Eclipse 4 on JavaFX

BestSolution

Tom Schindl <tom.schindl@bestsolution.at>

Twitter: @tomsontom
Blog: http://tomsondev.bestsolution.at
Website: http://www.bestsolution.at

About Tom

- CTO BestSolution.at Systemhaus GmbH
- Eclipse Committer
  - e4
  - Platform
  - EMF
- Project lead
  - e(fx)clipse
- Twitter: @tomsontom
- Blog: tomsondev.bestsolution.at
- Corporate: http://bestsolution.at

Agenda

BestSolution

- First hour
  - Something about application architecture
  - Learn about e(fx)clipse APIs
  - DI: @Preference, @ContextValue and EventBus
  - Transitions for Windows and Perspectives
  - Dialogs - heavyweight and lightweight

Agenda

- second hour
  - apply APIs in a bigger sample (3d lego builder)
- third hour
  - code editor development
  - apply APIs in a prepared TypeScript-Editor sample

e4 application dev

BestSolution

- you need to set a target platform - development against your IDE runtime platform is IMPOSSIBLE
- e4 runtime is delivered as a self-contained p2-repository to make is as simple as possible for the most use cases
- for BND-lovers we also provide a self-contained r5 bundle repository
- as of today PDE is best supported dev-time tooling (THIS WILL CHANGE IN FUTURE!)

Preparation

- Copy the Eclipse SDK matching your OS & Bitness to your computer, extract it and launch the eclipse instance (remember the location!)

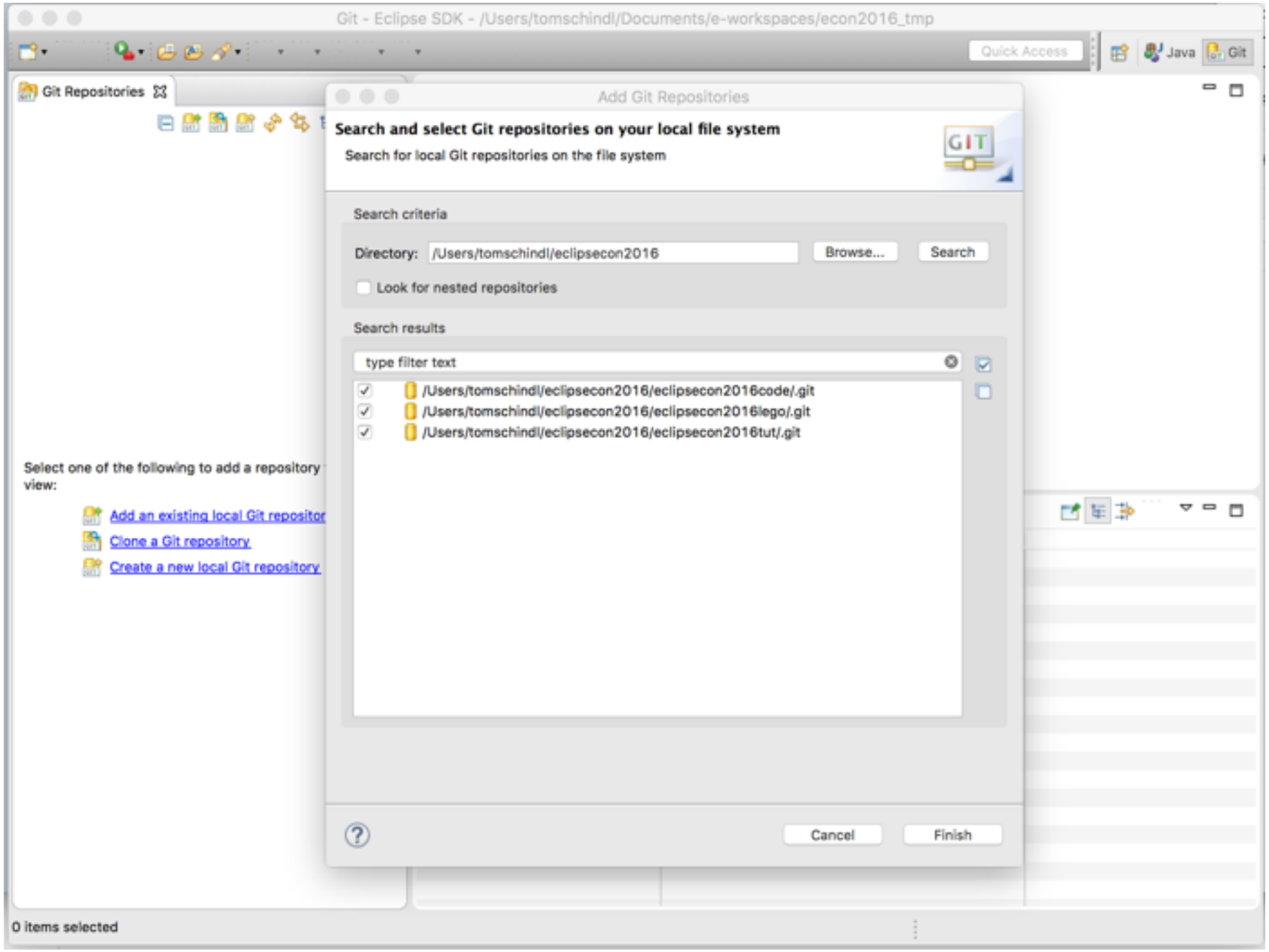
- Copy the sources.zip from the USB-Stick to your computer and extract it somewhere on your hard drive into the workspace-folder you've chose above

