Tom Schindl

Tom.schindl@bestsolution.at
http://www.bestsolution.at
http://tomsondev.bestsolution.at

Innsbruck, Austria

UFaceKit - Uniform UI Development

Eclipse Summit Europe
Wednesday, October 27, 2009

UFaceKit – About Me

- **Founder and Owner of BestSolution.at**
- **Eclipse Committer**
  - Platform UI
  - e4
  - EMF
- **Projectlead**
  - Nebula
  - UFaceKit

UFaceKit – Mission

- **Promote Eclipse Databinding to outside Eclipse Communities**
  - Making it available on other platforms
  - Providing implementations for Swing, Qt, GWT
- **Provide Toolkit Neutral Widget-API**
  - Using Declartive Approaches
  - Using Reflective API similar to EObject from EMF

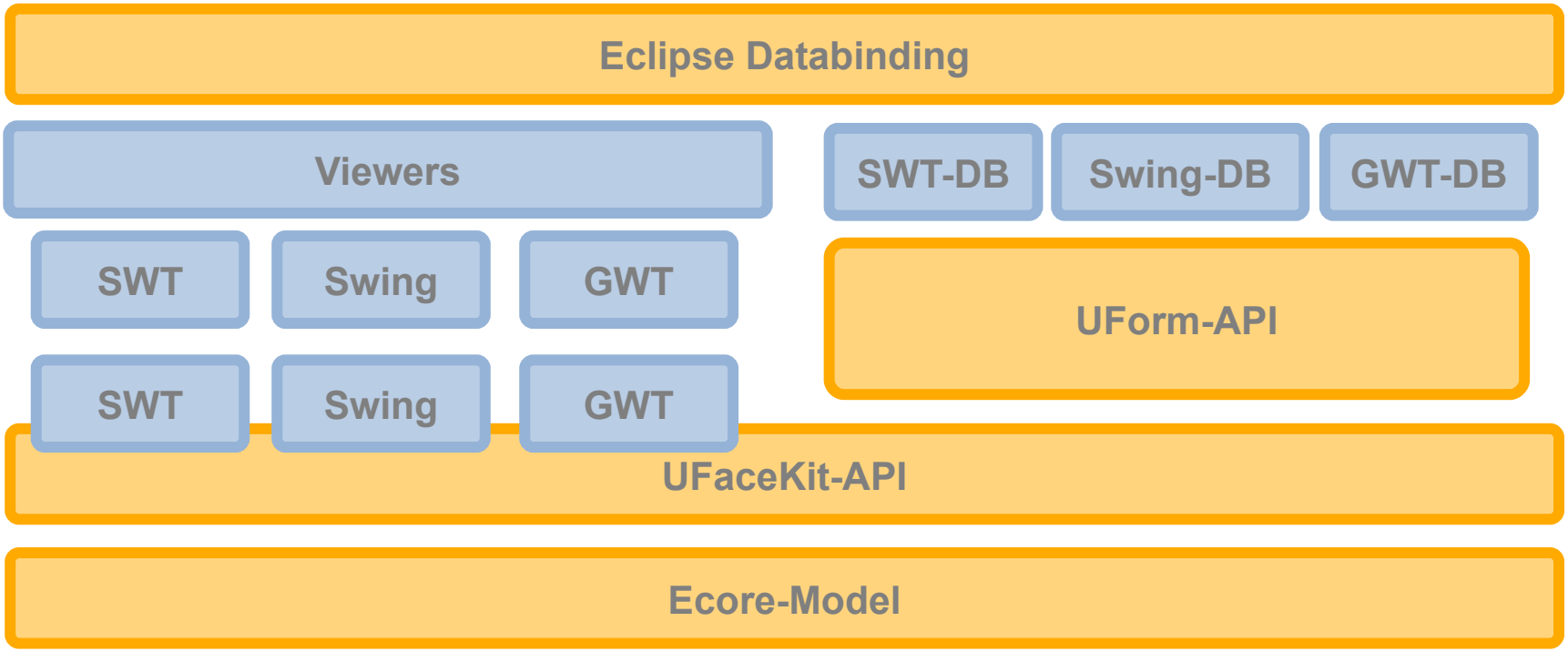
UFaceKit – Motivation

- **Switching between technologies requires to learn different APIs**
  - E.g. Swing-MVC vs. JFace
- **Write Form UI once and make it look native on different platforms**
  - E.g. Configuration Dialog in Netbeans and Eclipse

