



Eclipse e4 im Projekt KIHUB

Risiko und Verzögerungen oder
High Speed?

Beat Schaller

Agenda

1. Geschichte und Gründe für E4
2. Übersicht der neuen Features
3. Live Model
4. Dependency Injection
5. OSGi und Model Services
6. Renderer und CSS-Styling
7. KIHUB und e4
8. Fragen



1.1 Basis Kai Tödter, Tom Schindl, Lars Vogel



1.2 Geschichte und Gründe für e4

Eclipse 3.x ist schwer wartbar und hat einige Altlasten:

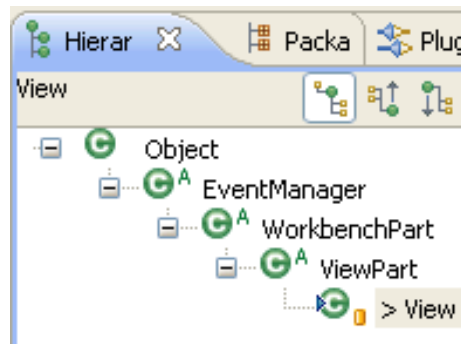
- Komplex
- Legacy Code auf Grund Abwärtskompatibilität
- Viele API's
- Singleton Problem
 - `PlatformUI.getWorkbench()`
 - `Platform.getExtensionRegistry()`
 - `ResourcePlugin.getWorkspace()`
- Schwer zu testen
- UI basiert auf einem starren Modell (Editors / Views)
- Unterschiedliche Registry (ViewRegistry, EditorRegistry)