- Open the Git-Perspective and select „Add an existing local git repository“

- Navigate workspace-folder and select the 3 git repositories shown

Preparation

BestSolution



Setup a target

- Open the Preferences and navigate to „Plug-in Development“ > „Target Platform“
- Select „Add..“
- Start with an empty target and select „Next >“
- On the next page select „Add…“ and select the „Software Site entry“ followed by selecting „Next“
- In Dialog select „Add…“ and followed by the selecting „Archive…“

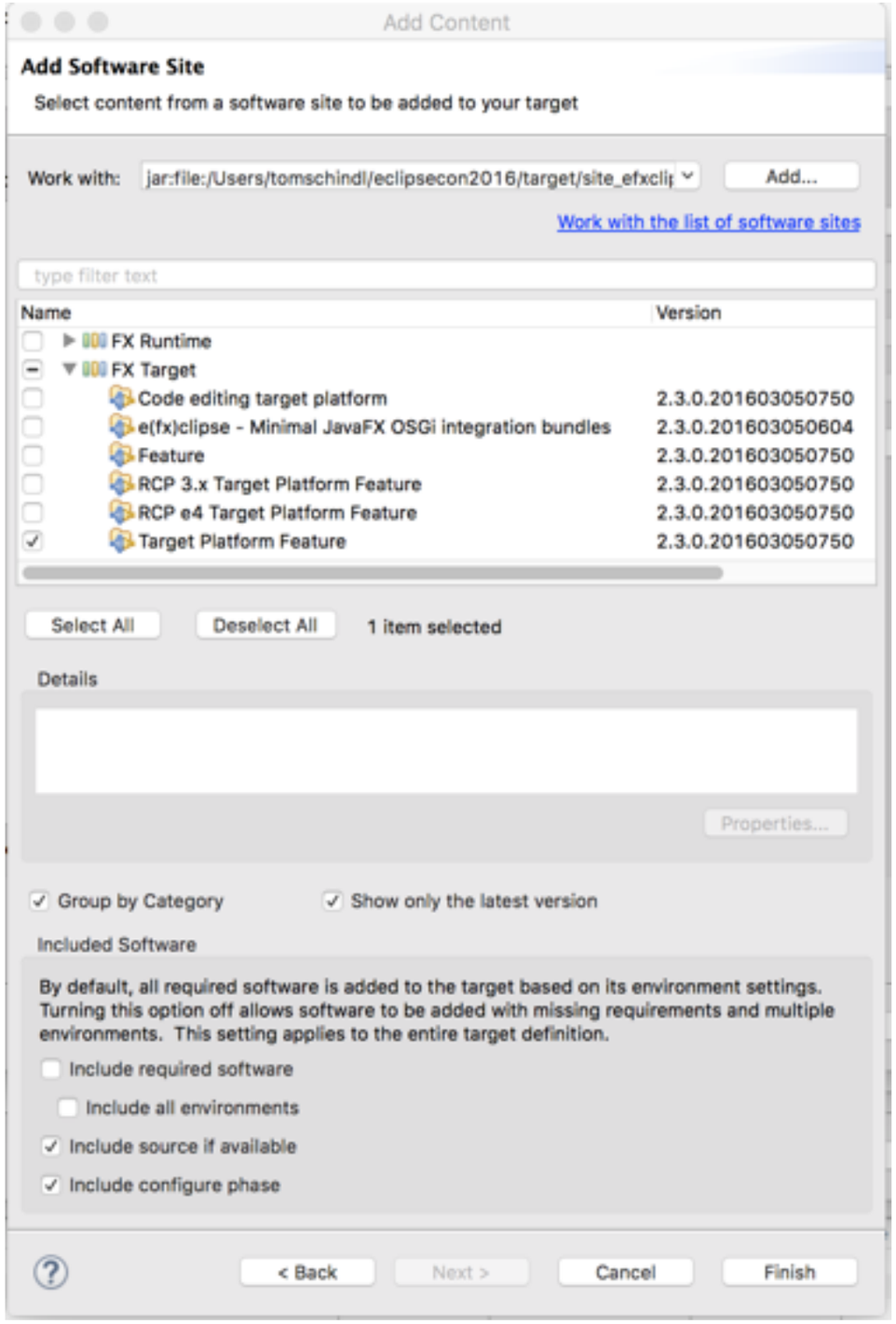
Setup a target

- Navigate to your workspace and inside the target-folder choose the site_efxclipse.zip archive and finish the dialog with selecting „OK“

- Select the „Target Platform Feature“ and UNCHECK „Include required software“ (see screenshot on next slide)

- Finish up (don’t forget to activate the target)

Setup target



Lab 1

- Bootstrap a e4 JavaFX application
  - Use the wizard in New > Project … > JavaFX > OSGI > e4 application projects to create a project use „my.sample“ as the bundle prefix
  - Add a „TrimmedWindow“ element, give it a label, …
  - Add a „Part“ element as the child of the window
  - Create a MyView class in your project who gets a BorderPane injected on @PostConstruct

Application architecture

BestSolution

- What is a good architecture

Requirement	Strategy
Loosely coupled	DI, Services and EventBus
Avoid heavy framework deps	DI with e(fx)clipse extensions

App architecture in e4

BestSolution

- You can depend on
  - Your UI-Toolkit technology (SWT, JavaFX, …)
  - Your services
- You can NOT depend on e4 APIs
  - IEclipseContext
  - IEventBroker, e4 @Preference because of API-Leakage
  - OSGi-APIs

