



e(fx)clipse - JavaFX Tooling and Runtime

Tom Schindl - BestSolution Systemhaus GmbH

Cluj June 2012

(c) Tom Schindl - BestSolution Systemhaus GmbH

About Tom

- ❖ CEO BestSolution Systemhaus GmbH
- ❖ Eclipse Committer
 - ❖ e4
 - ❖ Platform UI
 - ❖ EMF
- ❖ Main developer of e(fx)clipse



What is JavaFX

- ❖ modern Graphics Toolkit for Java
 - ❖ uses OpenGL and DirectX (when possible)
- ❖ Is almost cross-platform (Win & Mac GA, Linux is beta)
- ❖ Will get part of the JDK 8
- ❖ Successor of Swing

What technologies are needed?

- ❖ Minimum

- ❖ Java or any other VM-Language (Scala, Groovy, ...)

- ❖ Optimal

- ❖ Java or any other VM-Language
 - ❖ CSS
 - ❖ XML (FXML)

Why do we need e(fx)clipse

For Java we have JDT

```
package bsp;

import javafx.application.Application;
import javafx.stage.Stage;

public class Example extends Application {

    @Override
    public void start(Stage primaryStage) {
        HBox root = FXMLLoader.load(getClass().getResource("Example.fxml"));
        Scene s = new Scene(root);
        s.getStylesheets().add(getClass().getResource("example.css").toExternalForm());
        primaryStage.setScene(s);
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

What does e(fx)clipse provide

- ❖ Enhancements for Java Tooling
 - ❖ Wizards to create JavaFX projects (e.g. a specialized Classpath-Container)
 - ❖ Packageing support for JavaFX applications

DEMO TIME JDT- Project

Why do we need e(fx)clipse

For CSS we have WST

```
#root {  
  -fx-background-color:  
    linear-gradient(#686868 0%, #232723 25%, #373837 75%, #757575 100%),  
    linear-gradient(#020b02, #3a3a3a),  
    linear-gradient(#b9b9b9 0%, #c2c2c2 20%, #afafaf 80%, #c8c8c8 100%),  
    linear-gradient(#f5f5f5 0%, #dbdbdb 50%, #cacaca 51%, #d7d7d7 100%);  
}  
  
#button {  
  -fx-padding: 10;  
  -fx-background-color: red;  
}
```

What does e(fx)clipse provide

- ✧ CSS

- ✧ Provides auto-completion for JavaFX css properties
- ✧ Provides validation for JavaFX css properties
- ✧ Supports CSS 3 Selectors (which JavaFX itself doesn't fully)
- ✧ CSS-Editor based upon xtext
- ✧ Can be extended to teach it css new properties

Why do we need e(fx)clipse

For XML we have WST

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import java.lang.*?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.layout.HBox?>
```

```
<HBox xmlns:fx="http://javafx.com/fxml">
```

```
  <children>
```

```
    <Button text="Hello World"/>
```

```
  </children>
```

```
</HBox>
```

What does e(fx)clipse provide

- ✧ FXML
 - ✧ Is a serialization of an object graph (similar to XAML)
 - ✧ Has **NO** fixed Schema or DTD
- ✧ FXML-Editor is a customized WST-XML-Editor
 - ✧ provides auto-completion for attributes
 - ✧ provides live preview of result
- ✧ SVG / FXG to FXML Converter

What does e(fx)clipse provide

- ✧ fxgraph
 - ✧ JSON like object graph definition
 - ✧ reduces the noise created by XML in FXML
 - ✧ „compiles“ to FXML
 - ✧ no runtime libraries needed
 - ✧ interoperability with other tools like Scenebuilder, Netbeans, ...
- ✧ **No Expressions**

Basic Structure

```
package bsp
```

```
import javafx.scene.layout.HBox
```

```
import javafx.scene.control.Button
```

```
import bsp.ExampleController
```

```
component Example controlledby ExampleController styledwith "example.css" {
```

```
    HBox {
```

```
        children : [
```

```
            Button {
```

```
                text : "Hello World"
```

```
            }
```

```
        ]
```

```
    }
```

```
}
```