UFaceKit – Project structure

- **Make all parts consumeable on its own**
  - Inside / outside OSGi
  - Use Swing Databinding without buying in the abstraction API provided

UFaceKit – Project structure



UFaceKit – API Design

- **Factory Pattern**
- **Reflective Widget API**
  - Attributes accessable **without** reflection
    e.g. text.set(ITextProperty.TEXT,“Hello“)
- **Using of interfaces**
  - e.g. ITextPropertyOwner, ...
- **Usage of Java Language Features**
  - JFace-Viewers with typesafety

UFaceKit – Example code

SWT-UI

```
JFaceFactory factory = new JFaceFactory();
window = factory.newApplicationWindow(
  desktop,
  new UIApplicationWindow.ApplicationWindowUIInfo(factory.newFillLayout()));

UIComposite mainComposite = rootFactory.newComposite(
  window,
  new UIComposite.CompositeUIInfo(null,rootFactory.newGridLayout(3,false)));
```

Qt-UI

```
QTFactory factory = new QTFactory();
window = factory.newApplicationWindow(
  desktop,
  new UIApplicationWindow.ApplicationWindowUIInfo(factory.newFillLayout()));

UIComposite mainComposite = rootFactory.newComposite(
  window,
  new UIComposite.CompositeUIInfo(null,rootFactory.newGridLayout(3,false)));
```

UFaceKit – Declarative Styling

- **UFaceKit enforces declarative styling**
  - No setBackground-method
  - No setFont-method
  - …
- **Advantage of UfaceKit-Styling**
  - Take advantage of the platform support (e.g. Qt has direct CSS-Support)

UFaceKit – Declarative Styling

- **Styling DSL is plugable**
  - CSS
  - USML (a very very simple XML Description of styles)

```
<?xml version="1.0" encoding="UTF-8"?>
<styles>
  <style type="element" element-name="UILabel">
    <attribute name="color" value="#646464"/>
  </style>
  <style type="class" class-name="white_sep">
    <attribute name="background-color" value="#646464"/>
    <attribute name="margin-top" value='20px'/>
  </style>
</styles>
```

UFaceKit – Extra Features

- **Declarative Style**
  - style classes
  - style ids
  - style by element
  - Specials
    - Pseudo-classes like Focus, Hover
    - CSS-Selector like

