Tom Schindl

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

Innsbruck, Austria

e4 - The platform of the future

JUGS
Thursday, January 14, 2010

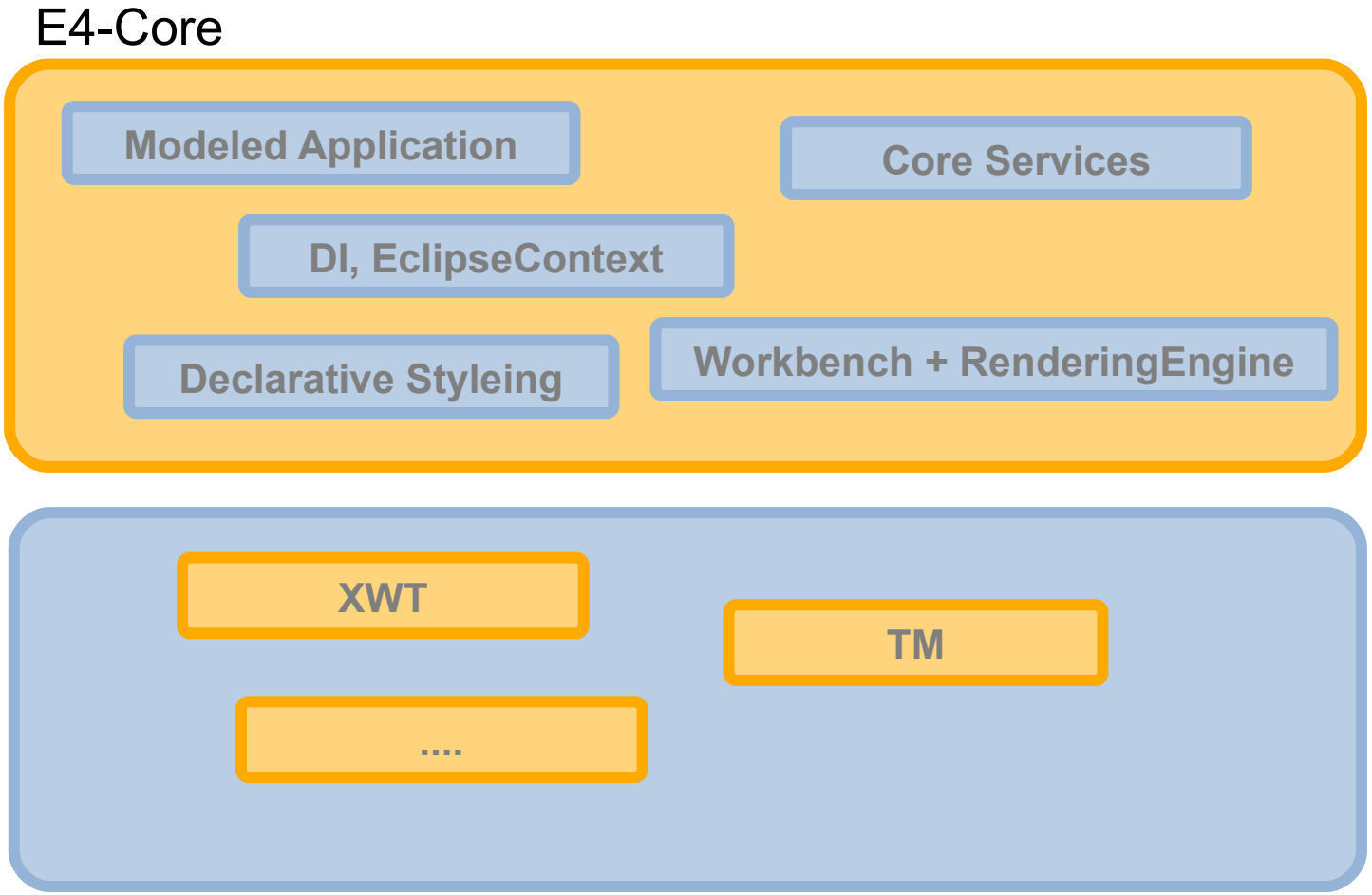
e4 – About Me

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

e4 – Presentation Topics

- History and Reasons
- Core - Programming Model
- Model - Modeled Workbench
- UI - Rendering and Styling

e4 – Presentation Topics

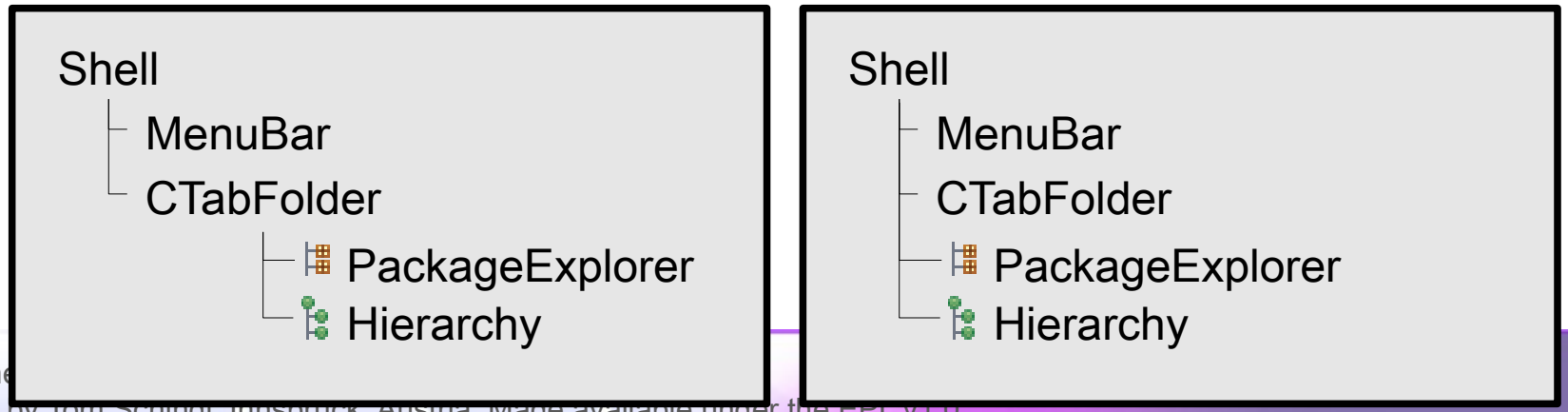


e4 – History and Reasons

- History and Reasons
- Core - Programming Model
- Model - Modeled Workbench
- UI - Rendering and Styling

e4 – History and Reasons

- **Current platform code is hard to maintain**
  - Different MVC implementations
  - Different Event Models
  - Legacy code because of (former) platform deficencies

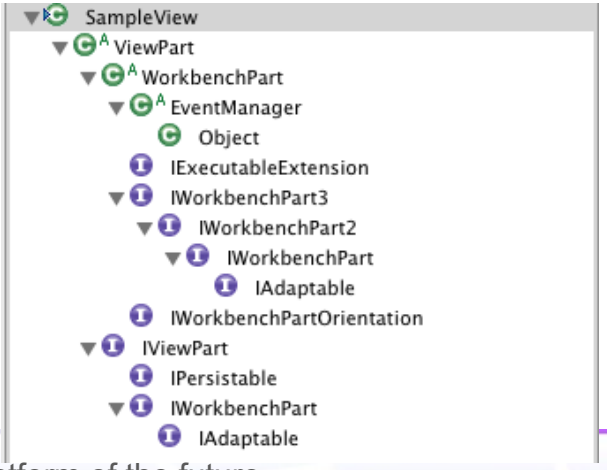


e4 – History and Reasons

- **Use of outdated patterns**
  - Singletons and statics all over the place

```
PlatformUI.getWorkbench()
```

  - Too much usage of inheritance



e4 – History and Reasons

- **New competitors**
  - RIA frameworks like Flex, Silverlight and JavaFX
  - GWT, Ajax-Frameworks (Ext-Js, …)
- **New UI Philosophies**
  - Shift away from native looking UIs

e4 – Programming Model

- History and Reasons
- **Core - Programming Model**
- Model - Modeled Workbench
- UI - Rendering and Styling