IEclipseContext

BestSolution

- What's the deal with IEclipseContext

```
public class V1 {
    @Inject
    IEclipseContext context;

    @PostConstruct
    void init(BorderPane parent) {
        ListView<String> data = new ListView<>();
        data.setItems(FXCollections.observableArrayList("A","B","C"));
        data.getSelectionModel()
            .selectedItemProperty().addListener(this::handleSelection);
        parent.setCenter(data);
    }

    private void handleSelection(Observable o, String oldV, String newV) {
        context.getParent() // get the window context
            .set("val", newV);
    }
}
```

IEclipseContext

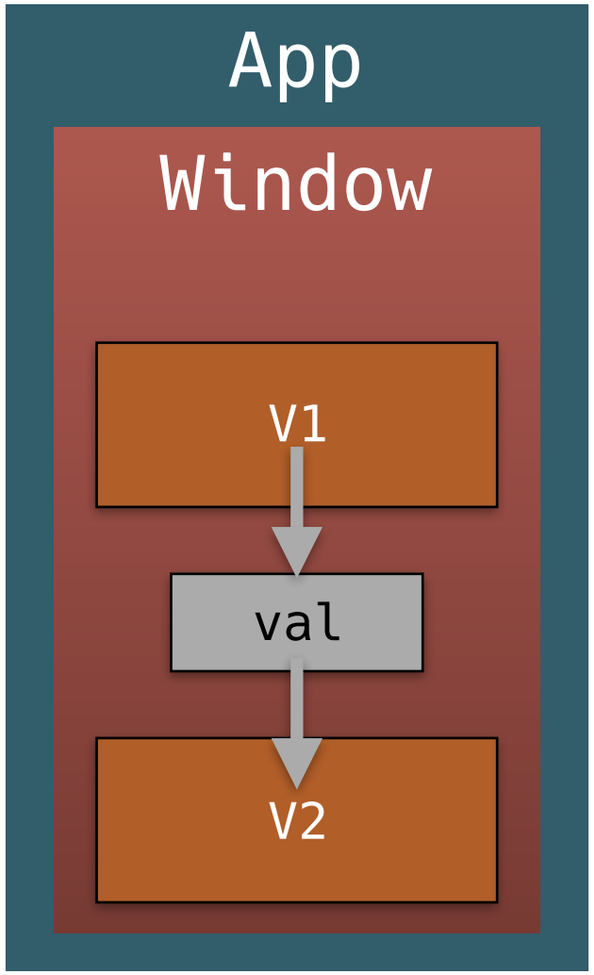
BestSolution

- What's the deal with IEclipseContext

```
public class V1 {
    @Inject
    IEclipseContext context;

    @PostConstruct
    void init(BorderPane parent) {
        ListView<String> data = new ListView<>();
        data.setItems(FXCollections.observableArrayList("A","B","C"));
        data.getSelectionModel()
            .selectedItemProperty().addListener(this::handleSelection);
        parent.setCenter(data);
    }

    private void handleSelection(Observable o, String oldV, String newV) {
        context.getParent() // get the window context
            .set("val", newV);
    }
}
```



IEclipseContext

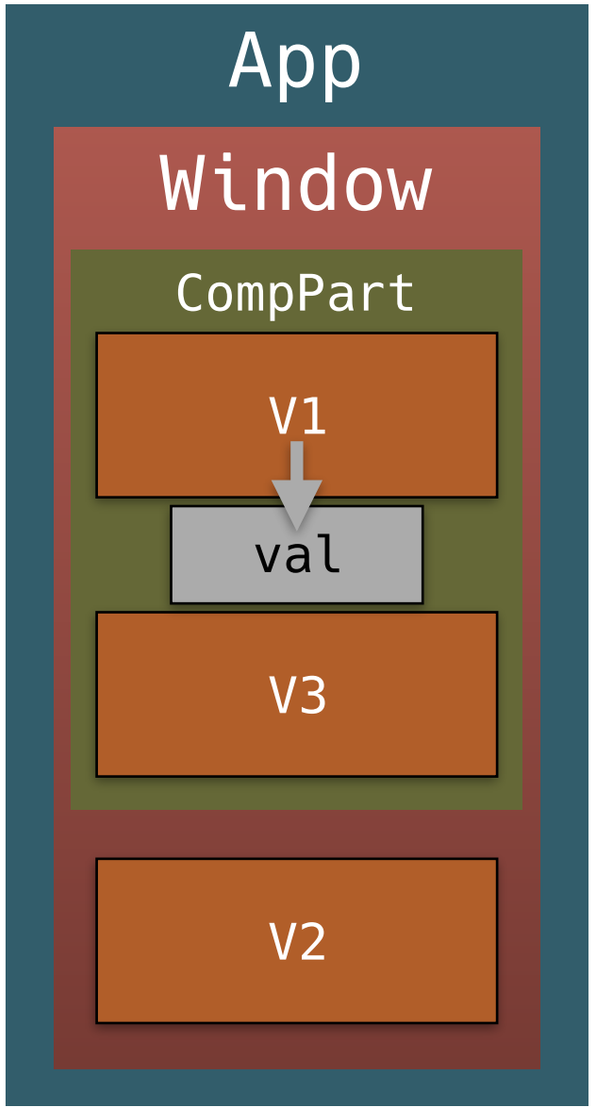
BestSolution

- What’s the deal with IEclipseContext

```
public class V1 {
    @Inject
    IEclipseContext context;

    @PostConstruct
    void init(BorderPane parent) {
        ListView<String> data = new ListView<>();
        data.setItems(FXCollections.observableArrayList("A","B","C"));
        data.getSelectionModel()
            .selectedItemProperty().addListener(this::handleSelection);
        parent.setCenter(data);
    }

    private void handleSelection(Observable o, String oldV, String newV) {
        context.getParent() // get the window context
            .set("val", newV);
    }
}
```