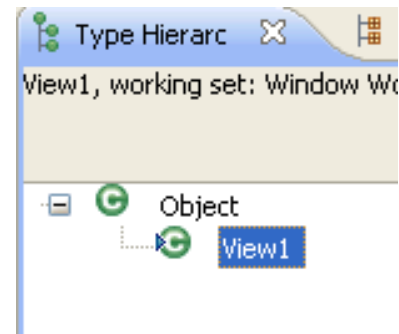
1.3 Geschichte und Gründe für e4

Vererbung

Eclipse 3.x



Eclipse e4



Alles wird zu einem POJO

1.4 Geschichte und Gründe für e4

Neue und bestehende Frameworks/Technologien

- RIA Frameworks wie Flex, Silverlight und JavaFX
- GWT, Ajax-Frameworks

Trend UI Philosophie

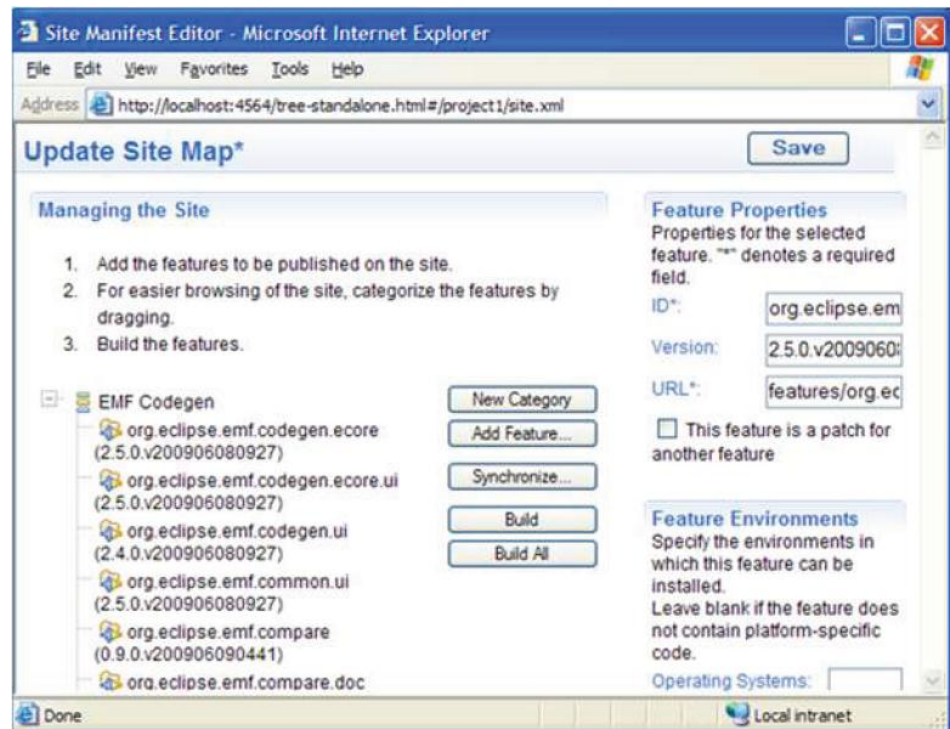
- Weg vom nativen UI

Web to Desktop

- JavaScript als Plugins

Desktop to Web