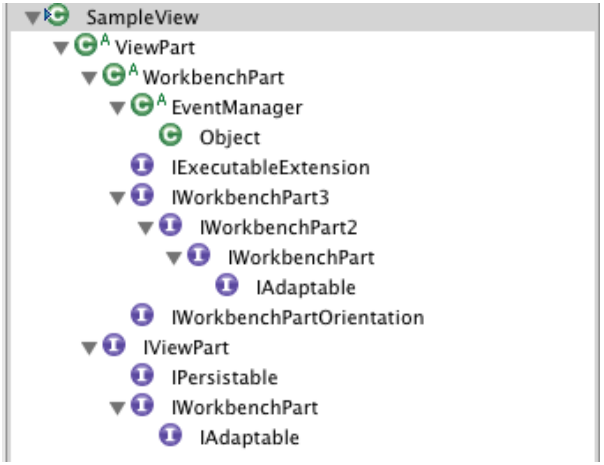
e4 – Programming Model

- **EclipseCon 08**
  - Mock up model based upon HashMaps
  - Mock hosted „hacked“ into 3.x
- **E4-Summit Ottawa (22nd /23rd May)**
  - May 20th: Mail to e4-dev „A radical approach to explore new paths for e4“
    - Platform designed from Scratch
    - No statics, no singletons, usage of DI

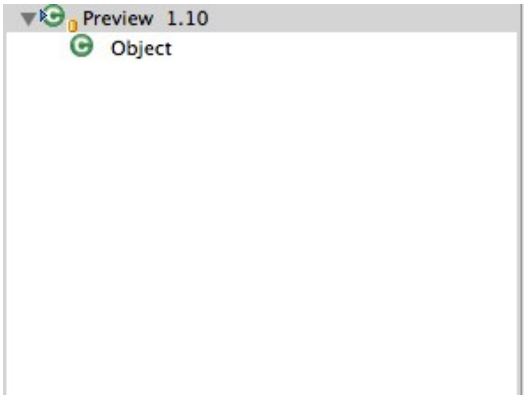
e4 – Programming Model

- **Usage of Dependency Injection to**
  - Flatten the inheritance

Eclipse 3.x



E4



All e4 building blocks are POJOs

e4 – Programming Model

- **Usage of Dependency Injection to**
  - Remove the need to reach out to statics to get access to workbench services

Eclipse 3.x

```
public class SampleView extends ViewPart {
  public void createPartControl(Composite parent) {
    ISelectionService service =
                  PlatformUI.getWorkbench().getActiveWorkbenchWindow().getSelectionService();
    service.addSelectionListener(new ISelectionListener() {
      public void selectionChanged(IWorkbenchPart part, ISelection selection) {
        IStructuredSelection s = (IStructuredSelection)selection;
        setSelection((Contact)s.getFirstElement());
      }
    });
  }
  private void setSelection(Contact contact) {
    // Handle changed selection
  }
  private void execCommand() {
    ICommandService cmdService =
      (ICommandService)PlatformUI.getWorkbench().getService(ICommandService.class);
    // Execute command
  }
}
```

e4 – Programming Model

- **Usage of Dependency Injection to**
  - Remove the need to reach out to statics to get access to workbench services

e4

```
public class SampleView {
  @Inject
  private ECommandService cmdService;

  @Inject
  public SampleView(Composite parent) {

  }

  @Inject
  private void setSelection(@Optional @Named(IServiceConstants.SELECTION) Contact contact) {
    // Handle changed selection
  }

  private void execCommand() {
    // Execute command
  }
}
```

e4 – Programming Model

- **e4 Dependency Injection Features**
  - JSR 330
  - Additionally @PostContruct, @PreDestroy

```
public class ViewA {
  @Inject
  private Composite parent;

  @PostConstruct
  void init() {
    // Create UI
  }

  @PreDestroy
  void preDestroy() {
    // Release e.g. Image-Resources, ...
  }
}
```

e4 – Accessing services

- **e4 Dependency Injection Features**
  - Easy access to OSGi-Services e.g. contributed through DS

DetailPart.java | selectionprovider.xm | ufkpart.ecore | Application.xmi | Application.xmi | Location.java | Librar

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="org.eclipse.ufacekit.ui.core.model.example">
3    <implementation class="org.eclipse.ufacekit.ui.core.model.example.AddressSelectionBroker"/>
4    <service>
5       <provide interface="org.eclipse.ufacekit.ui.core.model.example.IAddressSelectionBroker"/>
6    </service>
7 </scr:component>
8
```

```
public class ViewA {
  @Inject
  private IAddressSelectionBroker parent; // OSGi-Service registered through DS

}
```

e4 – Programming Model

- **One consitent Event-System**
  - Reuse the OSGi-Eventsystem

```
public class ViewA {
  @Inject
  private IEventBroker eventBroker;
  @Inject
  private Composite parent;

  public void deliverAsyncEvent() {
    eventBroker.post("myEvent", selected);
  }
}
```

```
public class ViewB implements EventHandler {
  @Inject
  public ViewB(Composite parent, IEventBroker eventBroker) {
    eventBroker.subscribe("myEvent", this);
  }

