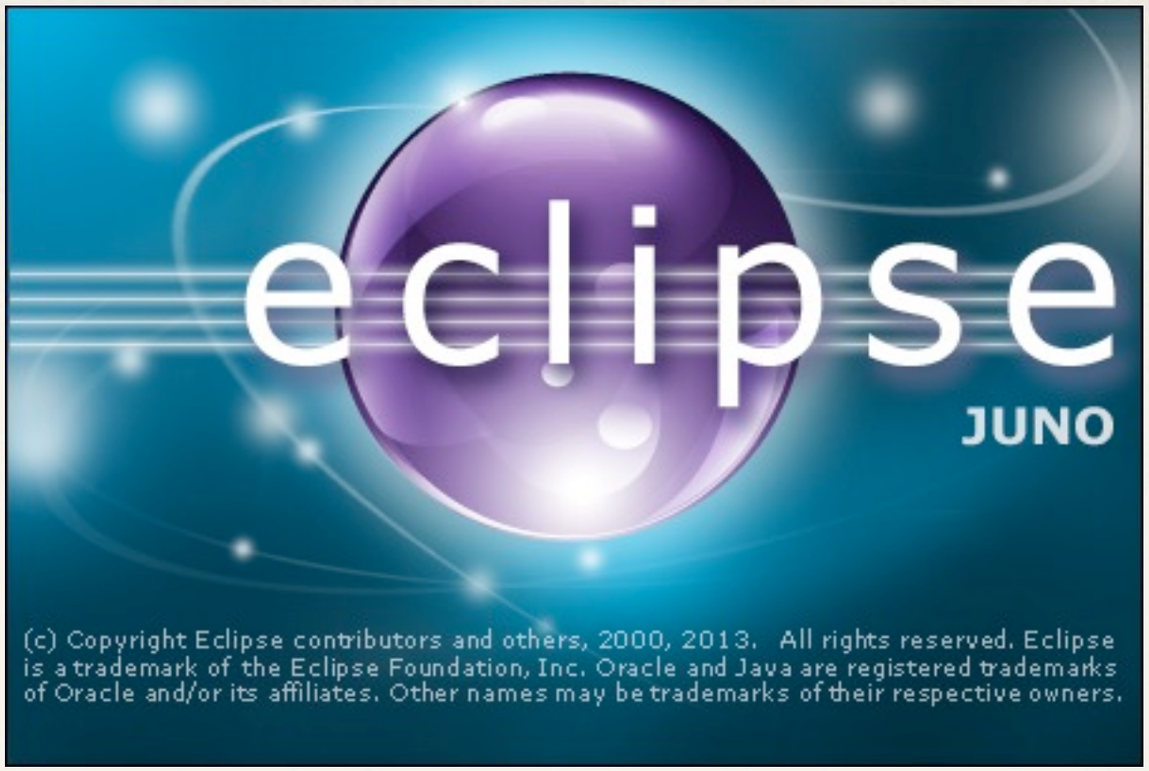


Advanced Eclipse 4 Application Platform

-

not for the weak-hearted

Tom Schindl - BestSolution Systemhaus GmbH
Sopot Çela - ICITAP

EclipseCon March 2013

About Tom Schindl

- CTO BestSolution Systemhaus GmbH
- Eclipse Committer
  - e4
  - Platform UI
  - EMF
- Twitter: @tomsontom

Required Software

- JDK7u7 / JDK8b81 or later
  - http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html
  - http://jdk8.java.net/download.html
- Eclipse Tutorial SDK
  - http://downloads.efxclipse.org/tutorial/sdks/
- Target Platform
  - http://downloads.efxclipse.org/tutorial/target-platform.zip