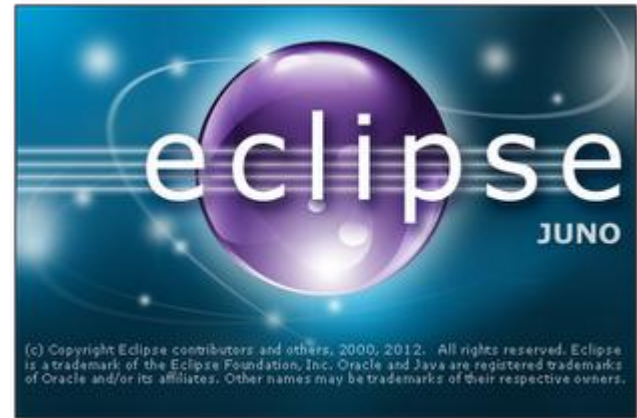
- RAP



E4 JavaScript PDE update site Editor

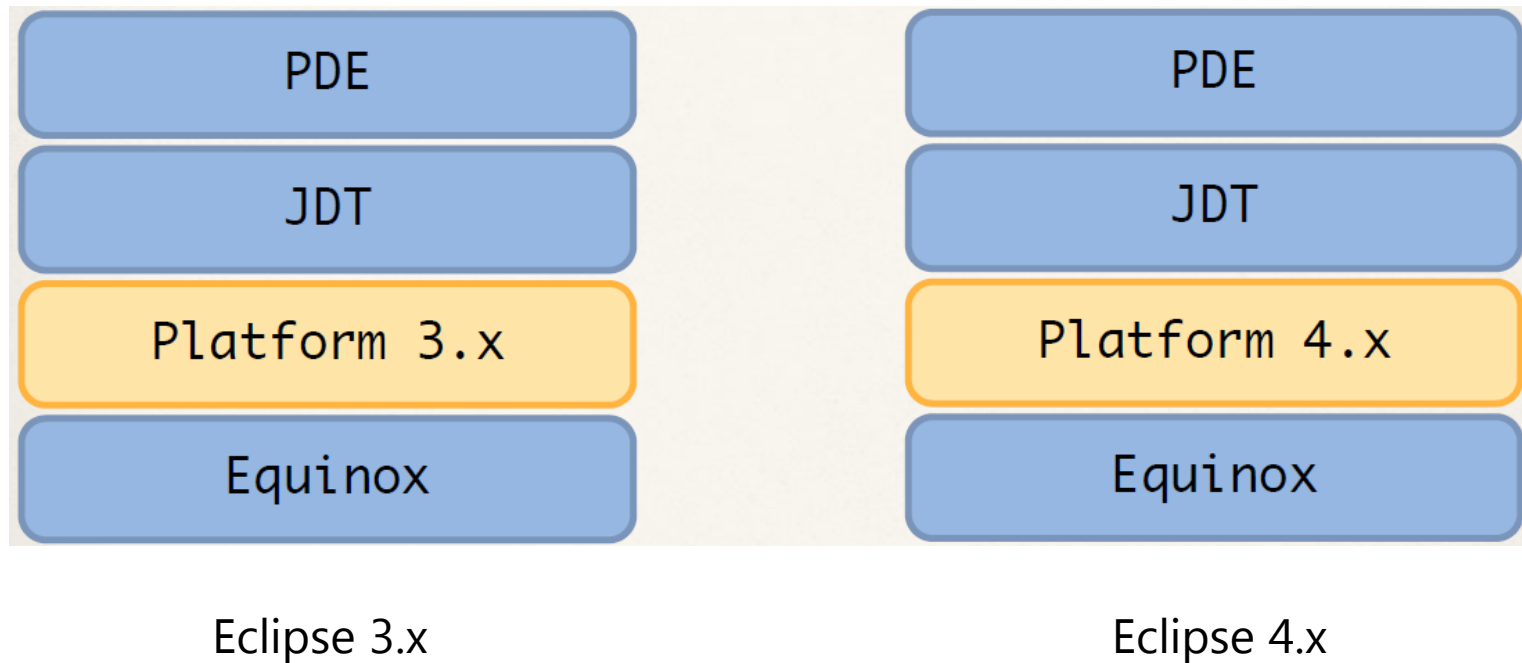
2.1 Übersicht der neuen Features

- Eclipse Applikation wird über ein Model definiert
- Das Model kann zur Entwicklungs- und Runtime verändert werden
- Das Model kann erweitert werden
- Eclipse e4 unterstützt Injection
- Styling über CSS
- Nicht mehr fixiert auf SWT -> Neue Renderer
- Kompatibilitätslayer für 3.x Applikationen