UFaceKit – Extra Features

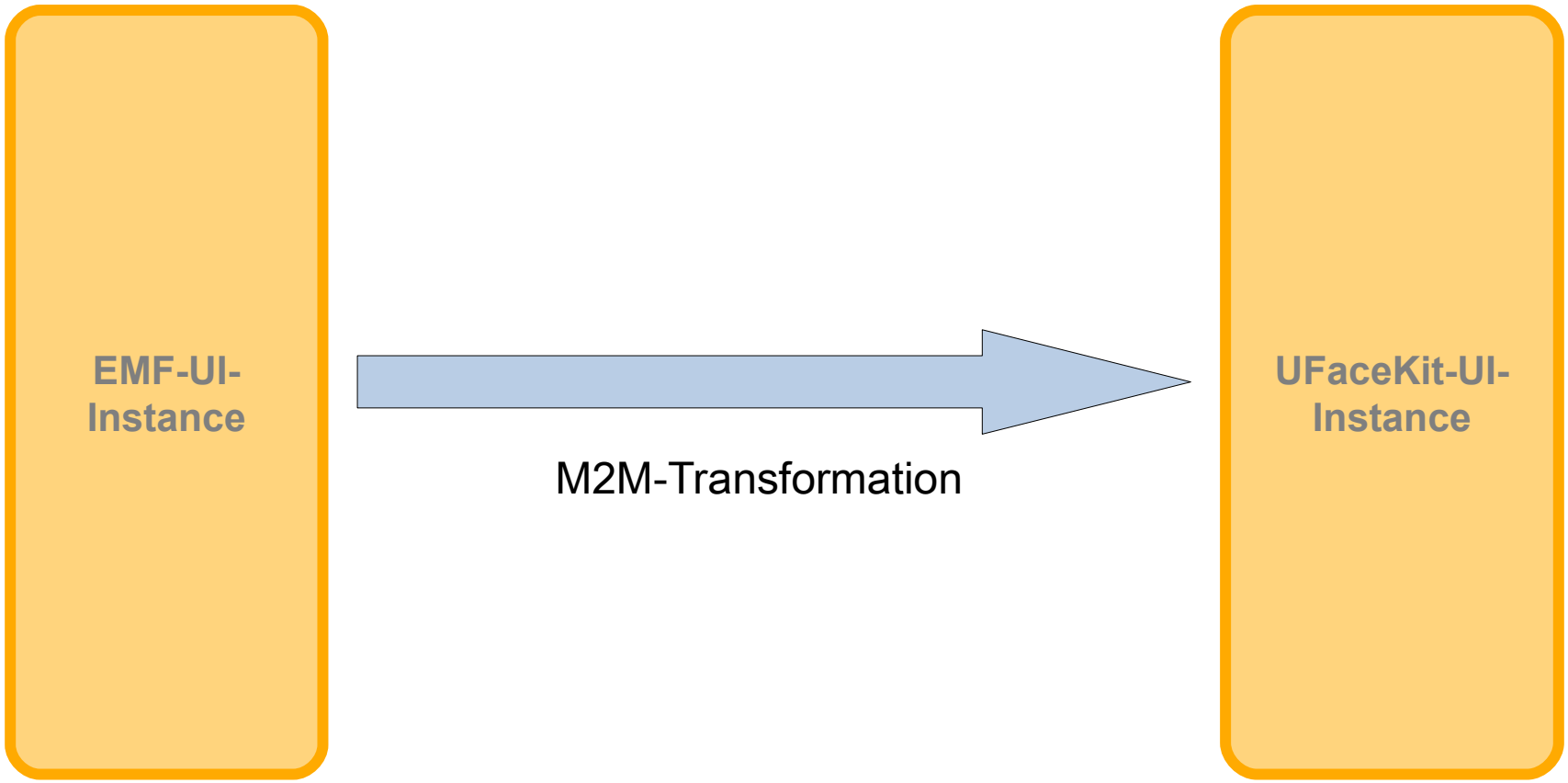
- Xpath Queries to Walk the widget-tree

```
XPathContextFactory<UIWidget> factory =
        UFacekitXPathContextFactory.newInstance();

XPathContext context = factory.newContext(containerComposite);
Iterator<?> iterator = context.iterate(„/label“);

for (Iterator<?> it = iterator; it.hasNext();) {
  UIControl control = (UIControl)it.next();
  control.getStyle().setBackgroundColor("red");
}
```

- JSON-Serialisation (e.g. useable for Testing)