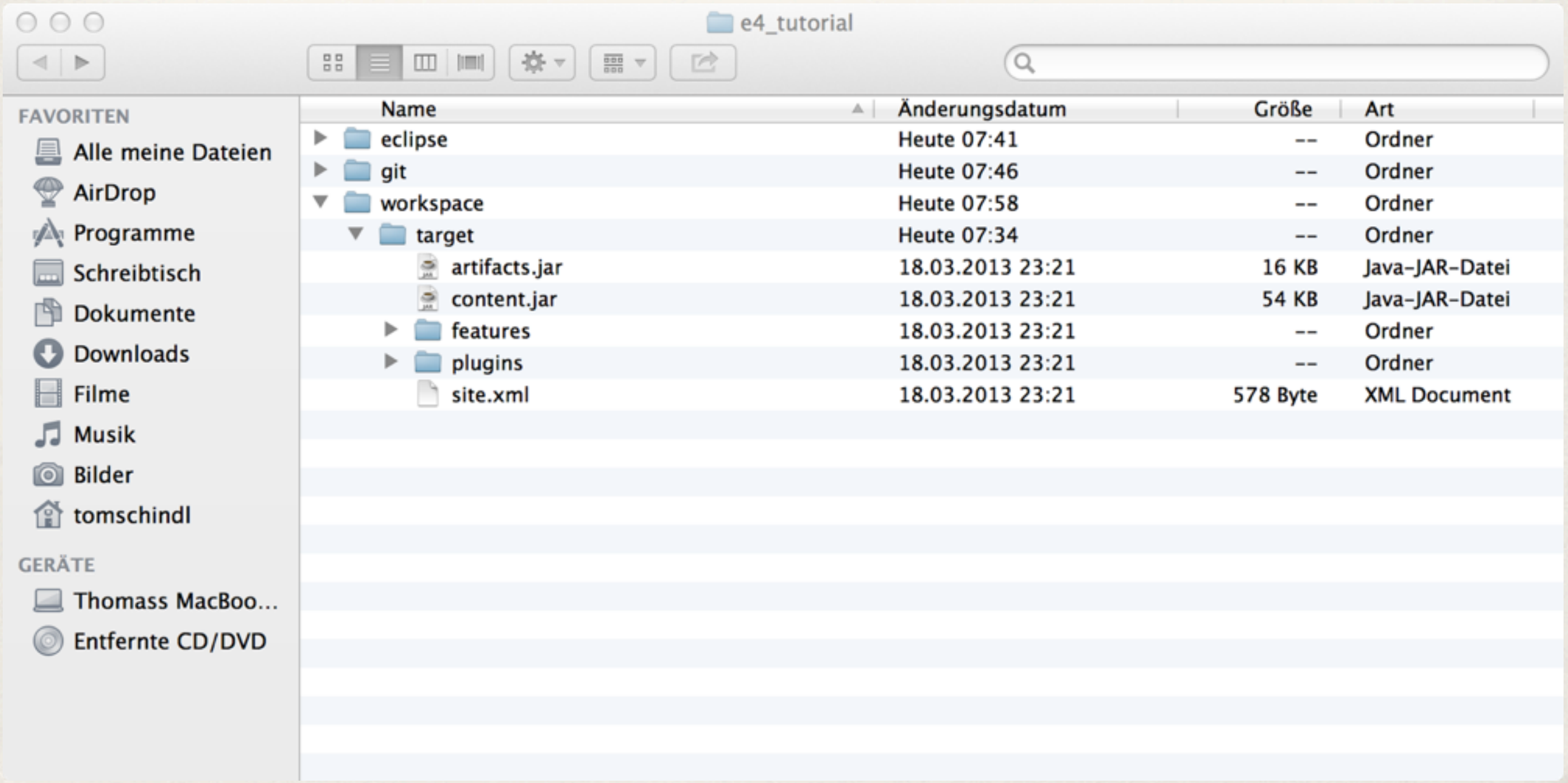
Setup Software

- Create a directory named **e4_tutorial**
- Create subdirectory **workspace**
- Create subdirectory **workspace/target**
- Copy **eclipse-SDK-4.2.2-....zip/tar.gz** to **e4_tutorial** & unpack it
- Copy **git.zip** to **e4_tutorial** and unzip it
- Copy **target-platform.zip** to **target** and unzip it

Setup Software

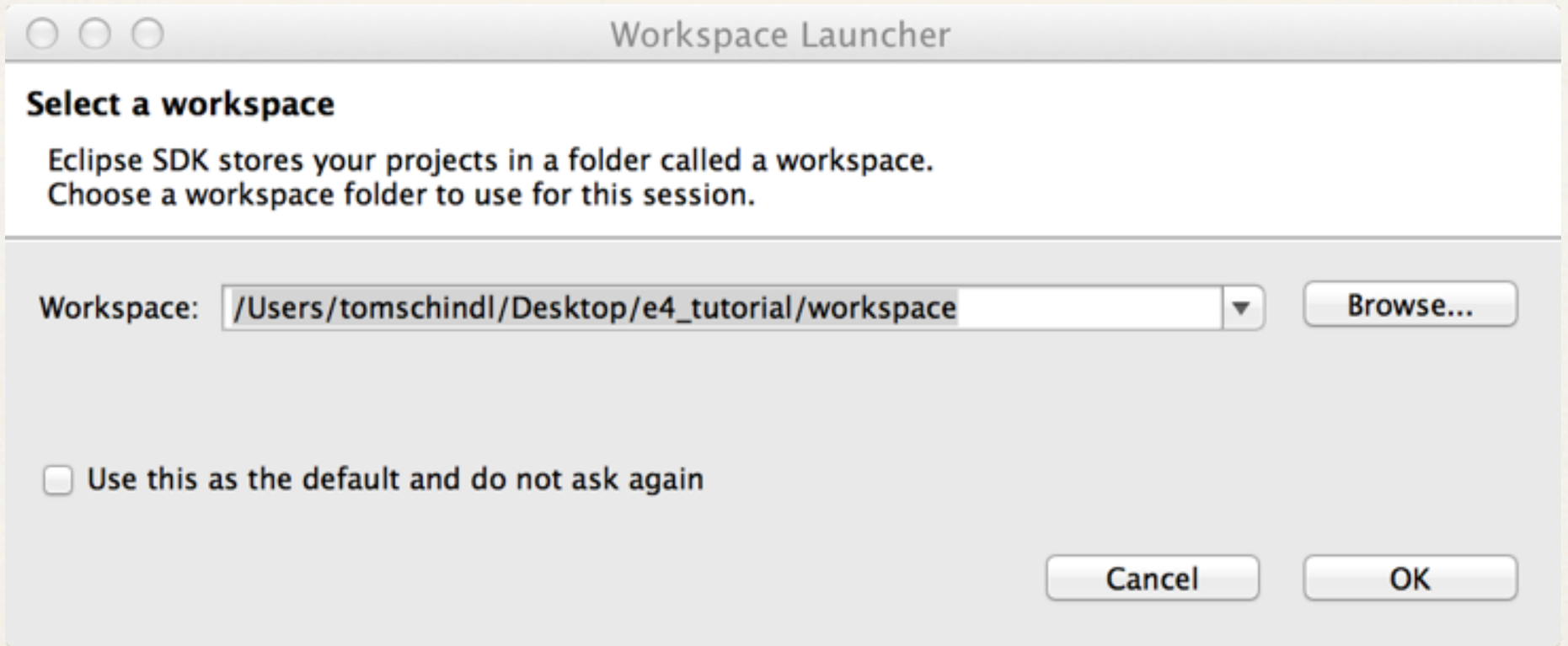
- Clone Repo from github
  git://github.com/tomsontom/eclipsecon-tutorial.git