IEclipseContext

- IEclipseContext#declareModifiable & modify

```
public class V1 {
    @Inject
    IEclipseContext context;

    @PostConstruct
    void init(BorderPane parent) {
        ListView<String> data = new ListView<>();
        data.setItems(FXCollections.observableArrayList("A","B","C"));
        data.getSelectionModel()
            .selectedItemProperty().addListener(this::handleSelection);
        parent.setCenter(data);
    }

    private void handleSelection(Observable o, String oldV, String newV) {
        context.modify("val", newV);
    }
}
```

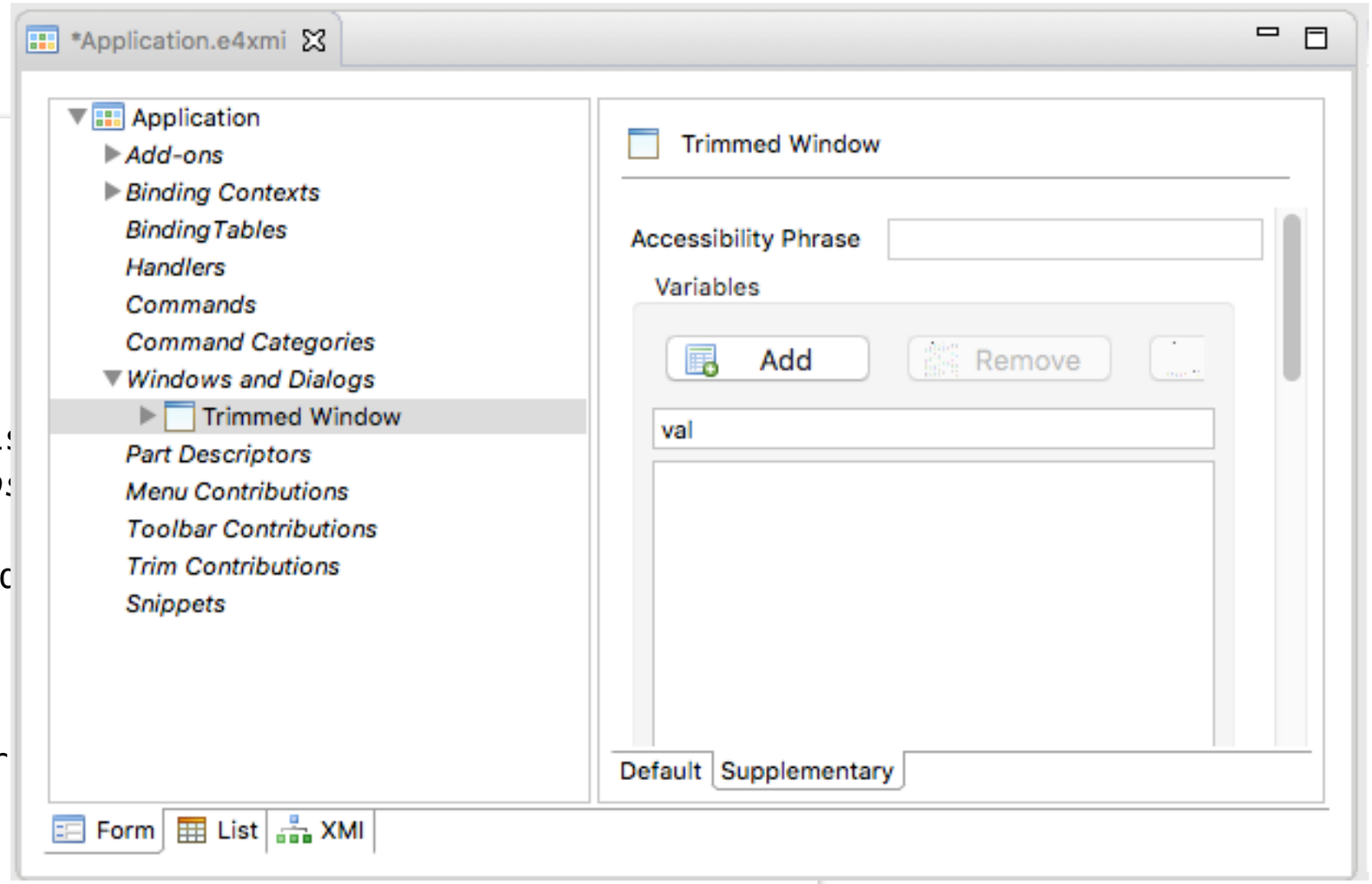
IEclipseContext

- IEclipseContext#declareModifiable & modify

```
public class V1 {
    @Inject
    IEclipseContext context;

    @PostConstruct
    void init(BorderPane parent) {
        ListView<String> data = new Li
        data.setItems(FXCollections.obs
        data.getSelectionModel()
            .selectedItemProperty().add
        parent.setCenter(data);
    }

    private void handleSelection(Obser
        context.modify("val", newV);
    }
}
```



IEclipseContext

BestSolution

- Think of publishing an Atomic-Operation always bound to the same key

```
public class V1 {
    @Inject
    IEclipseContext context;

    Consumer<String> publish = v -> context.modify("val", v);

    // ...

    private void handleSelection(Observable o, String oldV, String newV) {
        publish.accept(newV);
    }
}
```

IEclipseContext

BestSolution

- Enhance the DI-Container to create Consumer

```
public class V1 {
    @Inject
    @ContextValue(value = "val")
    Consumer<String> publish;
    // ...

    private void handleSelection(Observable o, String oldV, String newV) {
        publish.accept(newV);
    }
}
```