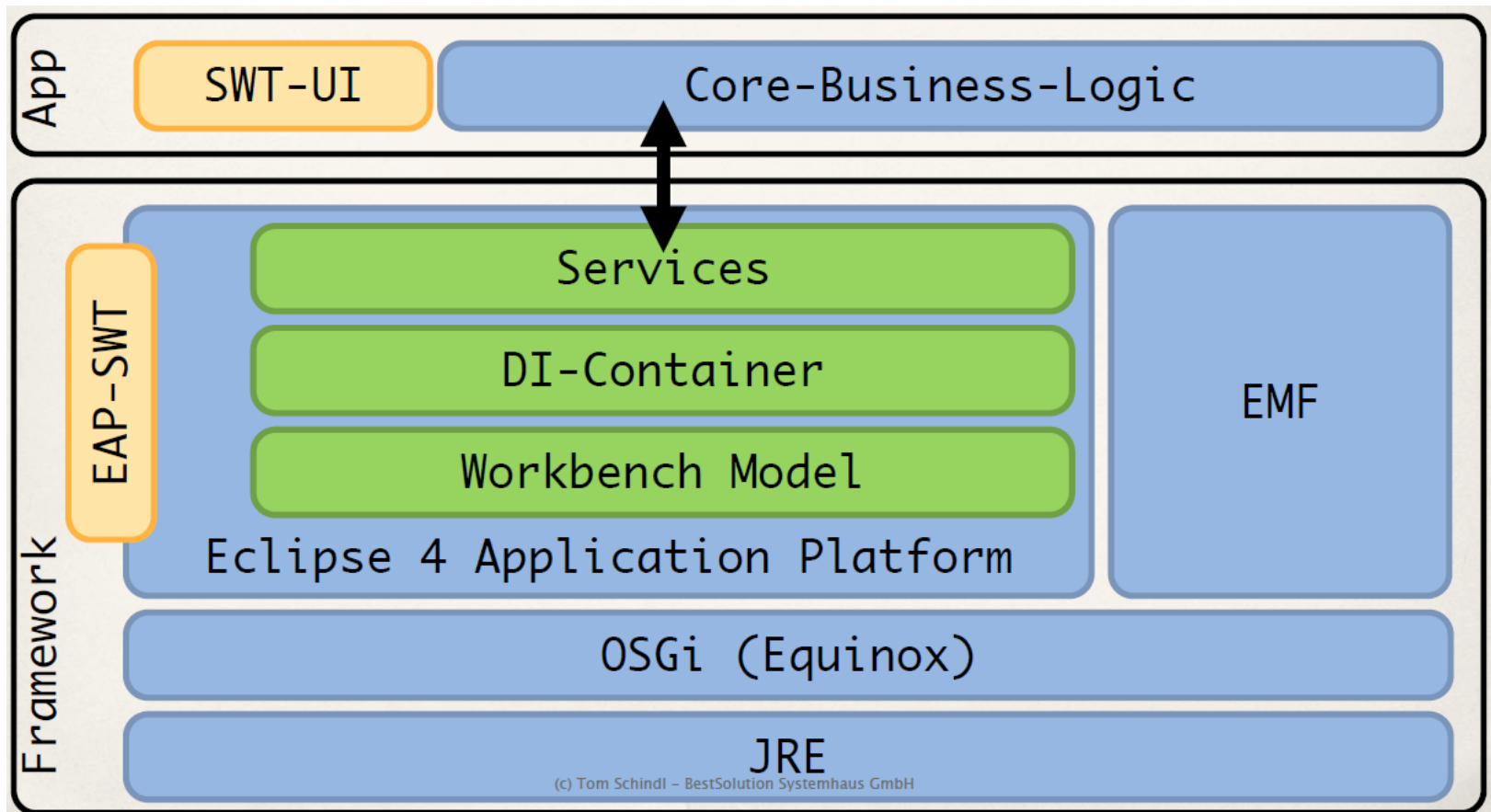


2.2 Übersicht Architektur

3.x versus e4



2.3 Übersicht Aufbau



3.1 Live Model Application

The screenshot displays the Eclipse IDE interface for a live model application. On the left, a tree view shows the application's structure, starting with 'Application' and branching into 'Addons', 'Binding Contexts', 'Handlers', 'Part Descriptors', 'Commands', 'Command Categories', and 'Windows'. The 'Windows' section is expanded, showing a 'Trimmed Window' which contains a 'Main Menu', 'Handlers', 'Windows', and 'Controls'. The 'Controls' section is further expanded, showing a 'PartSashContainer' which contains a 'Perspective Stack', a 'Label - Java', and another 'PartSashContainer'. This second 'PartSashContainer' contains a 'Part Stack' and a 'Placeholder - Area (org.eclipse.ui.editor)'. The 'Part Stack' contains several 'Placeholder' elements, including 'Part (Problems)', 'Part (Javadoc)', 'Part (Declaration)', and several 'not-rendered' placeholders. The right pane, titled 'Application', shows the 'XMI:ID' and 'Id' as 'org.eclipse.e4.legacy.ide.application'. Below this, the 'Binding Contexts' section shows a 'Binding Context - In Dialogs and Windows'. The central area displays a large 'Demo' text. The bottom status bar shows 'Form' and 'XMI' tabs.

3.2 Live Model Fragmente

The screenshot displays the Eclipse IDE interface for configuring a live model fragment. The Package Explorer on the left shows the project structure, including the 'ch.ipt.test' project and the 'fragment.e4xmi' file. The central editor shows the 'Part' configuration for 'MyPart' within the 'String Model Fragment'. The right-hand side displays the 'Part' configuration dialog with various properties:

- Id:** MyPart
- Label:** MyPart
- Accessibility Phrase:**
- Tooltip:**
- Icon URI:**
- Class URI:** bundleclass:///ch.ipt.extend/ch.ipt.extend.part.MyPart
- ToolBar:** ☐
- Container Data:**
- Closeable:** ☐
- To Be Rendered:** ☒
- Visible:** ☒
- Binding Contexts:**
- Persisted State:**

Key	Value
-----	-------

Demo

3.3 Live Model plugin.xml

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

  <extension
    id="product"
    point="org.eclipse.core.runtime.products">