Setup Software



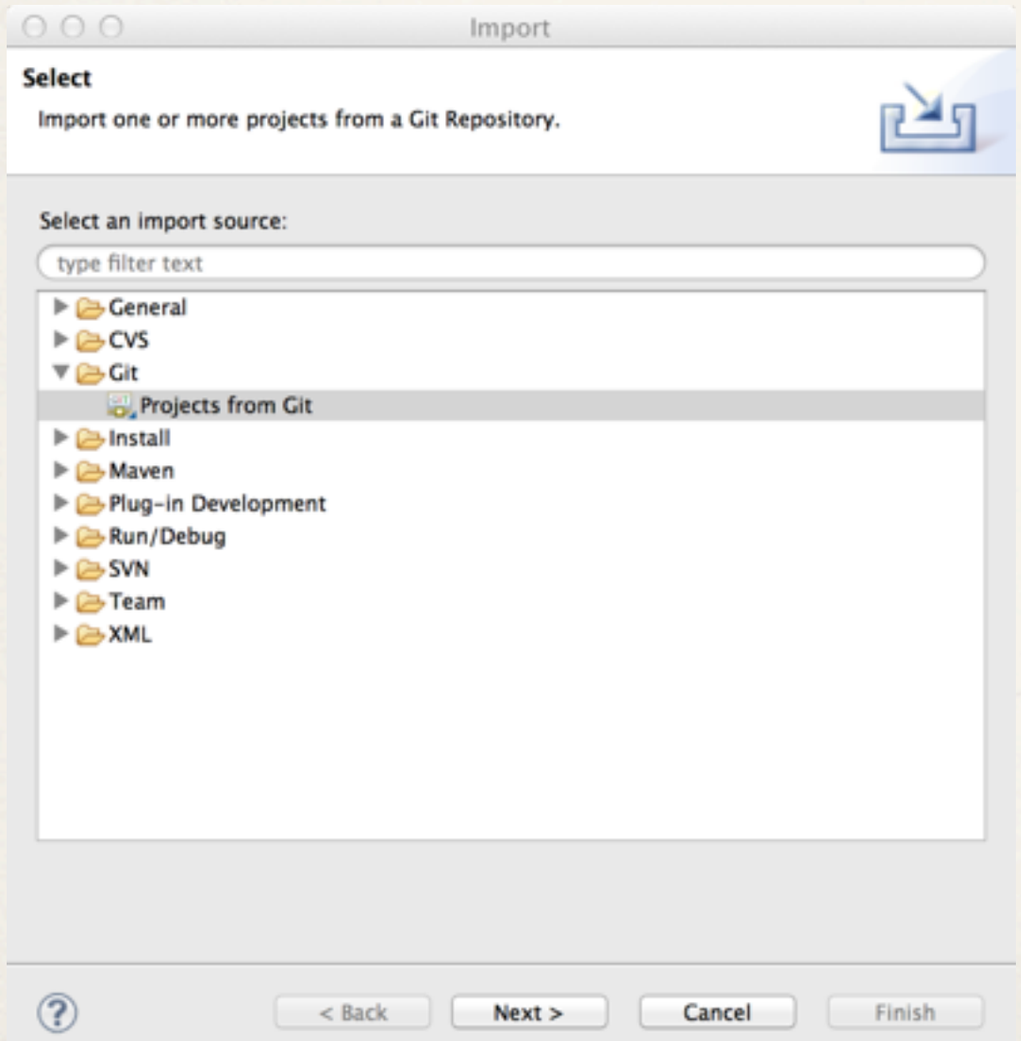
Setup Software

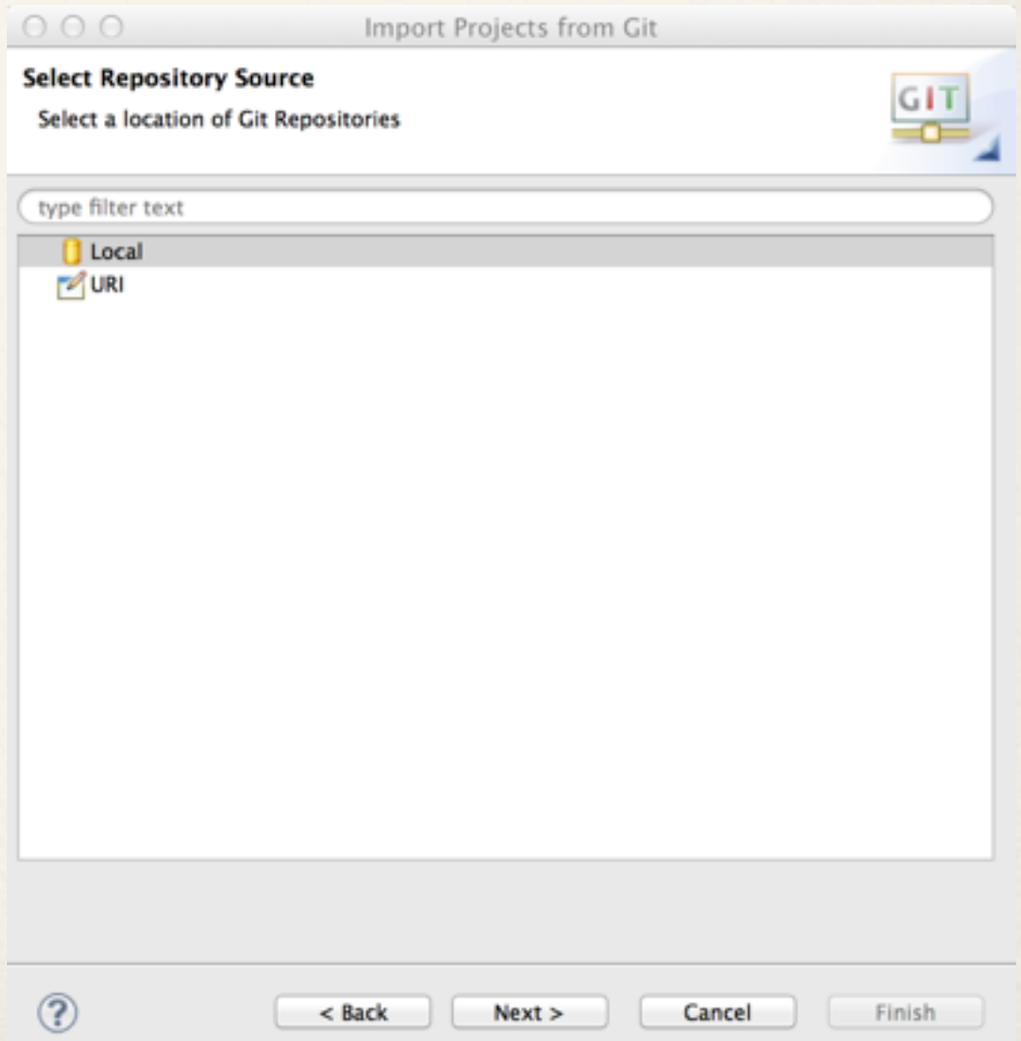
- Launch eclipse and point to select the workspace directory



