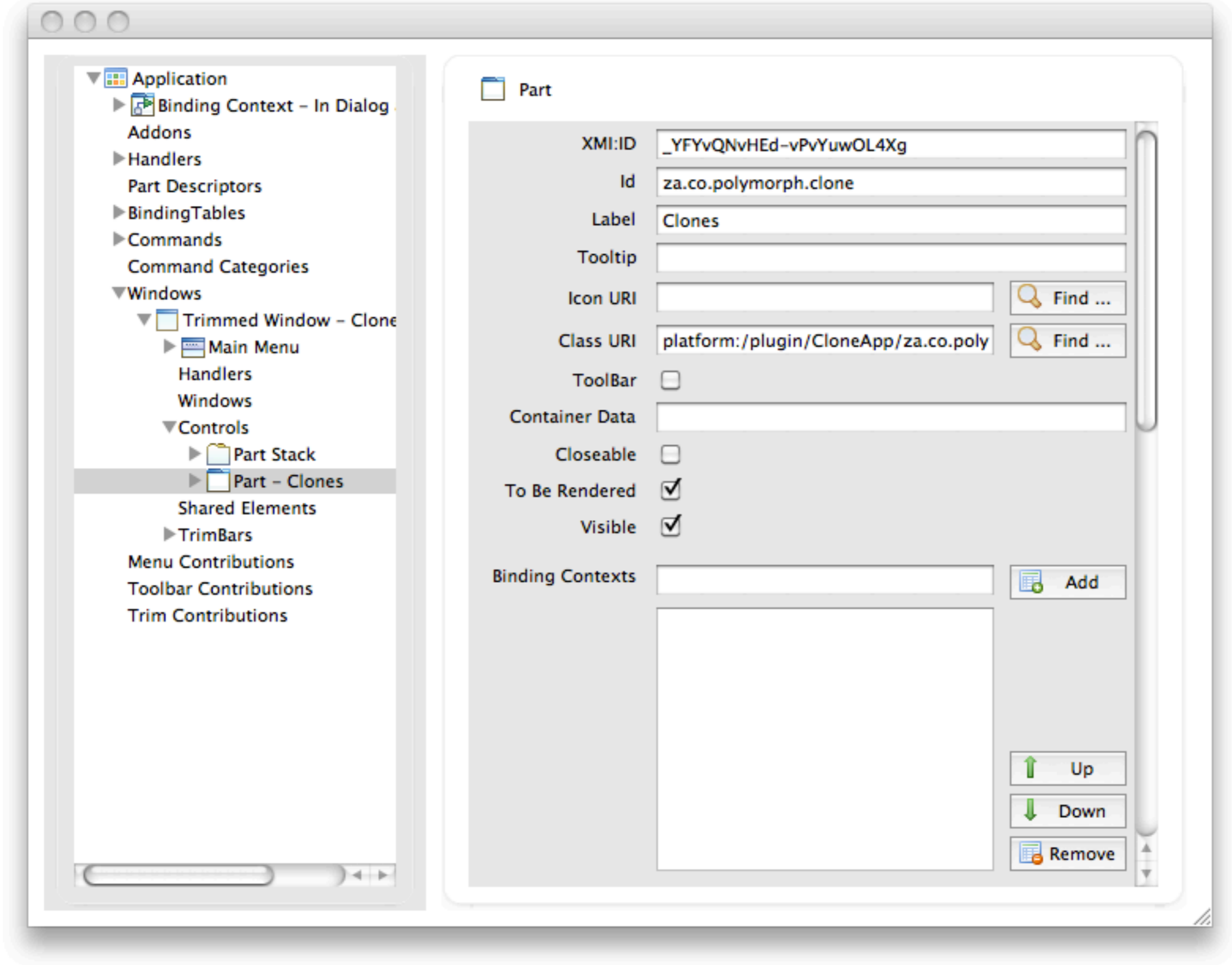
- ▼ Application
 - ▼ Binding Context – In Dialog and Windows
 - Binding Context – In Windows
 - Binding Context – In Dialogs
 - Addons
 - ▶ Handlers
 - ▼ Part Descriptors
 - ▶ PartDescriptor – Clones View
 - ▶ BindingTables
 - ▼ Commands
 - Command – quitCommand
 - Command – openCommand
 - Command – saveCommand
 - Command – aboutCommand
 - Command Categories
 - ▼ Windows
 - ▼ Trimmed Window – CloneApp
 - ▼ Main Menu
 - ▼ Children
 - ▼ Menu – File
 - Children
 - ▶ HandledMenuItem – Open
 - ▼ HandledMenuItem – Save
 - Parameters
 - ▶ HandledMenuItem – Quit
 - ▶ Menu – Help
 - Handlers
 - Windows
 - ▼ Controls
 - ▼ Perspective Stack
 - ▶ Perspective
 - Shared Elements
 - ▶ TrimBars
 - Menu Contributions
 - Toolbar Contributions
 - Trim Contributions