    <product
      name="ch.sbb.test"
      application="org.eclipse.e4.ui.workbench.swt.E4Application">
      <property
        name="appName"
        value="ch.sbb.test">
      </property>
      <property
        name="applicationXMI"
        value="ch.sbb.test/Application.e4xmi">
      </property>
      <property
        name="applicationCSS"
        value="platform:/plugin/ch.sbb.test/css/default.css">
      </property>
    </product>
  </extension>

</plugin>
```

Demo

3.4 Dynamisches Models

- Neue Parts an PartStack hängen
- Setzen der Default Perspektive

```
@SuppressWarnings("restriction")
public class OpenDefaultPerspectiveHandler {
    private final static Logger LOG = LoggerFactory.getLogger(OpenDefaultPerspectiveHandler.class);

    @Execute
    public void execute(final EModelService modelService, final MWindow window) {
        window.setLabel(Messages.Default_Perspective_Window_Title);
        final MPerspectiveStack pstack = (MPerspectiveStack) modelService.find(ElementIdConstants.PERSPECTIVESTACK_ID, window);
        final MPerspective perspective = (MPerspective) modelService.find(ElementIdConstants.DEFAULT_PERSPECTIVE_ID, window);
        pstack.setSelectedElement(perspective);
    }
}
```

4.1 Dependency Injection Standard

Standard Annotations and Classes

@Inject (javax.inject)

@Named (javax.inject)

@Singleton (javax.inject)

Provider<T> (javax.inject)

@PostConstruct, @PreDestroy (javax.annotation)



4.2 Dependency Injection E4AP

E4AP-specific Annotations

@Optional (org.eclipse.e4.core.di.annotations)

@Active (org.eclipse.e4.core.contexts)

@Preference (org.eclipse.e4.core.di.extensions)

@Creatable (org.eclipse.e4.core.di.annotations)

@CanExecute, @Execute (org.eclipse.e4.core.di.annotations)

@Focus (org.eclipse.e4.ui.di)

@GroupUpdates (org.eclipse.e4.core.di.annotations)

@EventTopic (org.eclipse.e4.core.di.extensions), @UIEventTopic
(org.eclipse.e4.ui.di)



4.3 Dependency Injection Scope

- Constructor

```
@Inject
public Ereignisse(final Composite injectedParent, final IStylingEngine stylingEngine, final IResourcePool resourcePool,
    final UiEreignisseModel model, final IStammdatenService stammdatenService) {
```

- Method Optional