Setup Workspace

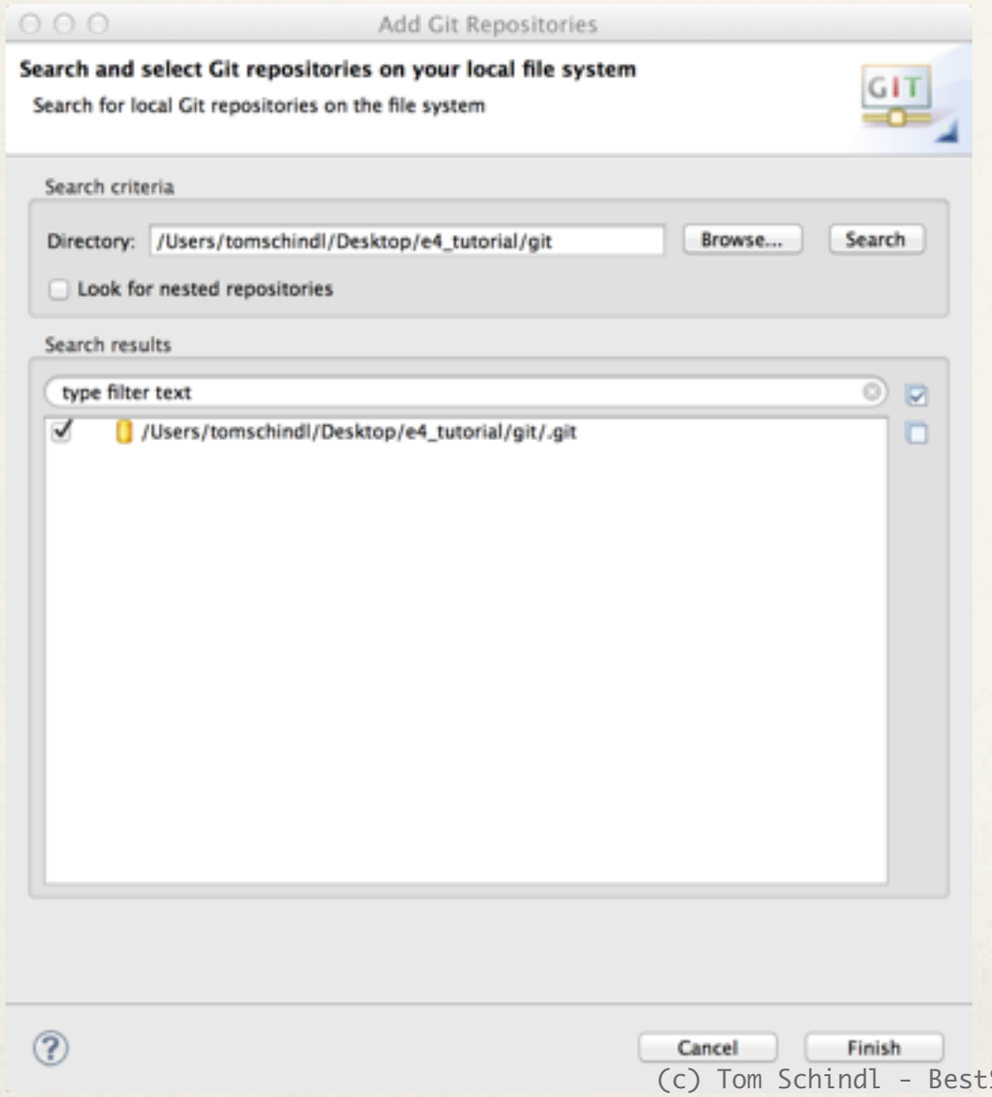
- Bring up the Import-Wizard (File > Import ...) and select Git > Projects from Git