Demo Time

DEMO TIME FXML & FXGraph Tooling

RUNTIME

e(fx)clipse runtime

❖ Integration in Eclipse Databinding

```
TextField f = new TextField();  
IEMFValueProperty mProp = EMFProperties.value(MEDIA__TITLE);  
IJFXBeanValueProperty tProp = JFXBeanProperties.value("text");  
dbc.bindValue(  
    tProp.observe(f),  
    mProp.observeDetail(currentSelection)  
);
```

❖ Adapting to FX-Observable

```
ObservableValue<Double> fxValue =  
AdapterFactory.adapt(EMFProperties.value(PHOTO__AREA__X).observe(eObject))  
  
ObservableList<PhotoArea> fxList =  
AdapterFactory.adapt(EMFProperties.list(PHOTO__AREAS).observe(eObject));
```

e(fx)clipse runtime

- ✧ Adapting to FX-Observable is very nice e.g. for Tables

```
photoAreas = new TableView<PhotoArea>();
```

```
TableColumn<PhotoArea, String> col = new TableColumn<PhotoArea, String>();
```

```
col.setText("Description");
```

```
col.setCellValueFactory(new Callback<TableColumn.CellDataFeatures<PhotoArea, String>,  
ObservableValue<String>>() {
```

```
    @Override
```

```
    public ObservableValue<String> call(CellDataFeatures<PhotoArea, String> param) {
```

```
        IObservableValue v = EMFProperties.value(PHOTO_AREA_DESCRIPTION).observe(param.getValue());
```

```
        return AdapterFactory.adapt(v);
```

```
    }
```

```
});
```

```
ObservableList<PhotoArea> list =
```

```
AdapterFactory.adapt(EMFProperties.list(PHOTO__AREAS).observeDetail(currentSelection));
```

```
photoAreas.setItems(list);
```

e(fx)clipse runtime

- ✧ Integration into Equinox-OSGi

e(fx)clipse runtime

- ❖ Integration into Equinox-OSGi

```
public class Example extends Application {  
  
    @Override  
    public void start(Stage primaryStage) {  
        // TODO Implement  
    }  
}
```

e(fx)clipse runtime

- ❖ Integration into Equinox-OSGi

```
public class Example extends Application {  
  
    @Override  
    public void start(Stage primaryStage) {  
        // TODO Implement  
    }  
}
```

```
public class Example extends AbstractJFXApplication {  
  
    @Override  
    protected void jfxStart(IApplicationContext context,  
        javafx.application.Application jfxApplication,  
        Stage primaryStage) {  
        // TODO Implement  
    }  
}
```

e(fx)clipse runtime

- ✧ How does the integration work
 - ✧ JavaFX does not directly support OSGi
 - ✧ JavaFX 2.1 improved by providing the possibility to set classloaders e.g. when loading stuff using FXML
- ✧ Current integration uses „Equinox Classloader Hooks“
 - ✧ Locate javafx.jar on the filesystem
 - ✧ Bundle javafx.jar inside a bundle

DEMO TIME

OSGi-App

Using JavaFX for Eclipse 3.x views

- ✧ JavaFX can be embedded into SWT and Swing Containers
 - ✧ for SWT integration widget is FXCanvas