  public void handleEvent(Event event) {
    // Handle the event
  }
}
```

e4 – Programming Model

- **Well defined set of services**
  - Core
    - Life Cycle
    - Authentication/Single Sign-on
    - Status Handling
    - Logging and Tracing
    - Extension Registry
    - Scheduling Work/Reporting Progress
    - Reading/Writing of Preferences
    - String Localization
    - Adapting Objects
    - Commands/Handlers
    - Eventing System
    - Participating in Undo/Redo

e4 – Programming Model

- **Well defined set of services**
  - UI
    - Receiving Input
    - Providing Selection Information
    - Persisting State and Data
    - Managing Shared Resources
    - Participating in Editor/Saveable Part Life Cycle
    - Updating UI Elements
    - Notifications
    - Status Reporting
    - Part Service

e4 – Programming Model

- **Well defined set of services**
  - Advanced UI
    - Shell Provider
    - Reacting to Workbench Model Changes.
    - Participating in Label and Icon Decoration
    - Reacting to Changes to the Context
    - Dynamically Contributing to the Workbench Model
    - Object Contributions
    - Focus Service

e4 – Modeled Workbench

- History and Reasons
- Core - Programming Model
- **Model - Modeled Workbench**
- UI - Rendering and Styling

e4 – Modeled Workbench

- **Central Point all application elements are registered in**
  - Holds the UI-Structure like Stacks, Perspectives, …
  - One stop shopping to get access to application elements for