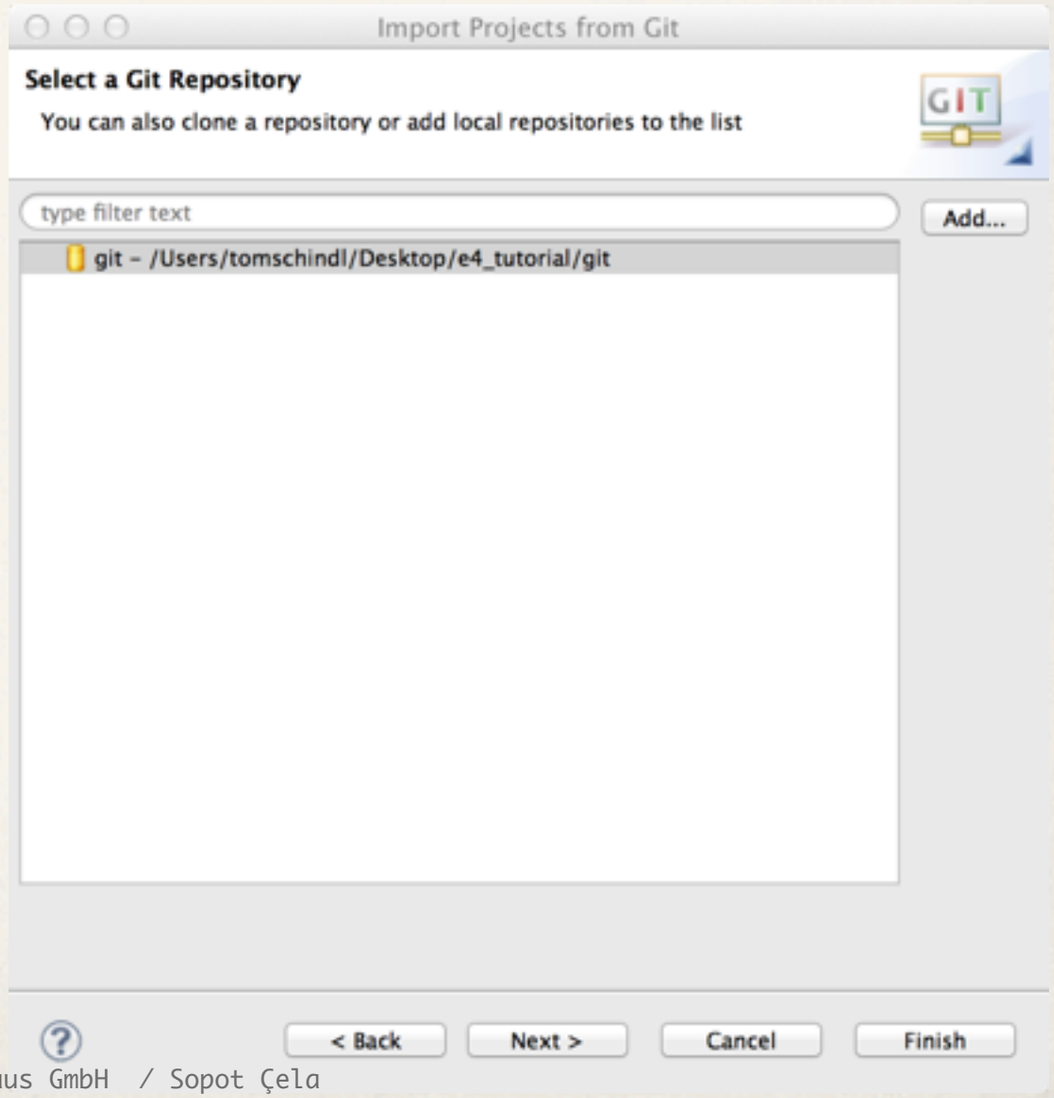




Setup Workspace

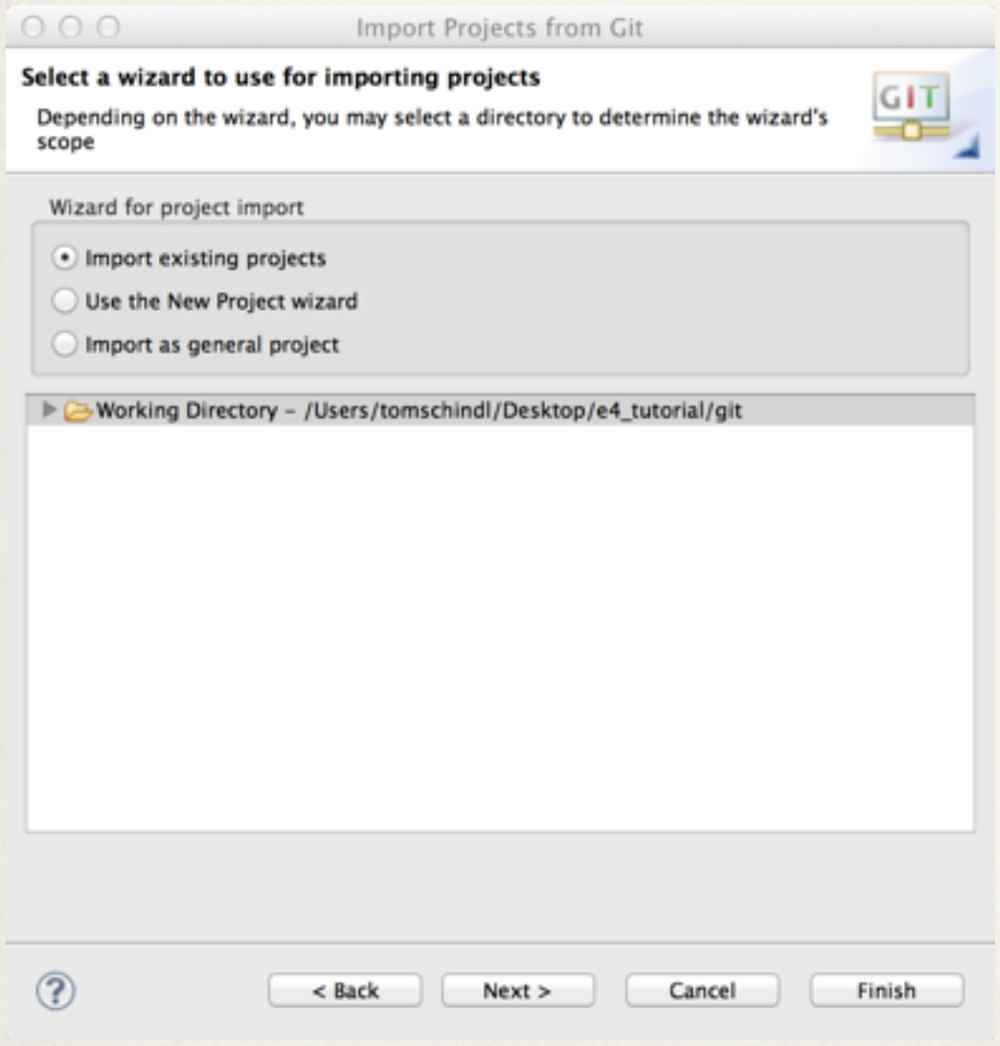
- Point it to the location you extracted your the git.zip to and push „Search“