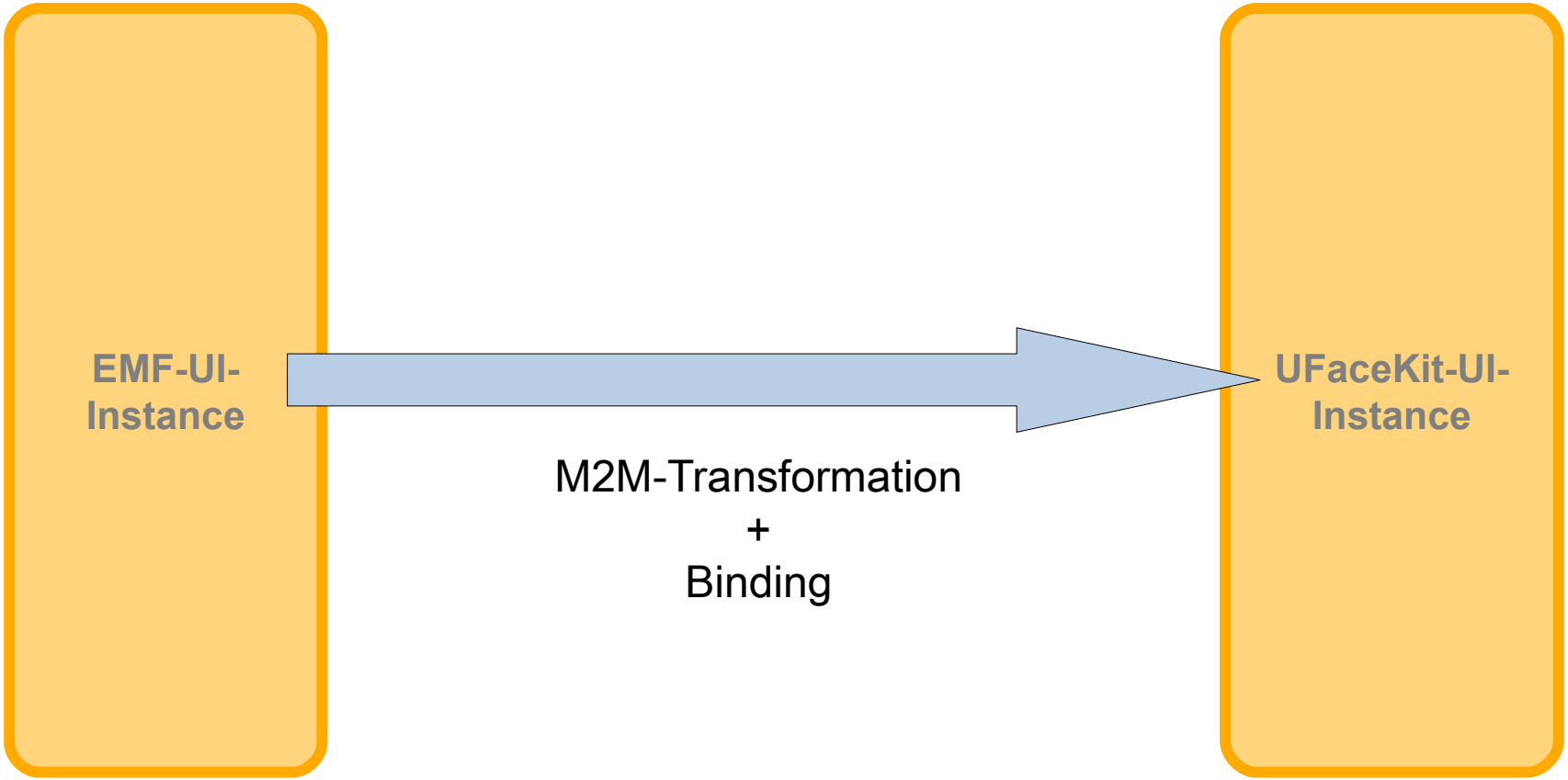
UFaceKit – Usage of EMF

- **Load UI From Model**



UFaceKit – Usage of EMF

- Backup the UI with EMF



UFaceKit – Web

- **Support for**
  - Plain GWT (Viewers, Observables)
  - SmartClient (Viewers, Observables)
- **Own Wrapper based upon Qooxdoo**

UFaceKit - Resources

- Open Source
  - www.eclipse.org/ufacekit
  - www.ufacekit.org
- Commerical Support
  - www.bestsolution.at

THE END