    - Filtering
    - Sorting
    - …
- **Wires POJOs to an application**

e4 – Modeled Workbench

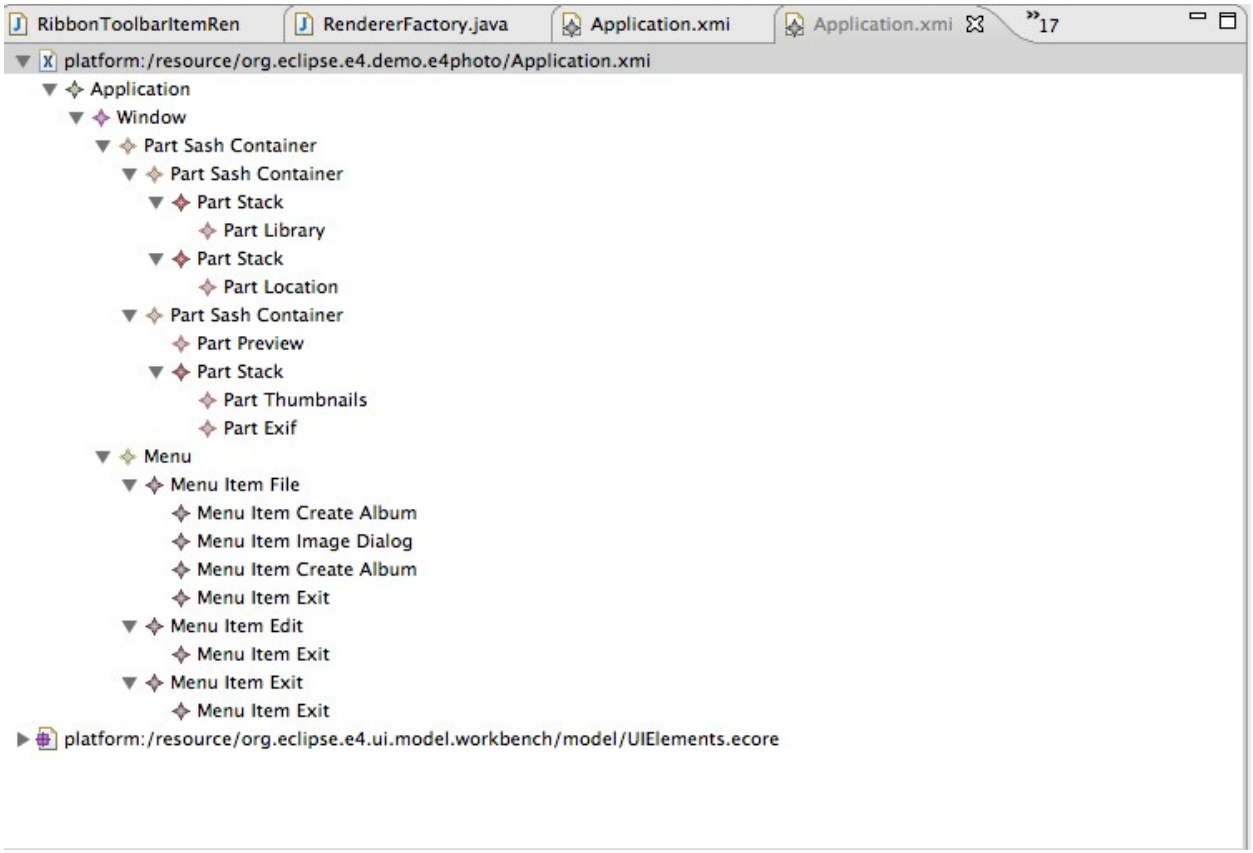
- **EMF? Why oh why?**
  - It's a proven domain model technology so why invent our own?
  - It has tooling (an Editor, …)
  - Integration points for different technologies like EMF-Compare, CDO, …

e4 – Modeled Workbench

- **EMF but isn't it bloat?**
  - Distinguish between installation and runtime bloat
  - Installation „bloat“ 1.5 MB
  - Runtime size of EMF is highly optimized (e.g. storage of booleans, …)
  - Benefit from upstream changes (Ultra Slim Diet in 3.5)

e4 – Modeled Workbench

- **Application model**



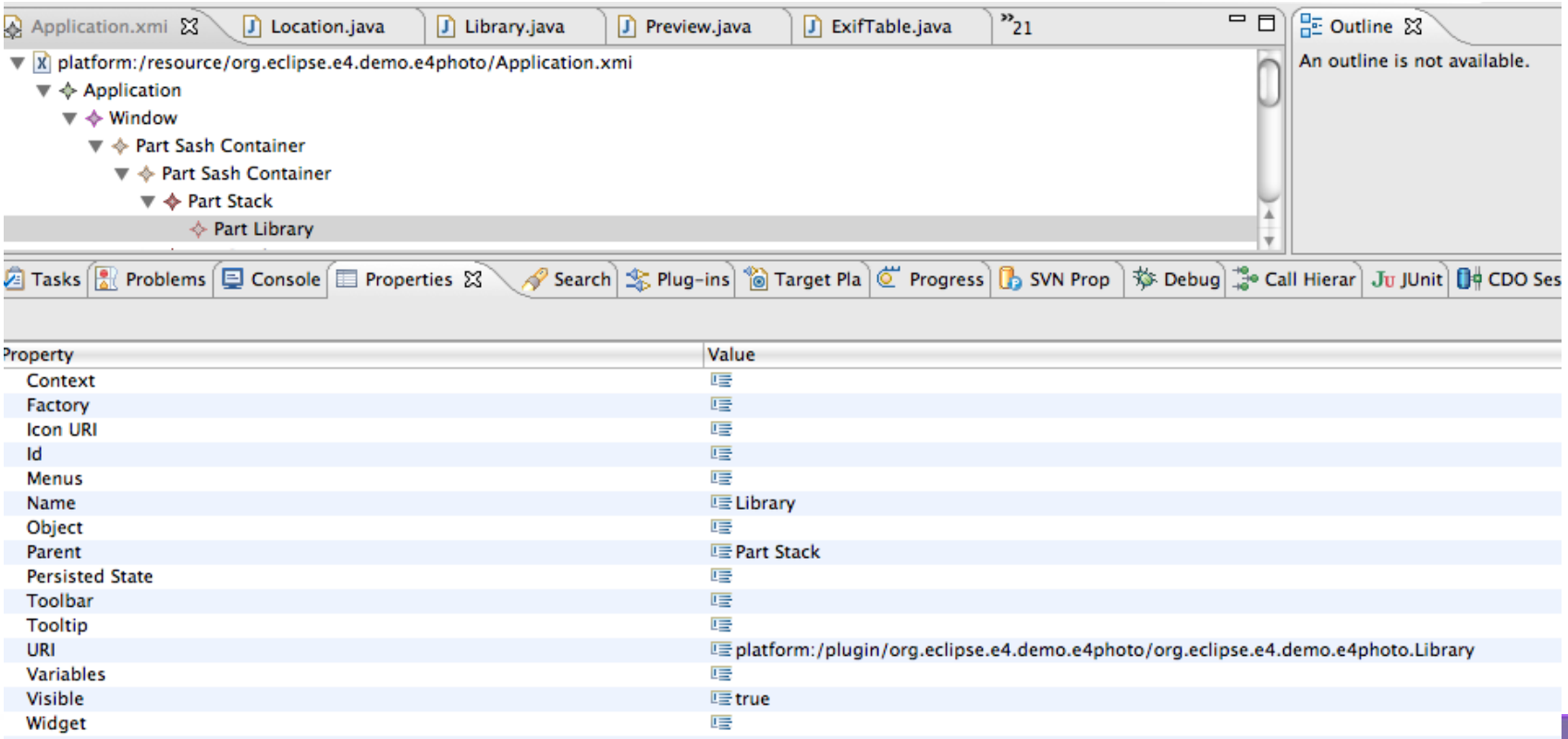
e4 – Modeled Workbench

- Part POJO

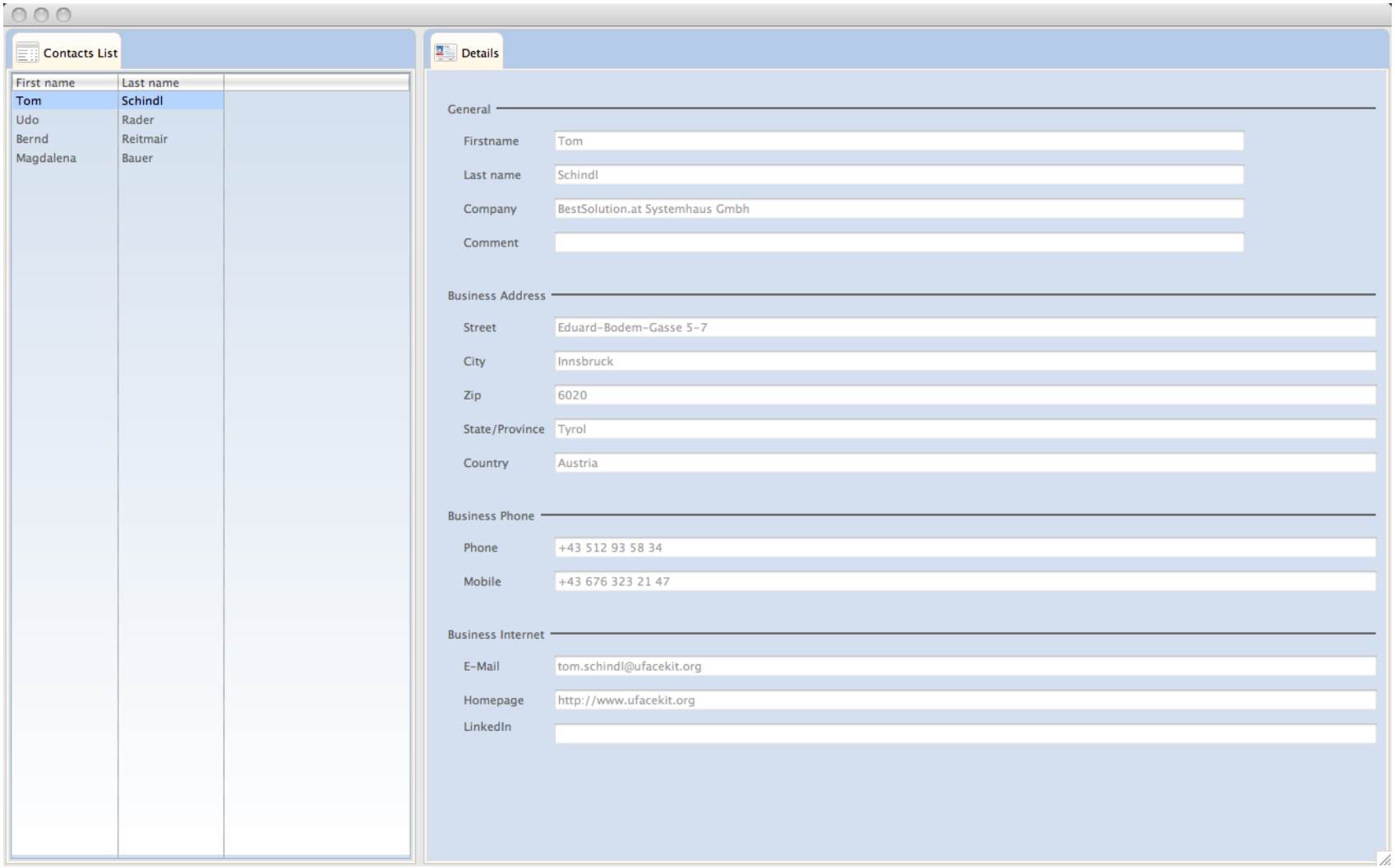
```
public class Library implements IDisposable {
  public Library(Composite parent, final IWorkspace workspace) {
      TreeViewer viewer = new TreeViewer(parent,
              SWT.MULTI | SWT.H_SCROLL | SWT.V_SCROLL);
  }
}
```

e4 – Modeled Workbench

- **Wiring the part POJO into the Application Model**

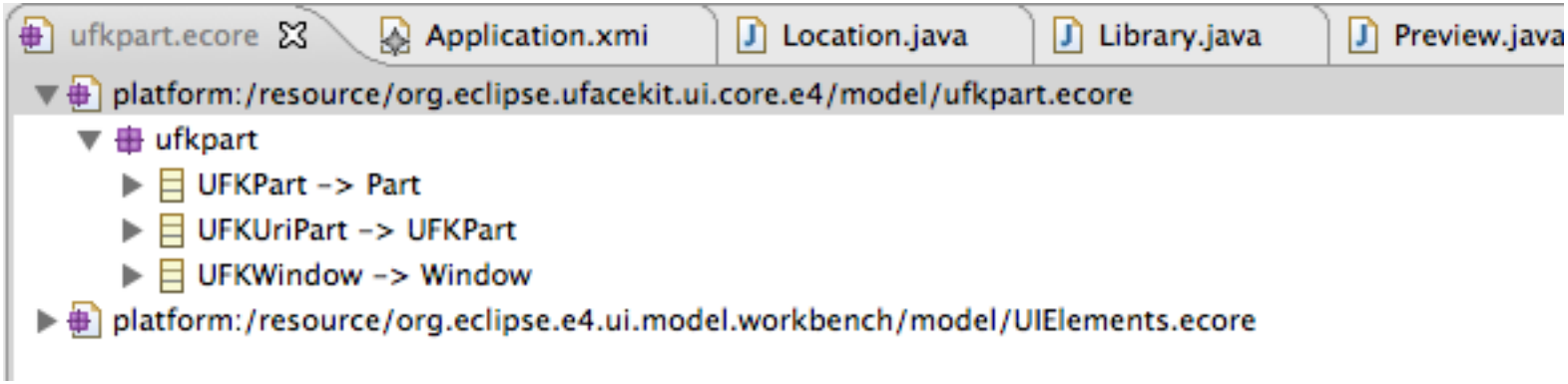


e4 – Modeled Workbench



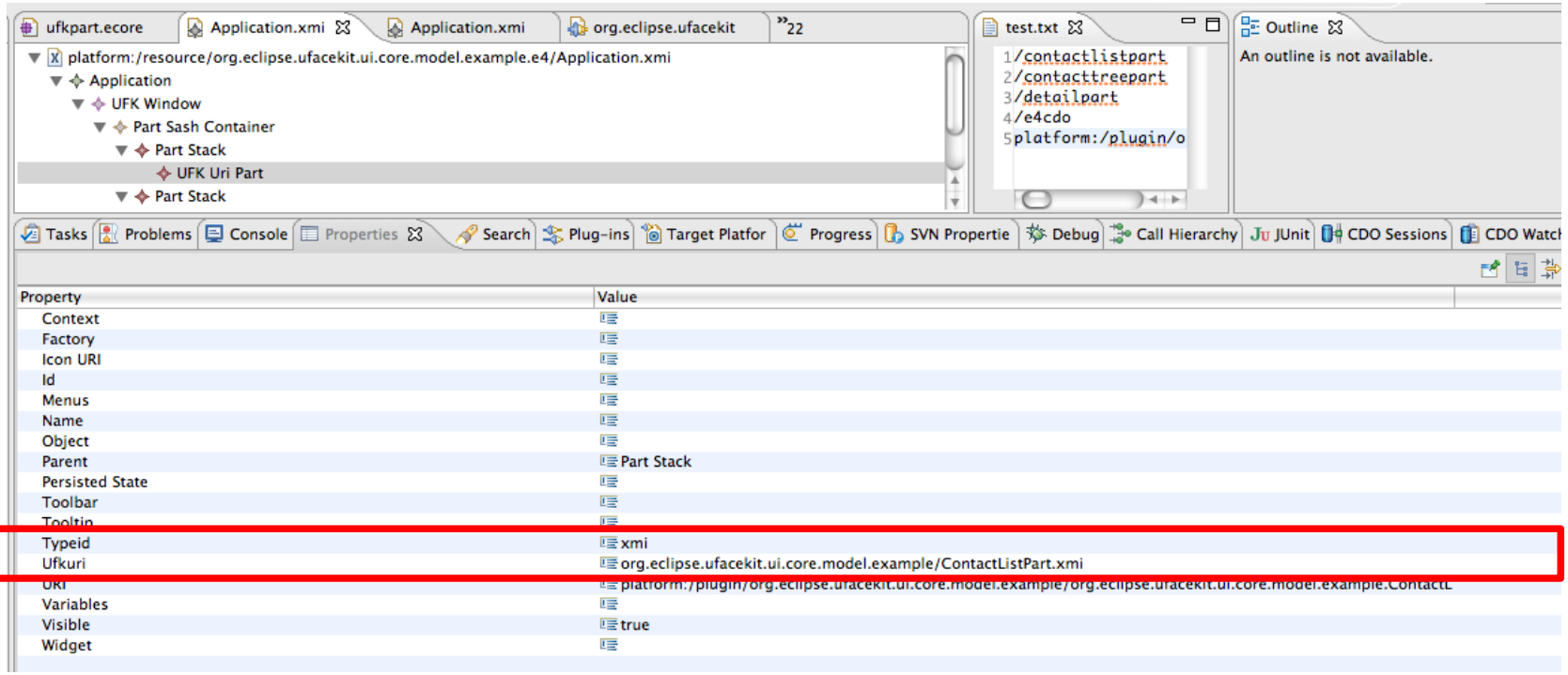
e4 – Modeled Workbench

- One can derive from the base .ecore and add features



e4 – Modeled Workbench

- **Extending the Application model**



e4 – Modeled Workbench

- Extending the Application model

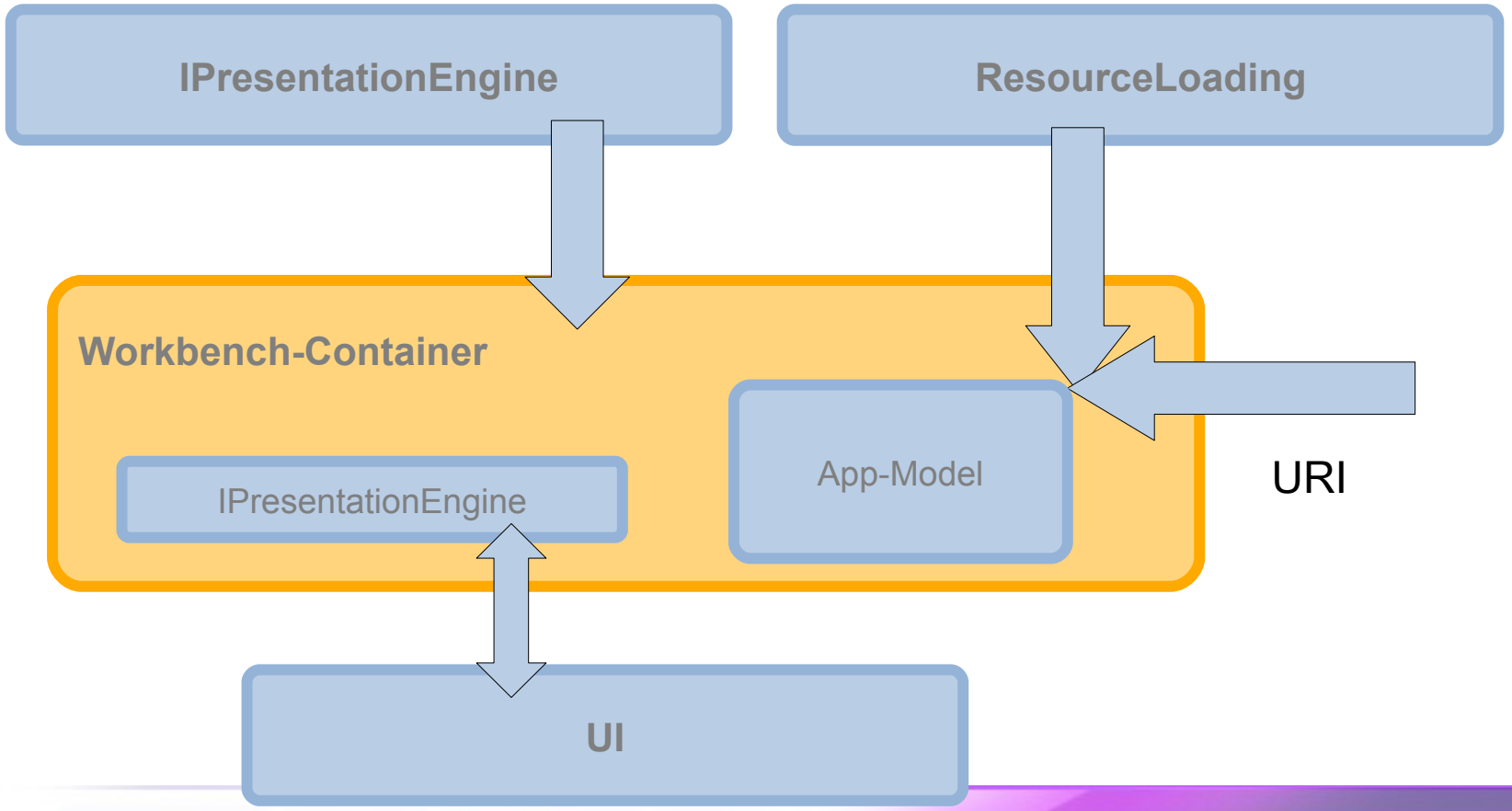
```
public class DetailPart {