```
@Inject
@Optional
private void setInput(@UIEventTopic(EventConstants.EREIGNIS_LOADING) final Boolean loading) {
```

- Method Optional Parameter

```
@SuppressWarnings("rawtypes")
@Inject
private void ereignisRefresh(@Optional @UIEventTopic(EventConstants.EREIGNIS_REFRESH) final Map emptyMap) {
```

- Field

```
@Inject
private EMenuService menuService;

@Inject
IAccessRightService accesRightService;
```



4.4 Dependency Injection bei eigenen Klassen

- Erweiterungen über **IEclipseContext**
- Erstellen des Objekts über **ContextInjectionFactory.make**

```
private void createComposite(final Composite parent, final Class<? extends Composite> componentClass) {  
    final IEclipseContext subcontext = context.createChild();  
    subcontext.set(Composite.class, parent);  
    ContextInjectionFactory.make(componentClass, subcontext);  
}
```

- **@Creatable**

```
@Creatable  
class Todo {  
    @Inject  
    public Todo(Dependent depend, YourOSGiService service) {  
        // placeholder  
    }  
}  
  
@Creatable  
class Dependent {  
    public Dependent() {  
        // placeholder  
    }  
}
```

4.5 LifeCycleHandler

```
@SuppressWarnings({ "restriction" })
public class StartupLifeCycleHandler {

    private final static Logger LOG = LoggerFactory.getLogger(StartupLifeCycleHandler.class);

    @Inject
    IEnvironment envService;

    @Inject
    IStammdatenCacheService stammdatenLoader;

    @Inject
    ILoginService loginService;

    @Inject
    @Optional
    ILoginDialogService loginDialog;

    @Inject
    IMessageReader messageReader;

    @Inject
    IEventBroker eventBroker;

    @PostContextCreate
    public void startup() {
        LOG.debug("starting in life cycle"); //$NON-NLS-1$
    }
}
```

5.1 OSGi Services und Model Services

- Struktur



- Manifest

```

org.eclipse.rcp.application.rcp.login.ui.logindialog.xml
Service-Component: OSGI-INF/logindialog.xml
Bundle-ClassPath: .

```

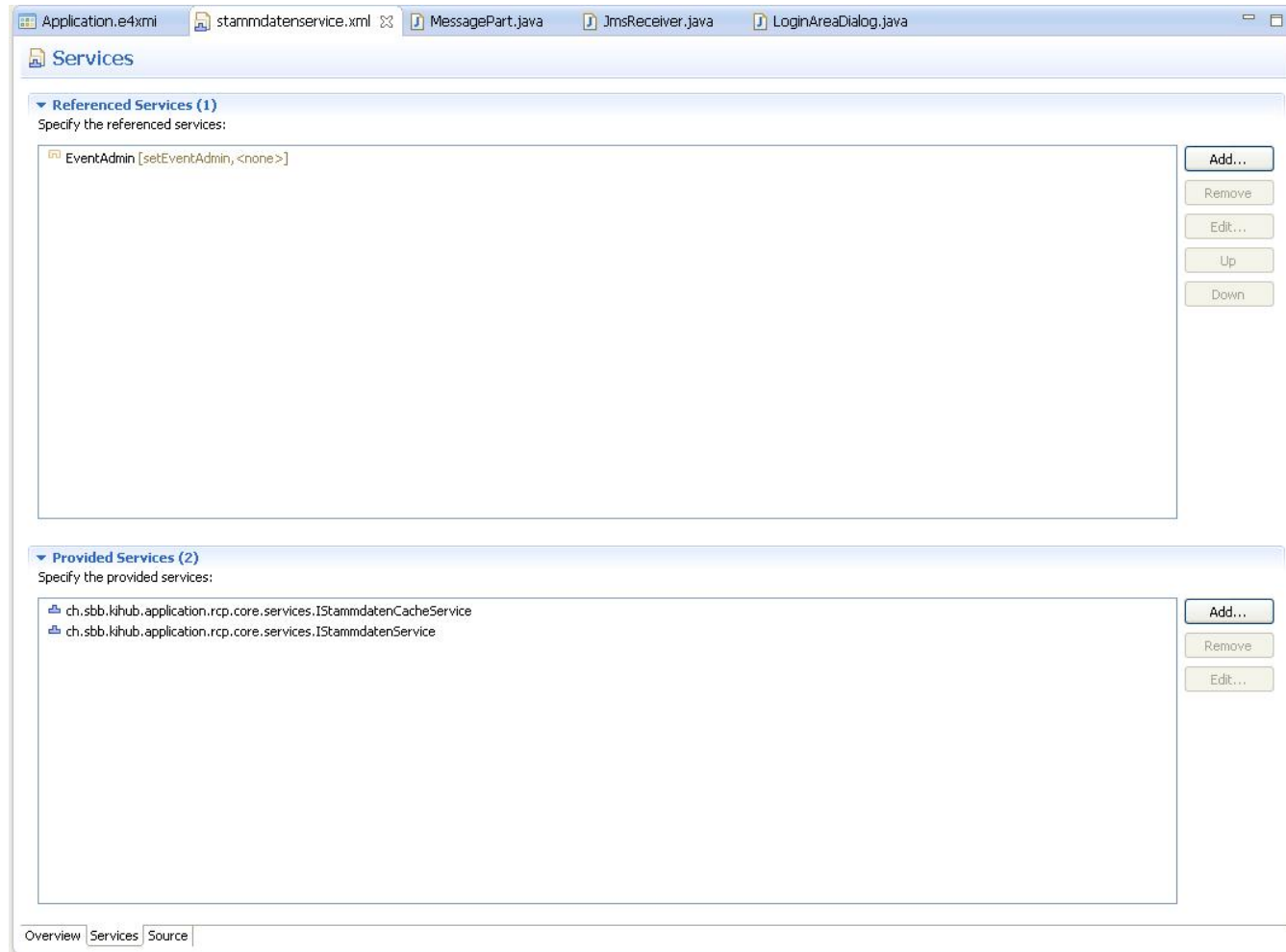
- Service-Definition