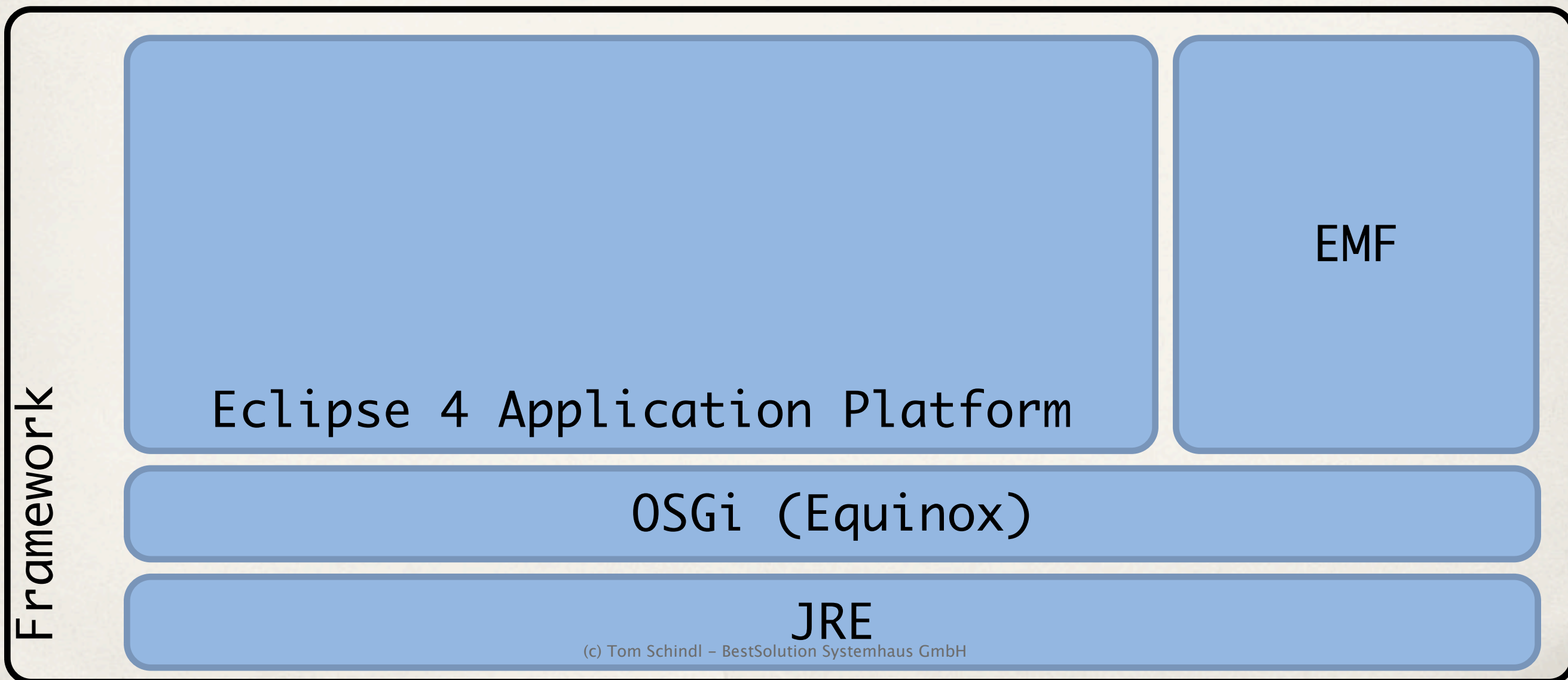
```
FXCanvas canvas = new FXCanvas(parent, SWT.NONE);  
Button b = new Button("Hello SWT");  
canvas.setScene(new Scene(b));
```