  @Inject
  public DetailPart(UIComposite parent, Resource resource) {

  UIFactory<?> factory = parent.getFactory();
  UIDesktop desktop = parent.getDesktop();
  UFaceKitBuilder builder = new UfaceKitBuilder(
    factory,
    new DefaultBindingStrategy(desktop.getRealm(), Type.DOMAIN_TO_UI)
  );
  builder.buildPart(
    parent,
    (IUIComposite) resource.getContents().get(0)
  );

}
```

e4 – Rendering & Styling

- History and Reasons
- Core - Programming Model
- Model - Modeled Workbench
- **UI - Rendering and Styling**

e4 – Rendering & Styling

- Main concept of rendering



e4 – Rendering & Styling

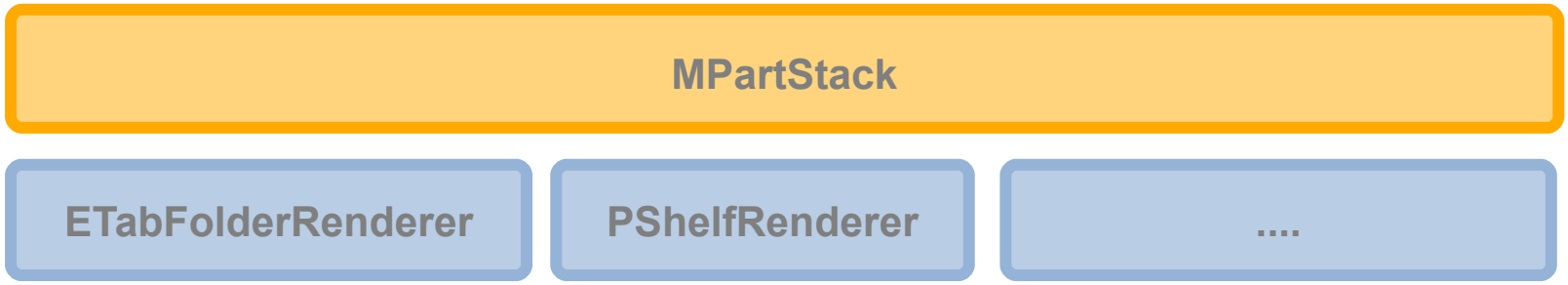
- **Concept of Renderers**
  - Make e4-application platform UI-Toolkit agnostic
  - Keep a clean seperate application model from ui
  - Allow full customization of the UI
  - Renderer the UI live from the application model

e4 – Rendering & Styling

- **Tasks of the renderer**
  - Manage Lifecycle of the UI-Element
    - Creation
    - Dipose
  - Synchronize attributes between both
    - Value changes
    - Structural changes

e4 – Rendering & Styling

- **Default Presentation Engine provided by e4**
  - Based on SWT
  - Extensible by plug in your own renderers
- **One Appmodel Element multiple renderers**



e4 – Rendering & Styling