IEclipseContext

BestSolution

- IEclipseContext is not a one way structure - why not present it as a JavaFX-Property!

```
public class V1 {
    @Inject
    @ContextValue(value = "val")
    Property<String> publish;
    // ...

    private void handleSelection(Observable o, String oldV, String newV) {
        publish.set(newV);
    }
}
```

IEclipseContext

BestSolution

- … but if it is a JavaFX-Property why not bind directly

```
public class V1 {
    @Inject
    @ContextValue(value = "val")
    Property<String> publish;

    @PostConstruct
    void init(BorderPane parent) {
        ListView<String> data = new ListView<>();
        data.setItems(FXCollections.observableArrayList("A","B","C"));
        publish.bind(data.getSelectionModel()
            .selectedItemProperty());
        parent.setCenter(data);
    }

}
```

Lab 2

- Import **„at.bestsolution.e4fx.*“** to your workspace using the git perspective
- Adjust V1 to NOT depend on **IEclipseContext** anymore

Event Publishing

- What’s the deal with IEventBroker?

Event Publishing

BestSolution

- Who

Hierarchical view of plug-ins required by 'org.eclipse.e4.core.services (2.0.0.v20150403-1912)':

- org.eclipse.e4.core.services (2.0.0.v20150403-1912)
  - javax.annotation.jre (1.2.0.201510061341)
  - javax.inject (1.0.0.v20091030)
  - org.eclipse.core.jobs (3.7.0.v20150330-2103)
    - org.eclipse.equinox.common (3.7.0.v20150402-1709)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
  - org.eclipse.e4.core.contexts (1.4.0.v20150828-0818)
    - javax.inject (1.0.0.v20091030)
    - org.eclipse.e4.core.di (1.5.0.v20150421-2214)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
  - org.eclipse.e4.core.di (1.5.0.v20150421-2214)
    - javax.annotation.jre (1.2.0.201510061341)
    - javax.inject (1.0.0.v20091030)
    - org.eclipse.e4.core.di.annotations (1.4.0.v20150528-1451)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
  - org.eclipse.equinox.common (3.7.0.v20150402-1709)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
  - org.eclipse.equinox.preferences (3.5.300.v20150408-1437)
    - org.eclipse.equinox.common (3.7.0.v20150402-1709)
    - org.eclipse.equinox.registry (3.6.0.v20150318-1503)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
    - org.eclipse.equinox.preferences (3.5.300.v20150408-1437)
  - org.eclipse.osgi (3.10.101.v20150820-1432)
    - org.eclipse.osgi.compatibility.state (1.0.100.v20150402-1551)
  - org.eclipse.osgi.services (3.5.0.v20150519-2006)
    - javax.servlet (3.1.0.v201410161800)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
    - org.eclipse.osgi.services (3.5.0.v20150519-2006)

Event Publishing

BestSolution

- e(fx)clipse runtime ships a tiny EventBus interface which delegates in an e4 world to IEventBroker!

```
public class V1 {
    @Inject
    IEventBroker broker;

    @PostConstruct
    void init(BorderPane parent) {
        Timer t = new Timer();
        t.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                broker.send(Constants.CURRENT_TIME, new Date().getTime());
            }
        }, 0, 1_000);
    }

}
```

```
public class Constants {
    public static final String CURRENT_TIME = "at/bestsolution/e4fx/ui/currentTime";
}
```

Event Publishing

BestSolution

- e(fx)clipse EventBus

```
public class V1 {
    @Inject
    IEventBroker broker;

    @PostConstruct
    void init(BorderPane parent) {
        Timer t = new Timer();
        t.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                broker.publish(Constants.CURRENT_TIME, new Date().getTime(), true);
            }
        }, 0, 1_000);
    }

}
```

Event Publishing

BestSolution

- What’s the problem with @UIEventTopic?
  - Puts unneeded pressure on the DI-System
  - Lacks typesafety

Event Publishing

- What's the problem with @UIEventTopic?
  - Puts unneeded pressure on the DI-System
  - Lacks typesafety

```
public class V2 {
    @Inject
    @Optional
    public void updateTime(@UIEventTopic(Constants.CURRENT_TIME) long time) {
        currentTime.setText(new Date(time).toString());
    }
}
```

Event Publishing

- What's the problem with @UIEventTopic?
  - Puts unneeded pressure on the DI-System
  - Lacks typesafety

```
public class V2 {
    @Inject
    @Optional
    public void updateTime(@UIEventTopic(Constants.CURRENT_TIME) long time) {
        currentTime.setText(new Date(time).toString());
    }
}
```

```
broker.send(Constants.CURRENT_TIME, new Date());
```

Event Publishing

- e(fx)clipse EventBus provides Typesafety through
  - Uses Topic<T> instances instead of a simple String

```
public class Constants {
    public static final Topic<Long> CURRENT_TIME = new Topic<>("at/bestsolution/e4fx/ui/currentTime");
}
```

```
public class V1 {
    @Inject
    EventBus broker;
    @PostConstruct
    void init(BorderPane parent) {
        Timer t = new Timer();
        t.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                broker.publish(Constants_solution.CURRENT_TIME, new Date(), true);
            }
        }, 0, 1_000);
    }
}
```