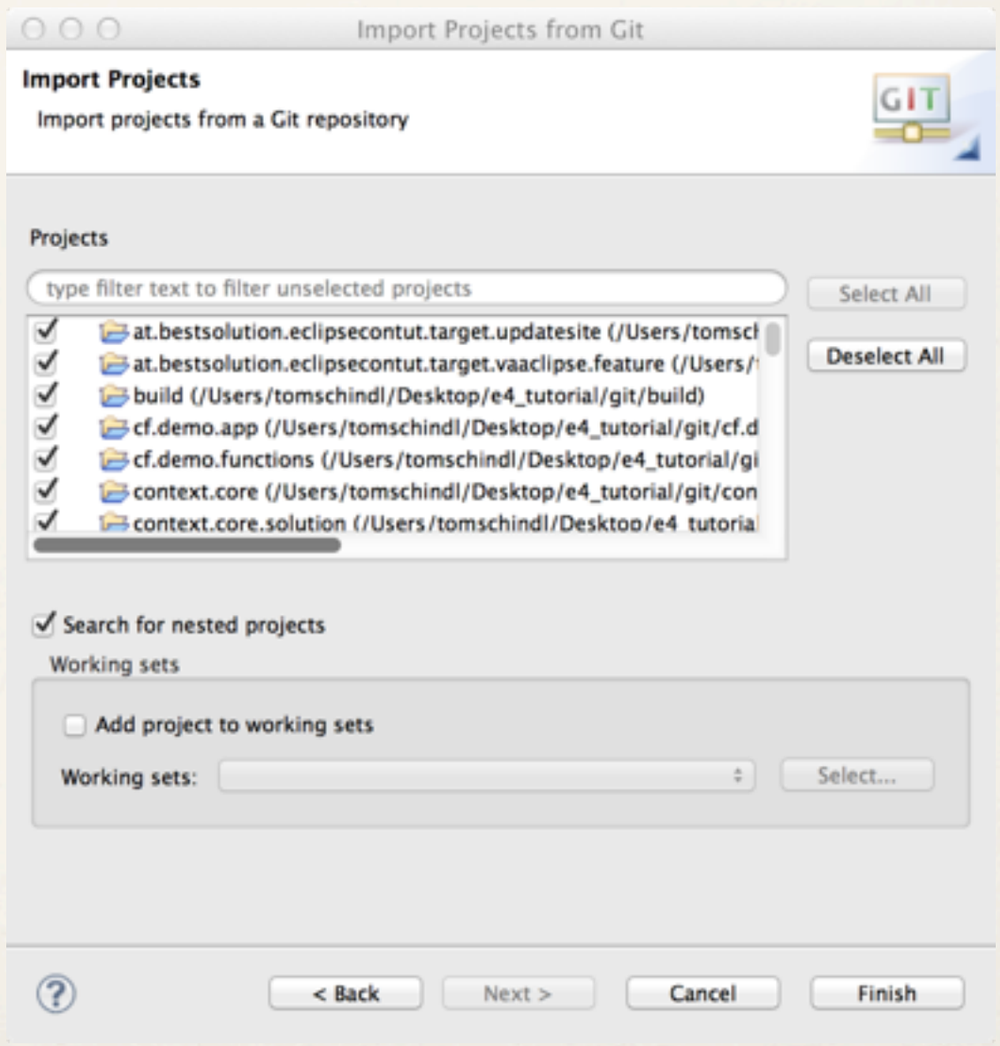




Setup Workspace

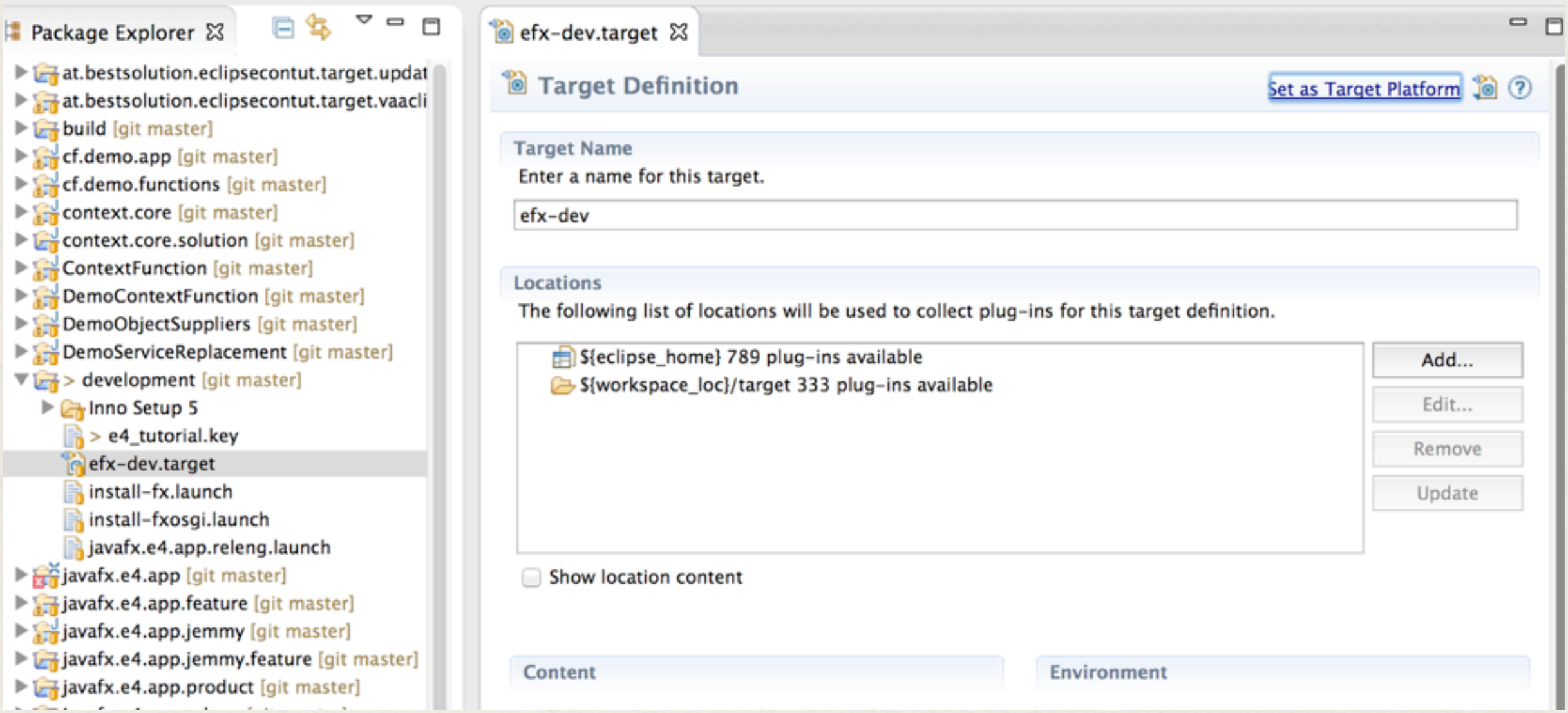
- Import all projects





Setup Workspace

- Set the target-platform



Planned Content

- Architecture Overview (Theory only)
  - Platform runtime
  - Platform UI

Planned Content

- Runtime
  - Working with the IEclipseContext (Theory + Hands-on)
- Extending Eclipse DI using ContextFunctions and Suppliers (Theory + Hands-on)

Planned Content

- UI
  - Application model (Theory only)
  - (Specialize) Workbench Services (Demo only)
- Renderering (SWT)
  - The concept behind the rendering - Model <=> Engine <=> Renderer (Theory + Hands-on)
  - Customize existing SWT-Renderer (Theory + Hands-on)

Planned Content

- UI (continued)
  - Rendering using vaaclipse (Theory + Hands-on)