```
public class RendererFactory extends WorkbenchRendererFactory {

  @Override
  public AbstractPartRenderer getRenderer(MUIElement uiElement,
    Object parent) {

    if (uiElement instanceof MPartStack && usePShelfRenderer() ) {

      if( stackRenderer == null ) {
        stackRenderer = new PShelfStackRenderer();
        initRenderer(stackRenderer);
      }

      return stackRenderer;
    }

    return super.getRenderer(uiElement, parent);
  }

}
```

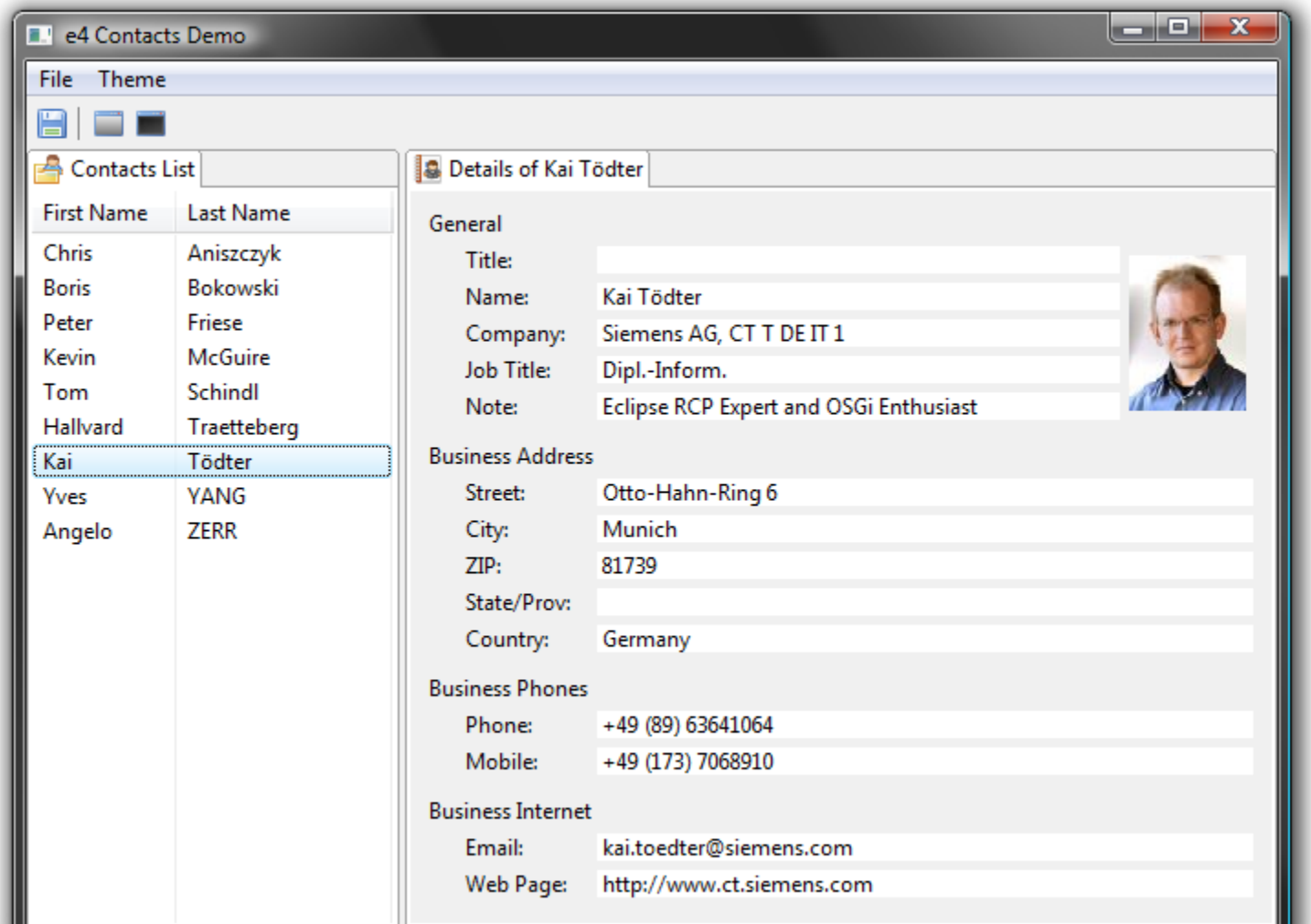
e4 – Rendering & Styling

- **Theming through Declartive Syntax**

```
Label {
   font: Verdana 8px;
   color: rgb(240, 240, 240);
}

Table {
   background-color: gradient radial #575757 #101010 100%;
   color: rgb(240, 240, 240);
   font: Verdana 8px;
}