Event Publishing

- e(fx)clipse EventBus provides Typesafety through
  - Uses Topic<T> instances instead of a simple String

```
public class Constants {
    public static final Topic<Long> CURRENT_TIME = new Topic<>("at/bestsolution/e4fx/ui/currentTime");
}
```

```
public class V1 {
    @Inject
    EventBus broker;
    @PostConstruct
    void init(BorderPane parent) {
        Timer t = new Timer();
        t.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                broker.publish(Constants_solution.CURRENT_TIME, new Date(), true);
            }
        }, 0, 1_000);
    }
}
```

Compile Error

Event Publishing

- Replace your @UIEventTopic usage to get typesafety and remove DI system pressure

```
public class V2_solution {
    @Inject
    public V2_solution(BorderPane parent, EventBus eventBus) {
        eventBus.subscribe(Constants_solution.CURRENT_TIME, EventBus.data( this::updateTime ));
    }
    public void updateTime(long time) {
        currentTime.setText(new Date(time).toString());
    }
}
```

Lab 3

BestSolution

- Update V1, V2, Constants in the sample to
  - Use the EventBus
  - Makes use of typesafety features
  - send a Date() instance instead of a long

Preference publishing

BestSolution

- What's the deal with publishing preferences

```
public class V1 {
    @Inject
    @Preference
    IEclipsePreferences preference;

    @PostConstruct
    void init(BorderPane parent) {
        data = new ListView<>();
        data.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
        data.getSelectionModel()
            .getSelectedItems().addListener(this::handleSelectionListChange);
    }

    private void handleSelectionListChange(Change<? extends String> o) {
        String value = data.getSelectionModel().getSelectedItems()
                .stream()
                .collect(Collectors.joining(","));
        preference.put(Constants.PREFERENCE_KEY, value);
        try {
            preference.flush();
        } catch (BackingStoreException e) {}
    }
```

Preference publishing

BestSolution

- What's the deal with publishing preferences

```
public class V1 {
    @Inject
    @Preference
    IEclipsePrefe

    @PostConstruc
    void init(Bor
        data = ne
        data.getS
        data.getS
            .getS
    }

    private void
        String va
                .s
                .c
        preferenc
        try {
            prefe
        } catch (
    }
```

- org.eclipse.e4.core.di.extensions (0.13.0.v20150421-2214)
  - javax.annotation.jre (1.2.0.201510061341)
  - javax.inject (1.0.0.v20091030)
  - org.eclipse.e4.core.di (1.5.0.v20150421-2214)
    - javax.annotation.jre (1.2.0.201510061341)
    - javax.inject (1.0.0.v20091030)
    - org.eclipse.e4.core.di.annotations (1.4.0.v20150528-1451)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
  - org.eclipse.equinox.preferences (3.5.300.v20150408-1437)
    - org.eclipse.equinox.common (3.7.0.v20150402-1709)
    - org.eclipse.equinox.registry (3.6.0.v20150318-1503)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
    - org.eclipse.equinox.preferences (3.5.300.v20150408-1437)
  - org.eclipse.osgi (3.10.101.v20150820-1432)
    - org.eclipse.osgi.compatibility.state (1.0.100.v20150402-1551)
  - org.eclipse.osgi.services (3.5.0.v20150519-2006)
    - javax.servlet (3.1.0.v201410161800)
    - org.eclipse.osgi (3.10.101.v20150820-1432)
    - org.eclipse.osgi.services (3.5.0.v20150519-2006)

Preference publishing

- Repeat the the thought experiment from IEclipseContext

Preference publishing

BestSolution

- Repeat the the thought experiment from IEclipseContext

```
public class V1 {
    @Inject
    @org.eclipse.fx.core.preferences.Preference(key = Constants.PREFERENCE_KEY)
    Consumer<String> preference;

    private void handleSelectionListChange(Change<? extends String> o) {
        String value = data.getSelectionModel().getSelectedItems()
                .stream()
                .collect(Collectors.joining(","));
        preference.set(value);
    }
}
```

Preference publishing

- e(fx)clipse preference support none primitive preferences

Preference publishing

BestSolution

- e(fx)clipse preference support none primitive preferences

```
public class V1 {
    @Inject
    @org.eclipse.fx.core.preferences.Preference(key = Constants.PREFERENCE_KEY)
    Consumer<List<String>> preference;

    private void handleSelectionListChange(Change<? extends String> o) {
        preference.accept(data.getSelectionModel().getSelectedItems());
    }

}
```

Preference publishing

BestSolution

- e(fx)clipse preference support none primitive preferences

```
public class V1 {
    @Inject
    @org.eclipse.fx.core.preferences.Preference(key = Constants.PREFERENCE_KEY)
    Consumer<List<String>> preference;

    private void handleSelectionListChange(Change<? extends String> o) {
        preference.accept(data.getSelectionModel().getSelectedItems());
    }

}
```

```
public class V2 {
    @Inject
    public void setPreferenceValues(
      @org.eclipse.fx.core.preferences.Preference(key=Constants.PREFERENCE_KEY)
      List<String> values) {
        preferenceValue.setText(values != null ?
                values.stream().collect(Collectors.joining(",")) : "");
    }
}
```

Lab 4

- Update V1 and V2 to use e(fx)clipse preferences

Window Transitions

- Implemented as a service
  - …renderers.base.services.WindowTransitionService<Stage>
  - Provided through the OSGi-Service registry

Window Transitions

BestSolution

- Implemented as a service
  - …renderers.base.services.WindowTransitionService<Stage>
  - Provided through the OSGi-Service registry

```
public class WindowTransitionService implements WindowTransitionService<Stage> {
    @Override
    public AnimationDelegate<Stage> getShowDelegate(
            MWindow window) {
        return new AnimationDelegate<Stage>() {

            @Override
            public void animate(Stage window, Runnable finished) {
                window.setOpacity(0);
                window.show();
                //…
            }
        };
    }
}
```