  - Rendering using JavaFX (Theory + Hands-on)
- Tooling
  - Extending the e4 application model and extending the tooling (Theory + Demo)

Runtime DI

- EAP has a custom DI-Engine
  - JSR 330 compatible
- All informations are stored in IEclipseContext(s)
  - IEclipseContexts are hierarchical
  - The root of the IEclipseContext is bound to the OSGI-Service Registry

Runtime DI - Annotations

- JSR 330
  @Inject, @Named, @Singleton

- JSR 250
  @PostContstruct, @PreDestroy

- Custom annotations
  @Optional, @Active, @Creatable

Runtime DI -important Classes + Methods

Runtime DI -important Classes + Methods

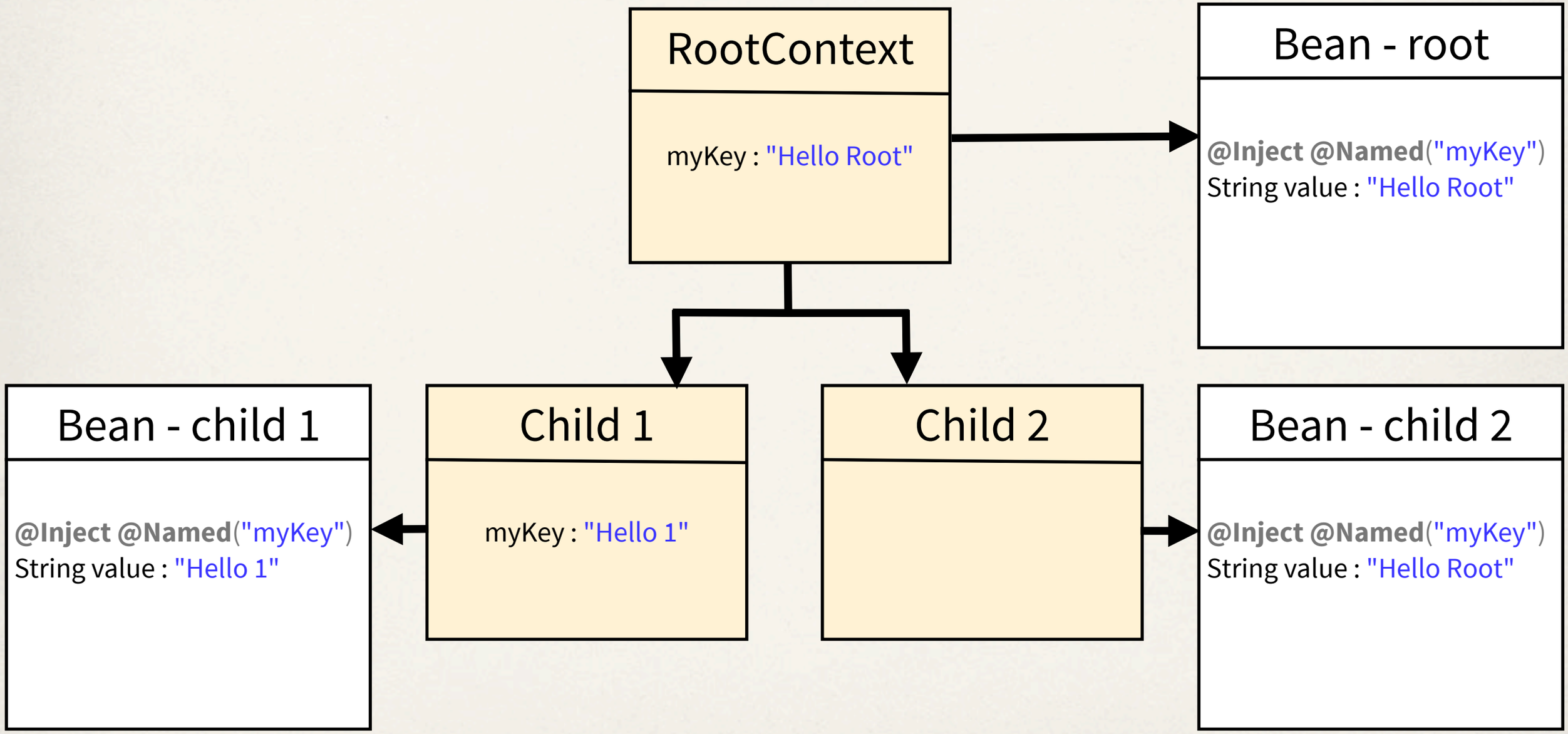
- **IEclipseContext** ... stores values
  #create ... create a child context
  #set ... store a value
  #get ... retrieve a value