ToolBar {
   background-color: #777777 #373737 #202020 50% 50%;
   color: white;
   font: Verdana 8px;
}
```

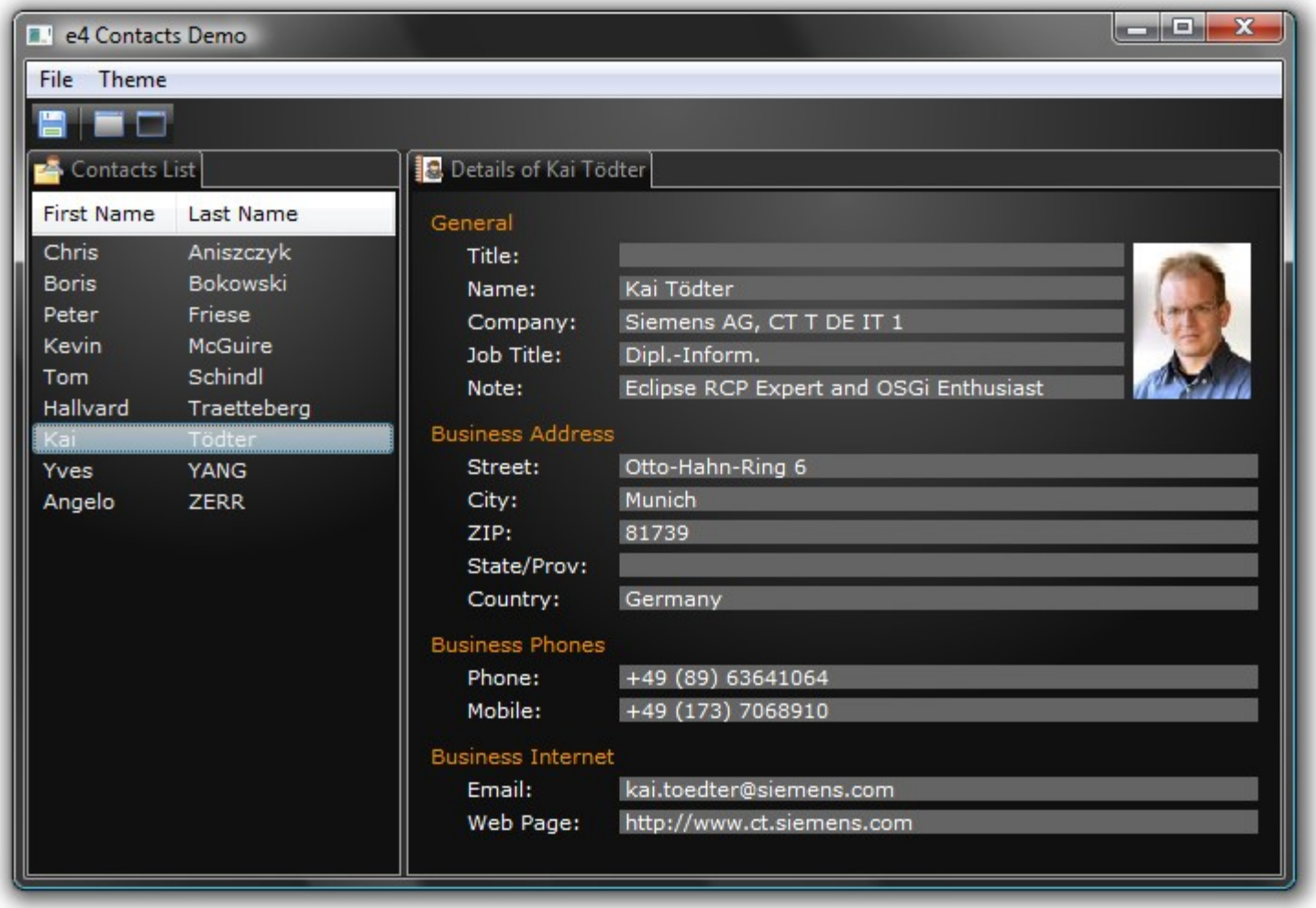
e4 – Rendering & Styling

- **Plain Application**



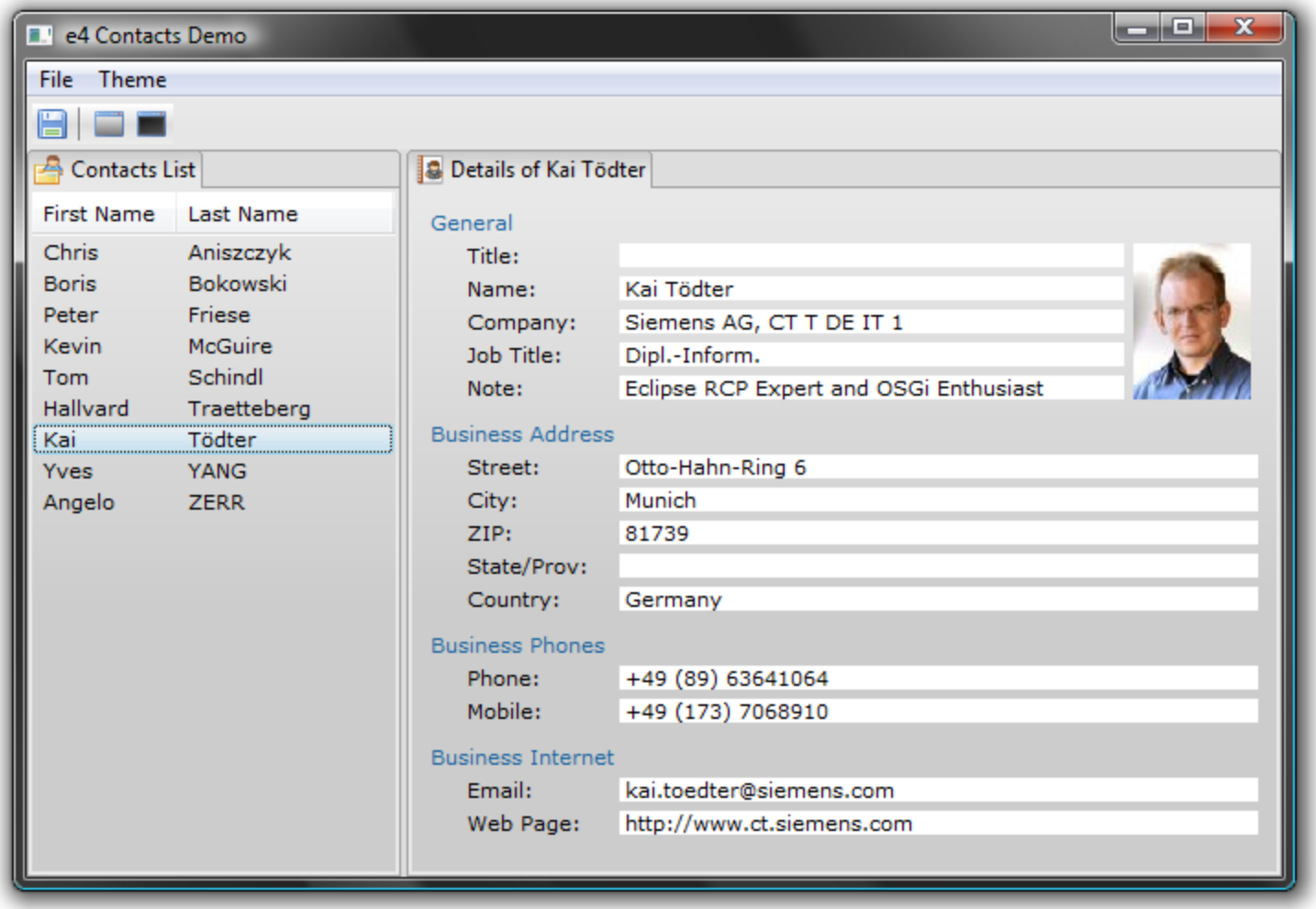
e4 – Rendering & Styling

- Dark.css



e4 – Rendering & Styling

- **Bright.css**

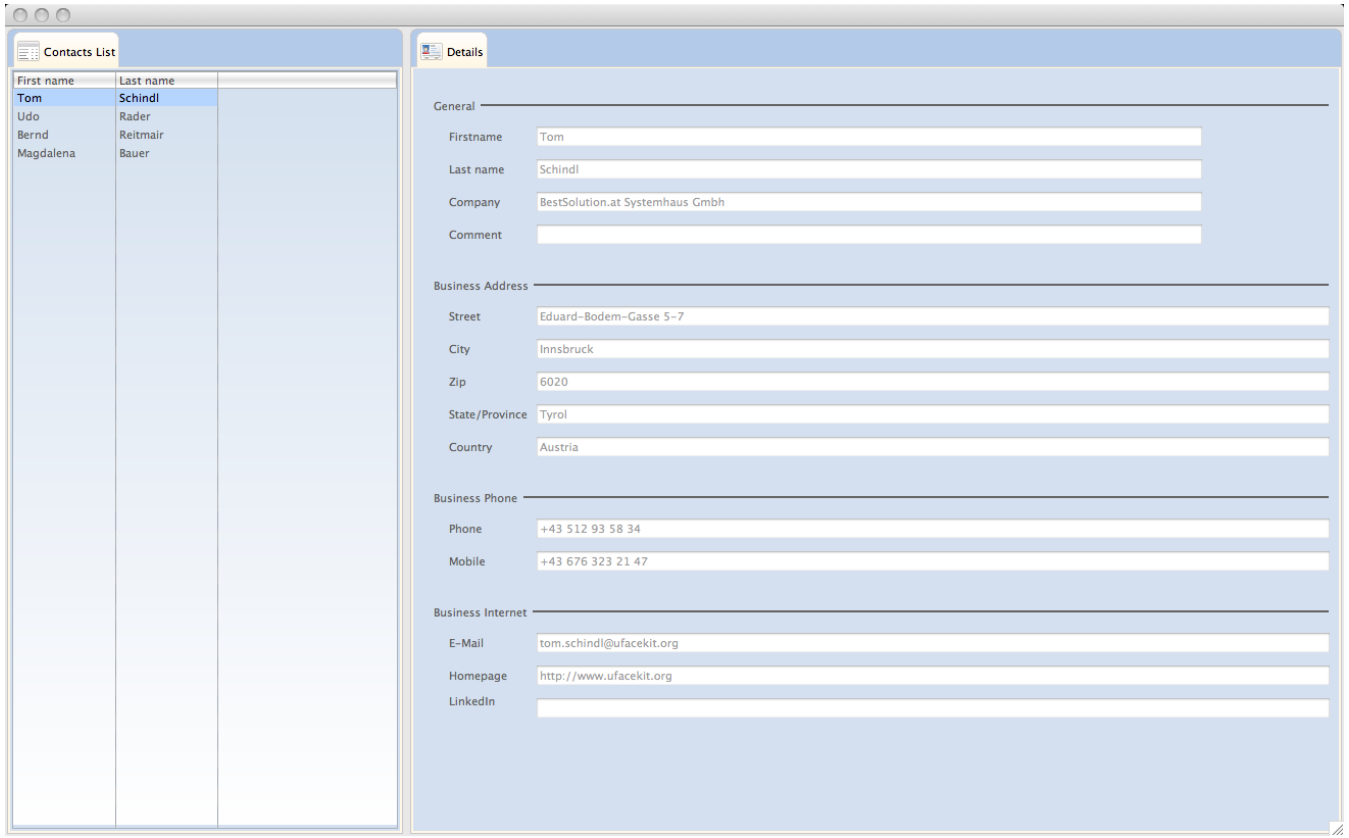


e4 – Rendering & Styling

- **Exchanging the Resource Loading**
  - E.g. for live application design of an E4-Application
  - Share model using CDO between JVMs

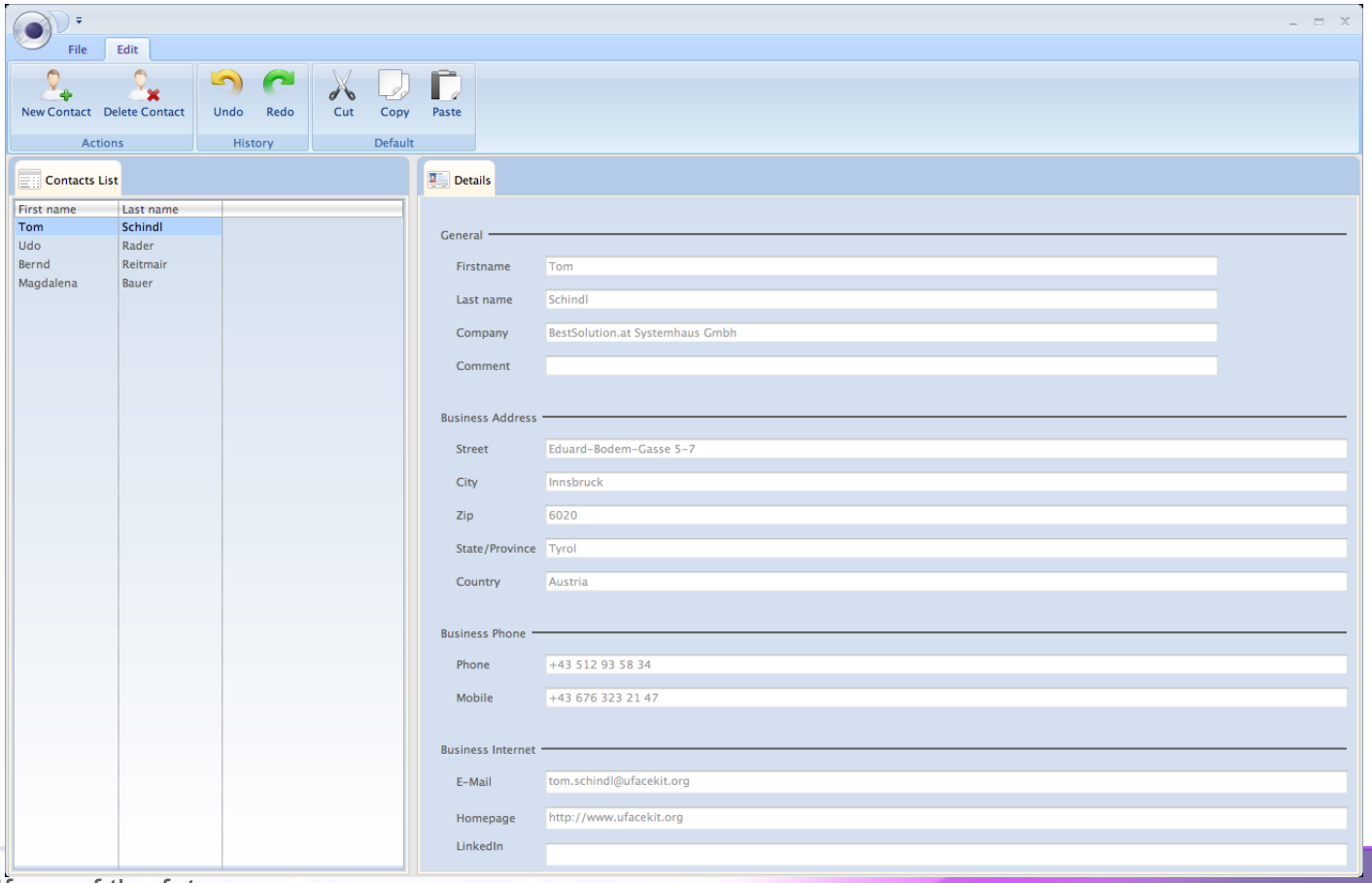
e4 – Rendering & Styling

- DEMO