Perspective Transitions

- Implemented as a service
  - …renderers.base.services.PerspectiveTransitionService<BorderPane,Node>
  - Provided through the OSGi-Service registry
  - Make use of provided transitions in org.eclipse.fx.ui.animation.pagetransition.animation

Perspective Transitions

- Implemented as a service
  - …renderers.base.services.PerspectiveTransitionService<BorderPane,Node>
  - Provided through the OSGi-Service registry

```
public class PerspectiveTransition implements PerspectiveTransitionService<BorderPane, Node> {

    @Override
    public AnimationDelegate<BorderPane, Node> getDelegate(MPerspective fromPerspective,
        MPerspective toPerspective) {
        return new AnimationDelegate<BorderPane, Node>() {

            @Override
            public void animate(BorderPane container, Node control, Runnable finished) {
                FadeAnimation animation = new FadeAnimation();
                animation.animate(container, control, finished);
            }
        };
    }
```

Lab 5

BestSolution

- Make the main window fade in
  - Use the provided StageFadeTransition
- Perspective transitions
  - Add a second perspective
  - Move the V2-Part to there
  - Add a transition by using the one of the prebuilt animations

Dialogs

- e(fx)clipse provides JFace like dialogs through org.eclipse.fx.dialogs
  - TitleAreaDialog
  - MessageDialog
- There are lightweight dialogs available in an e4 environment through a special service named org.eclipse.fx.ui.services.dialog.LightWeightDialogService

Lightweight dialogs

- Advantage
  - Can have narrower scope (Window,Perspective,Part) than heavyweight dialogs
  - You can do fancy animations for hiding/showing them
- Disadvantage
  - Can not moved outside the physical application window so they are only suitable for modal dialogs

Opening a Dialog

- You need to create a class who is a subclass of javafx.scene.Node and implements the org.eclipse.fx.ui.controls.stage.Frame interface

- Default implementations provided as part of the org.eclipse.fx.ui.controls bundle
  - Basic-Dialog: org.eclipse.fx.ui.controls.dialog.Dialog
  - Title-Dialog: …TitleAreaDialog
  - Message-Dialogs: …MessageDialog

Opening a Dialog

Opening a Dialog

BestSolution

```java
public class Dialog {
    @Execute
    public void run(LightWeightDialogService dialogService) {
        dialogService.openDialog(MyLightweightDialog.class, ModalityScope.WINDOW);
    }

    static class MyLightweightDialog extends TitleAreaDialog {
        @Inject
        public MyLightweightDialog(MApplication application) {
            super("Hello World", "Hello World", "This is a hello world message");
            setClientArea(createClientArea(application));
            addDefaultButtons();
            setPrefWidth(500);
        }
    }
}
```

Opening a Dialog

- If you want the dialog to open in an animated fashion provide a service of type org.eclipse.fx.ui.workbench.renderers.fx.services.LightweightDialogTransitionService
- There are 2 default implementations available
  - FlyInTransitionService
  - FadeDialogTransitionService
- Currently only Dialogs at the window level are shown animated

Lab6

- Finish the handler implementation named DialogSample
  - to show a TitleArea dialog with a simple message
  - add an animation for showing the dialog
- (optional) make the Dialog show a Tree of the current application model

Custom Window L&F

BestSolution

- Advantage
  - Provide a cross-platform L&F
  - No restrictions
- Disadvantage
  - You loose many features of the native window bar like docking, …

Custom Window L&F

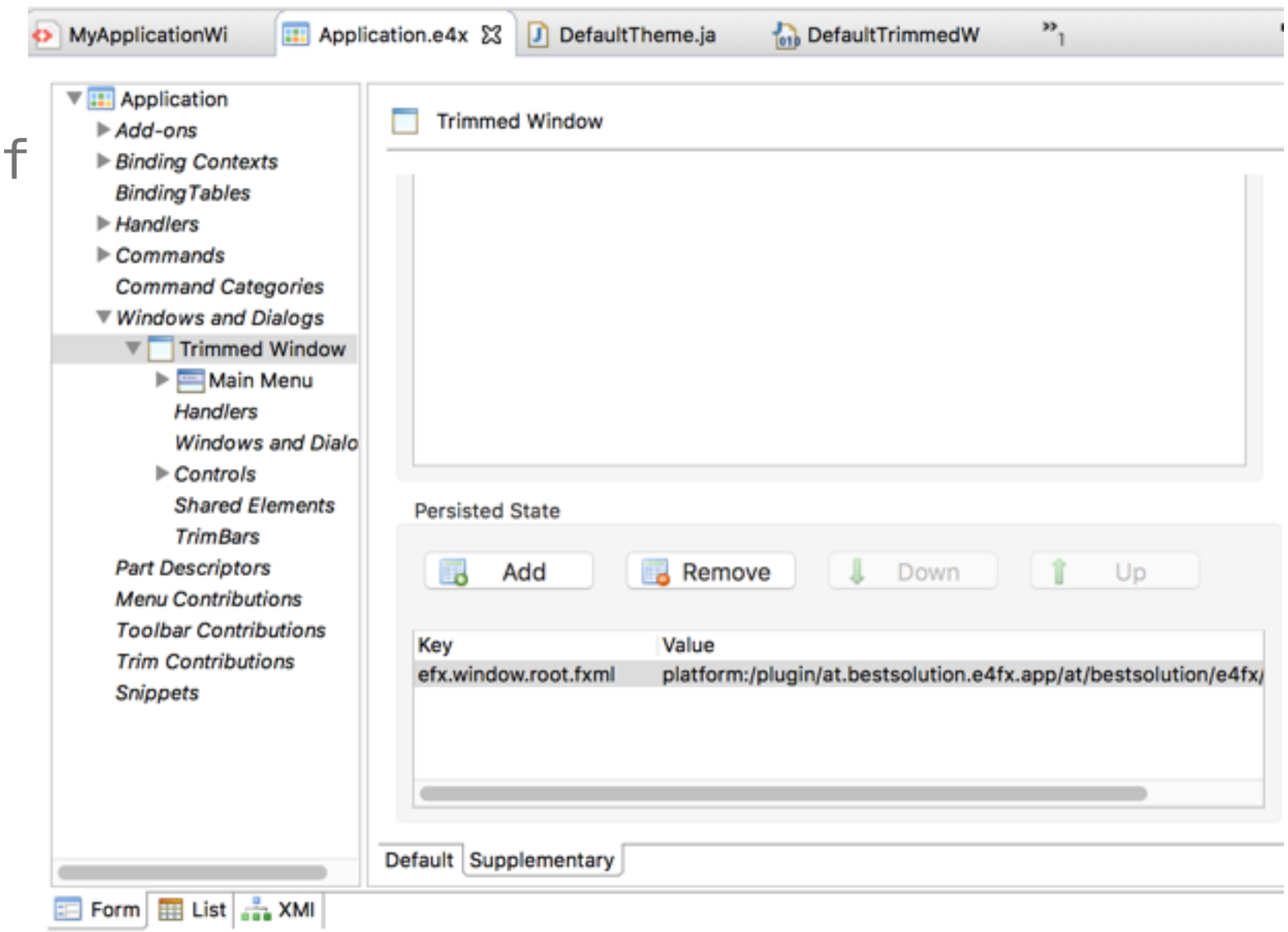
- You need to provide an FXML-File whose root-element is an org.eclipse.fx.ui.controls.stage.DefaultTrimmedWindowPane