```

<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" immediate="true" name="ch.sbb.kihub.application.rcp.login.ui.logindialog">
  <implementation class="ch.sbb.kihub.application.rcp.login.ui.service.LoginDialogService"/>
  <service>
    <provide interface="ch.sbb.kihub.application.rcp.login.service.ILoginDialogService"/>
  </service>
</scr:component>

```

5.2 OSGi Services und Model Services



5.3 OSGi Services und Model Services

EModelService	Find elements in the model, create or clone snippets and insert new elements model structures
ESelectionService	Retrieve and set the current workbench selection.
ECommandService	Access, create, change and trigger commands.
EHandlerService	Access, change and trigger handlers.
EPartService	Provides API to access parts, e.g. find parts by their ID. It allows you to switch Perspectives and to hide and show Parts.
IEventBroker	Provides functionality to send event data and to register for certain events.
StatusReporter	Allows you to report Status objects, concise interface compared to standard IStatus reporting.
EContextService	Activate and deactivate keybindings defined as BindingContext in the application model.
IThemeEngine	Define themes and switch the styling at runtime.



5.4 OSGi Services und Model Services

```
@Inject  
private IEventBroker eventBroker;
```

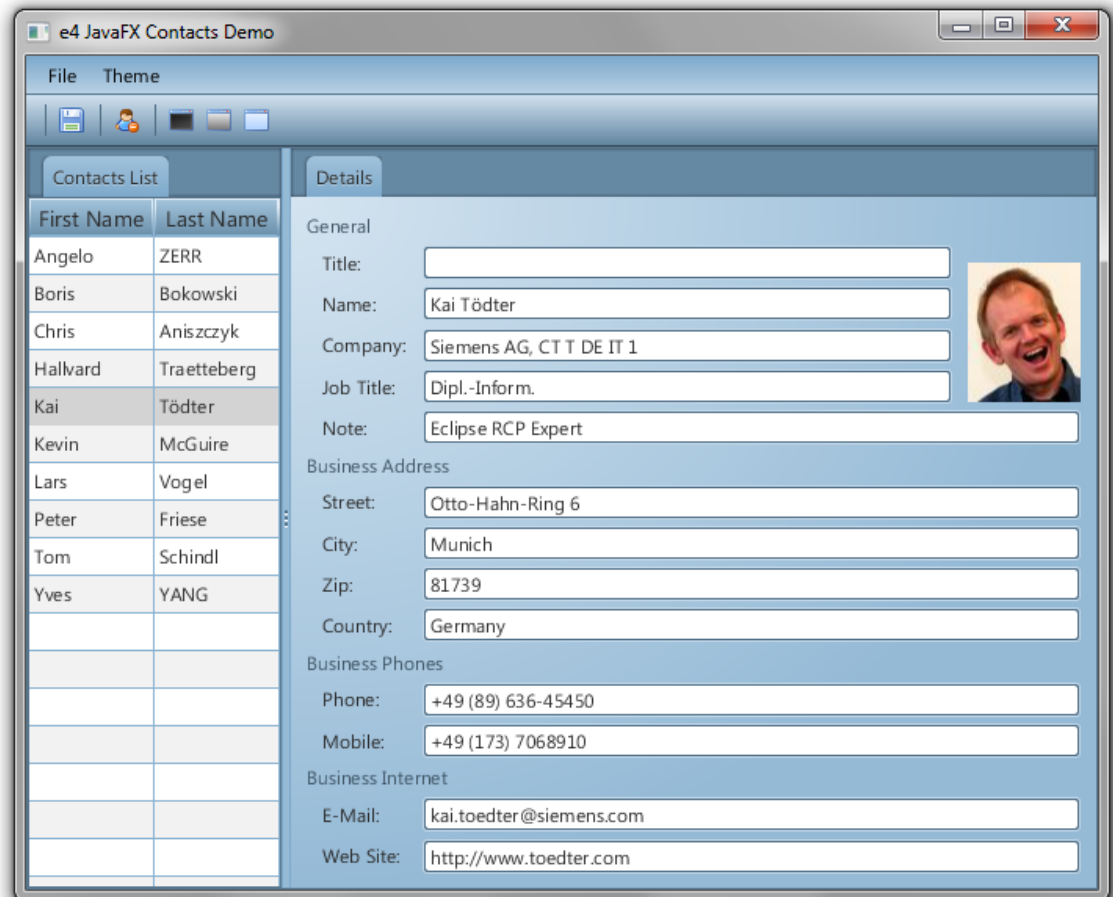
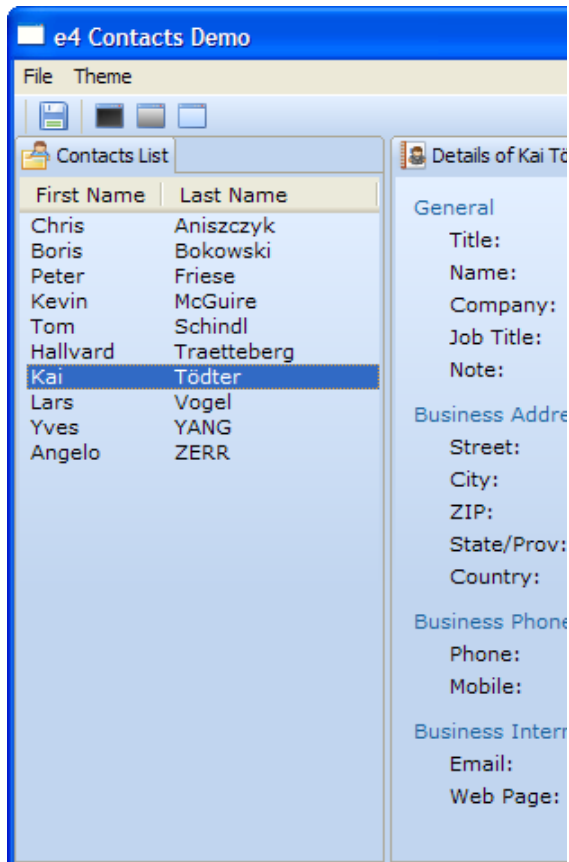
The IEventBroker service collects all events and sends them to the registered classes. This can be done asynchronously or synchronously.