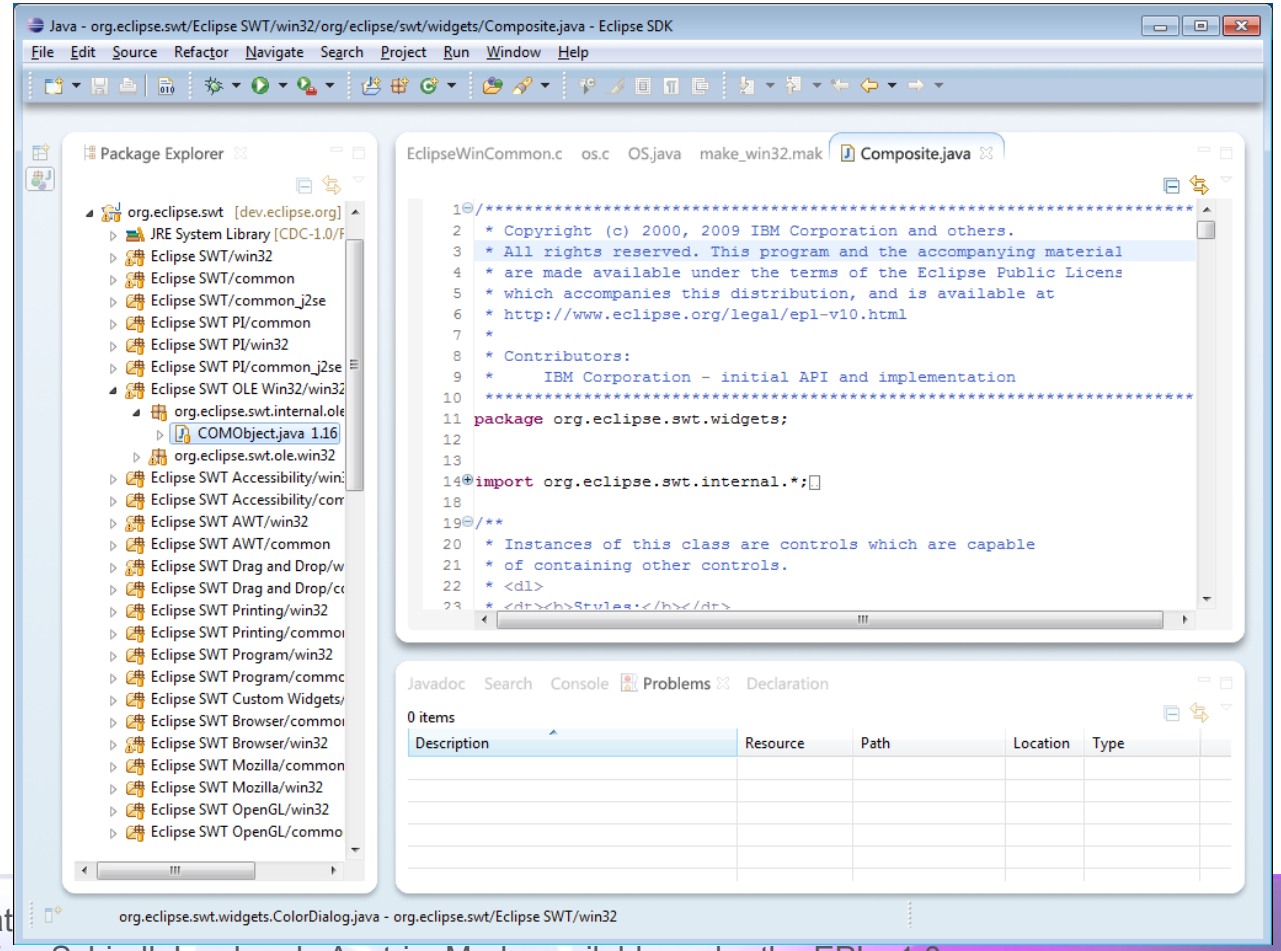
e4 – Rendering & Styling

- **Enhancing the renderers DEMO**



e4 – Rendering & Styling

- e4 - IDE mockups



e4 – Roundup

- **Main goals of e4**
  - Make it easy to develop UI components
    - Programming model with DI
    - Well defined Application-Services „The 20 things“
    - Declarative UI
  - Make it easy to create an application and/or adjust exiting ones to your needs
    - Modeled Workbench
    - PresentationEngine and Renderers
    - CSS

e4 – Roundup

- **Resources**
  - Official e4-Wiki-Page
    - http://wiki.eclipse.org/e4
  - Personal blog
    - http://tomsondev.bestsolution.at

THE END