DEMO TIME

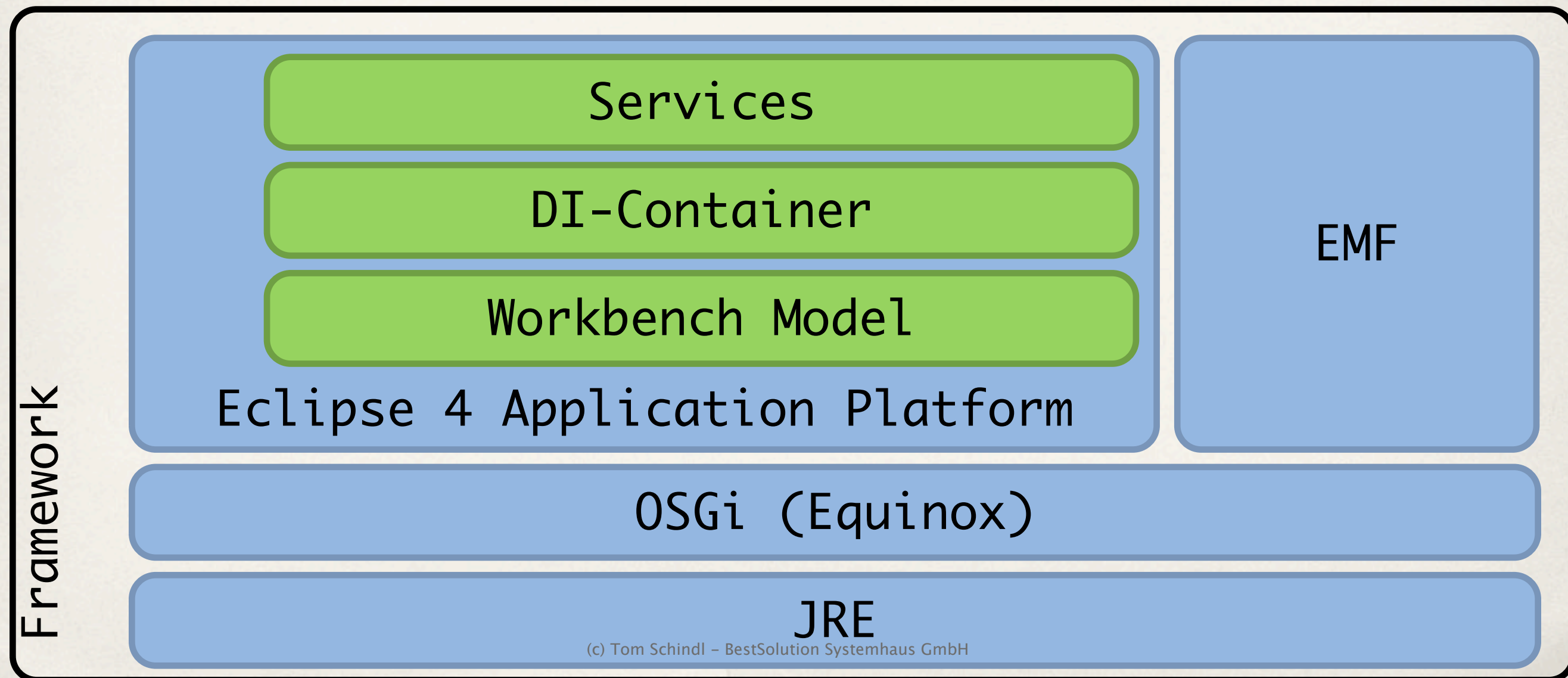
Create 3.x View

Eclipse 4.1 Application Platform

Eclipse 4.1 Application Platform

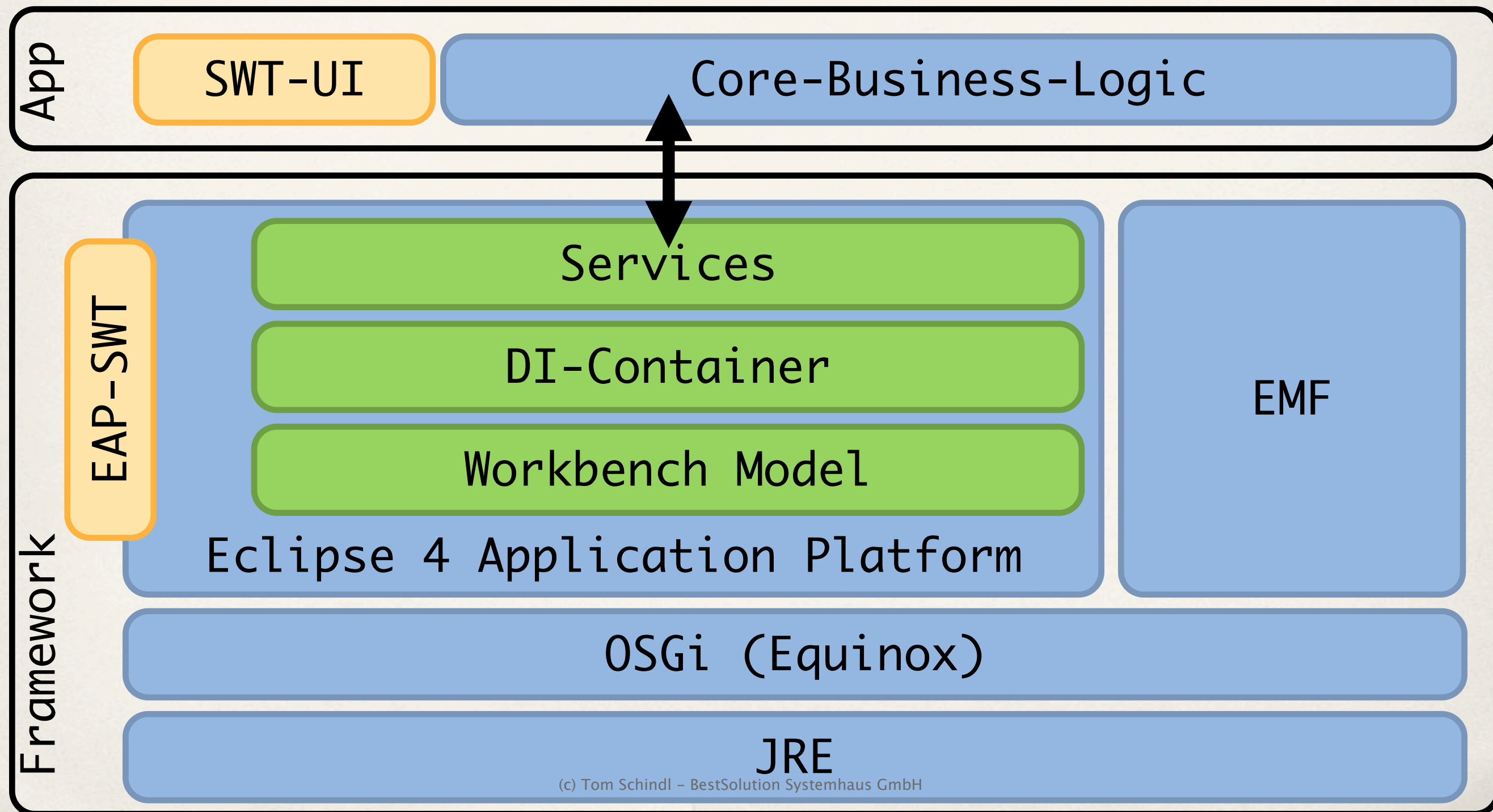


Eclipse 4.1 Application Platform

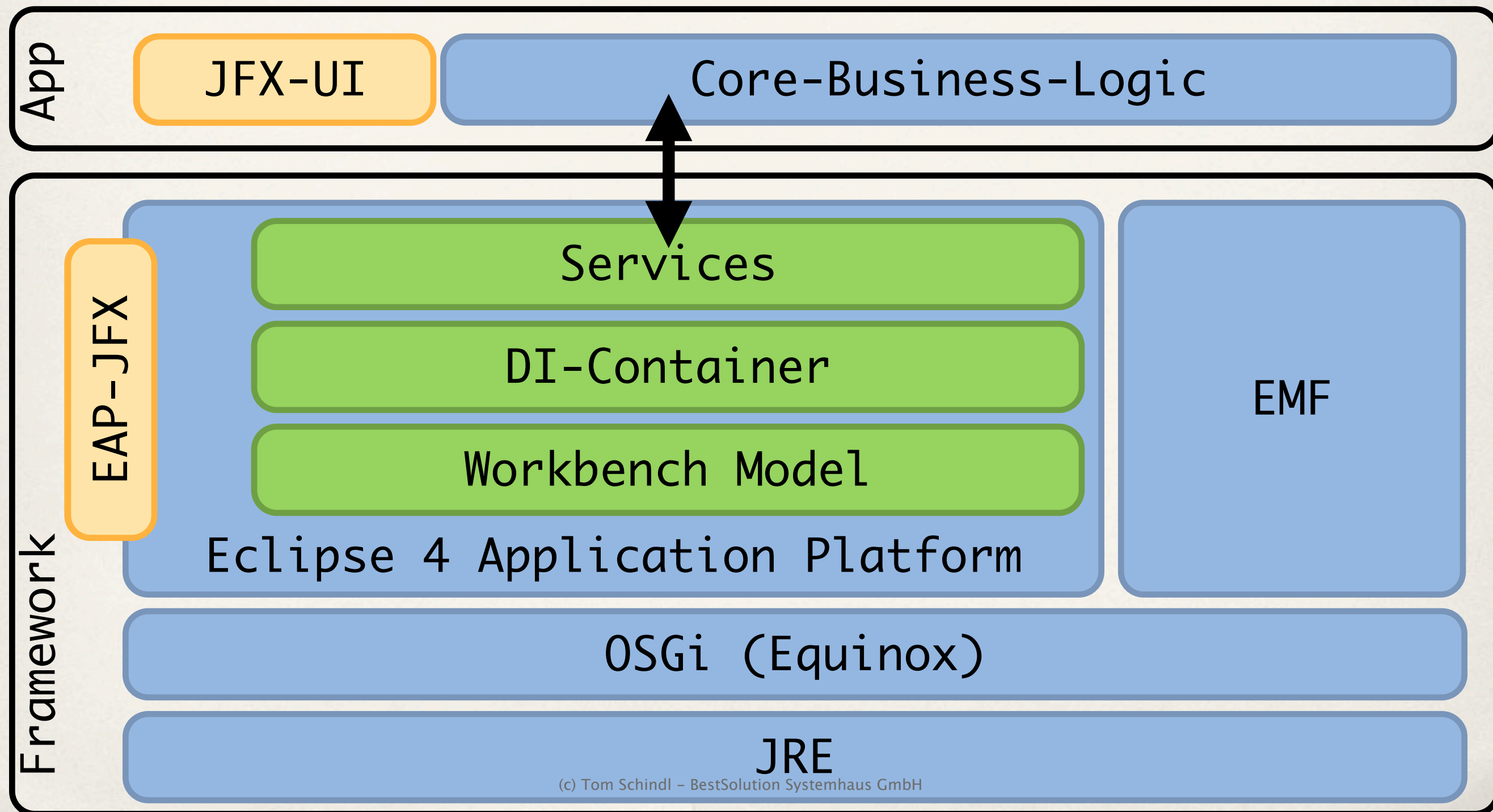


(c) Tom Schindl - BestSolution Systemhaus GmbH

Eclipse 4.1 Application Platform



Eclipse 4.1 Application Platform



DEMO TIME

e4-App

What does the future hold?

- ❖ All major tooling feature are done with 0.0.13
 - ❖ CSS-Editor
 - ❖ FXML and fxgraph-Editor
- ❖ Integration with external tools (Scenebuilder, FXperience-Tools)
- ❖ All runtime features are done
 - ❖ Maybe Felix support?
- ❖ e4 integration will stay in prototype stage longer

What does the future hold?

- ❖ Moveing to Eclipse
 - ❖ Maybe after Juno Release done
 - ❖ Freeing our bandwidth (last release ~2.400 downloads in 8 weeks)
 - ❖ Problem that PDE has to be patched for now

Resources

- ❖ Homepage: <http://www.efxclipse.org>
- ❖ My blog: <http://tomsondev.bestsolution.at>
- ❖ e4-Wiki: <http://wiki.eclipse.org/e4>
- ❖ e4-newsgroup: eclipse.e4
- ❖ e4-mailinglist: e4-dev@eclipse.org
- ❖ Twitter: @tomsontom