- You need to set a persistedState-Property named „efx.window.root.fxml“ on the window you want custom decorations

Custom Window L&F

- You need to provide an FXML-File whose root-element is an org.eclipse.fx.ui.controls.stage.DefaultTrimmedWindowPane

- You need to set a „efx.window.root.f decorations



Custom Window L&F

- Custom L&F FXML
  - Needs to provide slots for positioning the various elements use Pane-Nodes with specific ids
    - MenuBar: menu-bar-area
    - Content-Area: client-area
    - Left/Right/Top/Bottom-Trim: left-trim-area, right-trim-area, top-trim-area, bottom-trim-area

Lab 7

- Modify the Application.e4xmi to load the MyApplicationWindow.xml file
  - As the url use platform:/plugin/at.bestsolution.e4fx.app/at/bestsolution/e4fx/app/MyApplicationWindow.fxml
- Play a bit around in the FXML eg add a big spacer between the menu and client area

Themes

- Themes as defined as OSGi-Services who implement the interface org.eclipse.fx.ui.services.theme.Theme
- A base implementation is available as org.eclipse.fx.ui.theme.AbstractTheme
- Themes are made up of
  - a primary CSS-File
  - Contributing further CSS-Files is done through OSGi-Services of type Stylesheet and MultiURLStylesheet

Themes

BestSolution

- The default theme is specified in the product-extension point

```
<extension id="product" point="org.eclipse.core.runtime.products">
    <product name="Sample" application="org.eclipse.fx.ui.workbench.fx.application" >

        <property name="cssTheme" value="theme.default" />
```

- Themes can be switch by using the org.eclipse.fx.ui.services.theme.ThemeManager-Service

Lab 8

BestSolution

- Add a new Dark theme who uses a css-file with the following content

```
.root {
    -fx-base: #4d5052;
    -fx-background: #4d5052;
    -fx-control-inner-background: #4d5052;
    -fx-text-base-color: #c7c7c7;
    -fx-font-size: 1em;
}
```

- Modify the plugin.xml to boot with that theme by default

- (Optional) Add a command, handler and keybinding to toggle the theme when hitting CTRL+T

Apply knowledge

- Prepare
  - Import all projects from eclipsecon2016lego-git repo
- Fix the following stuff
- TODO1 in AssemblyListView to publish the current assembly
- TODO2 in ModelViewer to retrieve the current selected assembly
- TODO3 Make AddBrick show the NewBrickDialog

Apply knowledge

- TODO4 Make the NewAssemblyDialog publish the new assembly through the EventBus
- TODO5 Update the assembly list in AssemblyListView when a new assembly is created
- TODO6 Make the Application.e4xmi use the CustomWindowDecoration.fxml
- TODO7 Uncomment the code in at.bestsolution.lego.app/default.css
- TODO 8 Contribute the LegoStylesheet to OSGi-Service-Reg

Apply knowledge (optional)

- Import all projects from eclipsecon2016code
- Open your target-Platform setting and add
  - Add the feature from site_typescript.zip
  - Add the appropriate feature for your platform from site_j2v8.zip

Apply knowledge (optional)

- Run the application and set a root-folder eg the tssample folder in „eclipsecon2016tut/tutorial/“
- TODO1 - Finish the implementation in NewFile to show a dialog
- TODO2 - Register the TypescriptConfigurationProvider to get syntax highlighting
- TODO3 - Register the TypescriptProposalComputerProvider to enabled auto complete

Apply knowledge (optional)

- TODO 4a/b: Register the TypescriptAnnotationModelTypeProvider and TypescriptAnnotationPresenterTypeProvider to get error markers, …