Clone Name Receive Salary
Share Bank Account?

```
<Composite xmlns="http://www.eclipse.org/xwt/presentation"
  xmlns:x="http://www.eclipse.org/xwt"
  xmlns:c="clr-namespace:za.co.polymorph.cloneapp.domain.ui"
  xmlns:j="clr-namespace:java.lang"
  x:Class="za.co.polymorph.cloneapp.domain.ui.ClonePresentationView">
  <Composite.layout>
    <GridLayout numColumns="4" />
  </Composite.layout>
  <Label text="Clone Name"/>
  <Text x:Style="Border" text="{Binding path=name}">
    <Text.layoutData>
      <GridData grabExcessHorizontalSpace="true"
        horizontalAlignment="GridData.FILL" widthHint="100"/>
    </Text.layoutData>
  </Text>
  <Label text="Receive Salary"/>
  <Button x:Style="SWT.CHECK" text="{Binding path=receiveSalary}">
    <Button.layoutData>
      <GridData grabExcessHorizontalSpace="true"
        horizontalAlignment="GridData.FILL" widthHint="100"/>
    </Button.layoutData>
  </Button>
  <Label text="Share Bank Account?">
  <Button x:Style="SWT.CHECK" text="{Binding path=shareBankAccount}">
    <Button.layoutData>
      <GridData grabExcessHorizontalSpace="true"
        horizontalAlignment="GridData.FILL" widthHint="100"/>
    </Button.layoutData>
  </Button>
  <Label></Label></Composite>
```

Node	Content
▼ Composite	
ⓐ xmlns	http://www.eclipse.org/xwt/presentation
ⓐ xmlns:x	http://www.eclipse.org/xwt
ⓐ xmlns:c	clr-namespace:za.co.polymorph.cloneapp.domain.ui
ⓐ xmlns:j	clr-namespace:java.lang
ⓐ x:Class	za.co.polymorph.cloneapp.domain.ui.ClonePresentationView
▼ Composite.layout	
▼ GridLayout	
ⓐ numColumns	4
▼ Label	
ⓐ text	Clone Name
▼ Text	
ⓐ x:Style	Border
ⓐ text	{Binding path=name}
▶ Text.layoutData	
▶ Label	
▼ Button	
ⓐ x:Style	SWT.CHECK
ⓐ text	{Binding path=receiveSalary}
▼ Button.layoutData	
▶ GridData	
▶ Button.	
📄	layoutData>
▼ Label	
ⓐ text	Share Bank Account?
▶ Button	
▶ Label	



2. Styling

Use of web **styling** technology (CSS),

- allows the **presentation** of user interface elements
- to be infinitely tweaked and reconfigured
- **without** any **modification** of application **code**.

File Theme

Contacts List

First Name	Last Name
Chris	Aniszczyk
Boris	Bokowski
Peter	Friese
Kevin	McGuire
Tom	Schindl
Hallvard	Traetteberg
Kai	Tödter
Yves	YANG

Details

General

Full name: Kevin McGuire

Company: IBM Canada

Job Title: Eclipse UI Guy

Note:

Business Address

Street: 2670 Queensview Drive

City: Ottawa

Zip code: K2B 8K1

Province: Ontario

Country: Canada

Phones:

Internet: kevin_mcguire@ca.ibm.com

Web Page:

CSS

```
#SeparatorLabel {  
    color: #f08d00;  
}
```

3. Other languages

Bringing Eclipse runtime technology into the JavaScript world,

- and enabling software written in JavaScript
- to be executed in the Eclipse runtime.
- i.e. UI model can be extended using other languages, DSLs

4. Declarative Design and Structure

A framework for defining the

- design
- and structure
- of Standard Widget Toolkit (SWT) applications declaratively.
- This eliminates
 - writing of repetitive boilerplate SWT code,
 - thus reducing development cost,
 - improving UI consistency,
 - and enabling customized application rendering.

5. Multiple Targets

A new port of SWT, dubbed "browser edition",

- that allows **existing** SWT applications
- to be executed on web **platforms** such as
ActionScript/Flash.
- i.e. same UI declaration, multiple target platforms

E4 Contact Demo - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://127.0.0.1:10080/org.eclipse.e4.demo.contacts.product?startup=org.eclipse.e4.ui.workbench.swt.applicatic

Main

File Theme

Contacts List

First Name	Last Name
Chris	Aniszczyk
Boris	Bokowski
Peter	Friese
Kevin	McGuire
Tom	Schindl
Hallvard	Traetteberg
Kai	Tödter
Yves	YAIG

Details


General

Full Name: Peter Friese

Company: itemis

Job Title:

Note:



Business Address

Street: Schauenburgerstraße 116

City: Kiel

ZIP: 24118

State/Prov:

Country: Germany

Business Phones

Phone: +49 431 5606-338

Mobile:

Business Internet

Email: peter.friese@itemis.de

Web Page:

Done

The Promised Land?



- **Fast-tracking Java GUI development**
 - 2 day workshop
 - UI DSL, XText, Principles and Practices
- **Various Java and OO courses**
- **Visit jtraining.co.za**





PÖLYMÖRPHsystems
USER • INTERFACE • EXCELLENCE

www.polymorph.co.za



PÖLYMÖRPHsystems
MOBILE • FUN



PÖLYMÖRPHsystems
TRAINING • EXPERTS