```
// Asynchronous delivery  
// Caller will not be blocked until delivery  
IEventBroker.post(String topic, Object data)
```

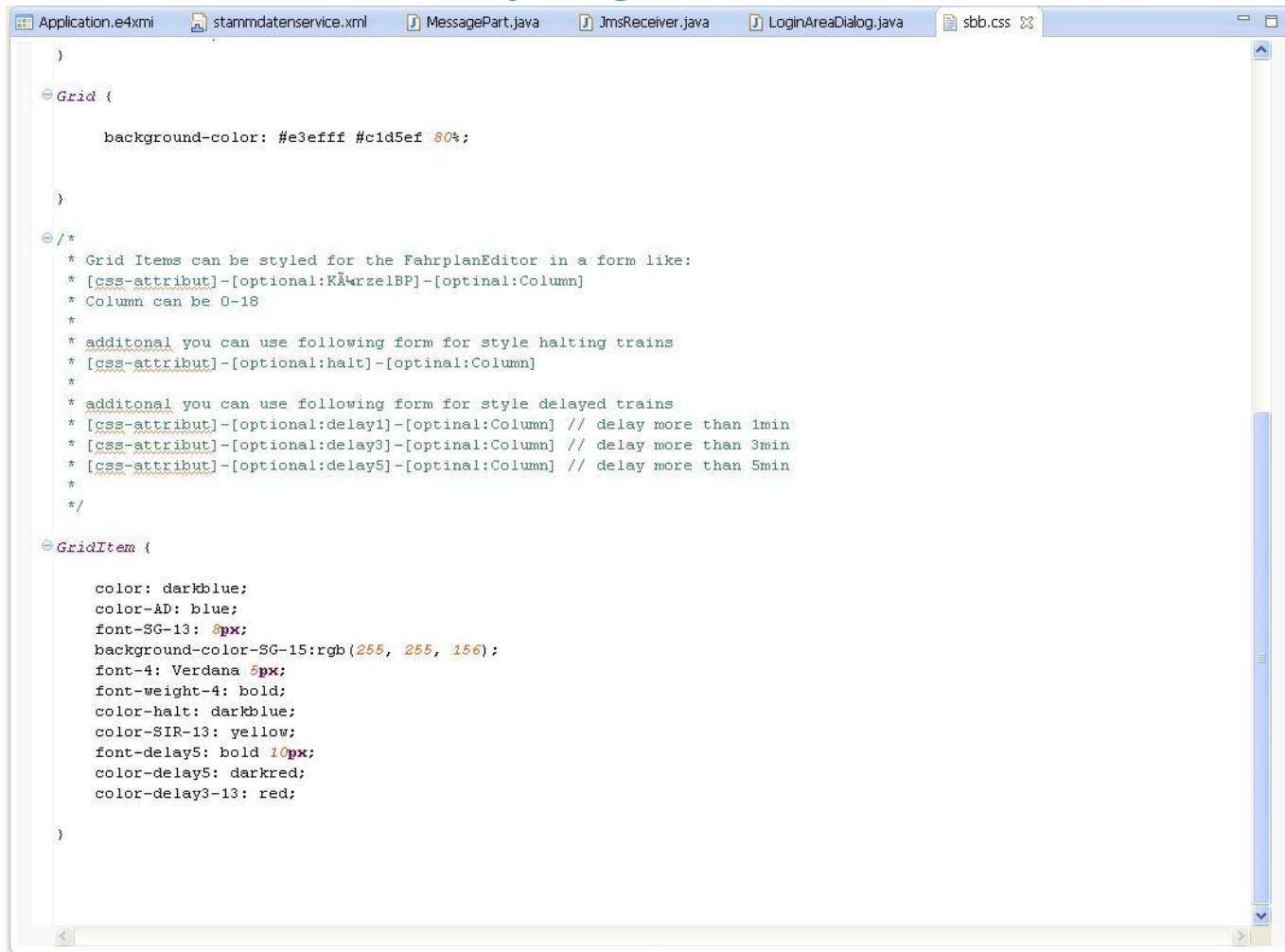
```
// Synchronous Delivery  
// Caller will block until delivery  
IEventBroker.send(String topic, Object data)
```

6.1 Renderer und CSS-Styling

- e4 mit e(fx)clipse



6.2 Renderer und CSS-Styling



```

Application.e4xmi  stammdatenservice.xml  MessagePart.java  JmsReceiver.java  LoginAreaDialog.java  sbb.css
}
Grid {
    background-color: #e3efff #c1d5ef 80%;
}

/*
 * Grid Items can be styled for the FahrplanEditor in a form like:
 * [css-attribut]-[optional:KürzelBP]-[optional:Column]
 * Column can be 0-18
 *
 * additional you can use following form for style halting trains
 * [css-attribut]-[optional:halt]-[optional:Column]
 *
 * additional you can use following form for style delayed trains
 * [css-attribut]-[optional:delay1]-[optional:Column] // delay more than 1min
 * [css-attribut]-[optional:delay3]-[optional:Column] // delay more than 3min
 * [css-attribut]-[optional:delay5]-[optional:Column] // delay more than 5min
 */

GridItem {
    color: darkblue;
    color-AD: blue;
    font-SG-13: 8px;
    background-color-SG-15:rgb(255, 255, 156);
    font-4: Verdana 8px;
    font-weight-4: bold;
    color-halt: darkblue;
    color-SIR-13: yellow;
    font-delay5: bold 10px;
    color-delay5: darkred;
    color-delay3-13: red;
}
  
```


KIHUB und e4

- Start mit Risiken
- Workshop mit Tom Schindl
- Milestones
- Forum und Bug Reports

Fazit:

- Leichtgewichtig
- Stabil
- Services
- Dependency Injection
- Workarounds
- Monkey see, Monkey do



Das Risiko hat sich gelohnt

Fragen?