Runtime DI -important Classes + Methods

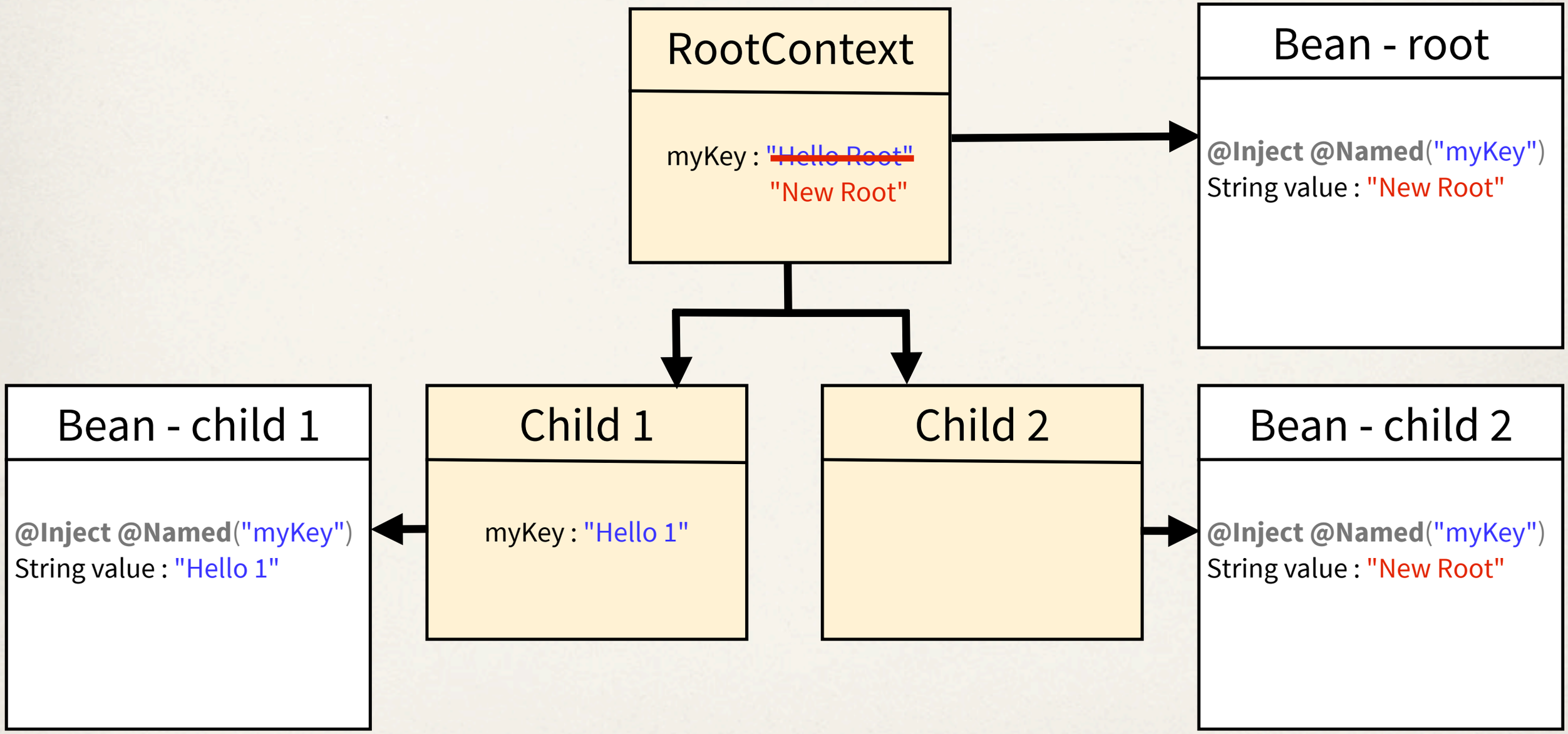
- **IEclipseContext** ... stores values
  #create ... create a child context
  #set ... store a value
  #get ... retrieve a value

- **ContextInjectionFactory** ... creation of instances
  #make ... create an instance of a class
  #inject ... inject infos into an existing object
  #invoke ... invoke a method on an object

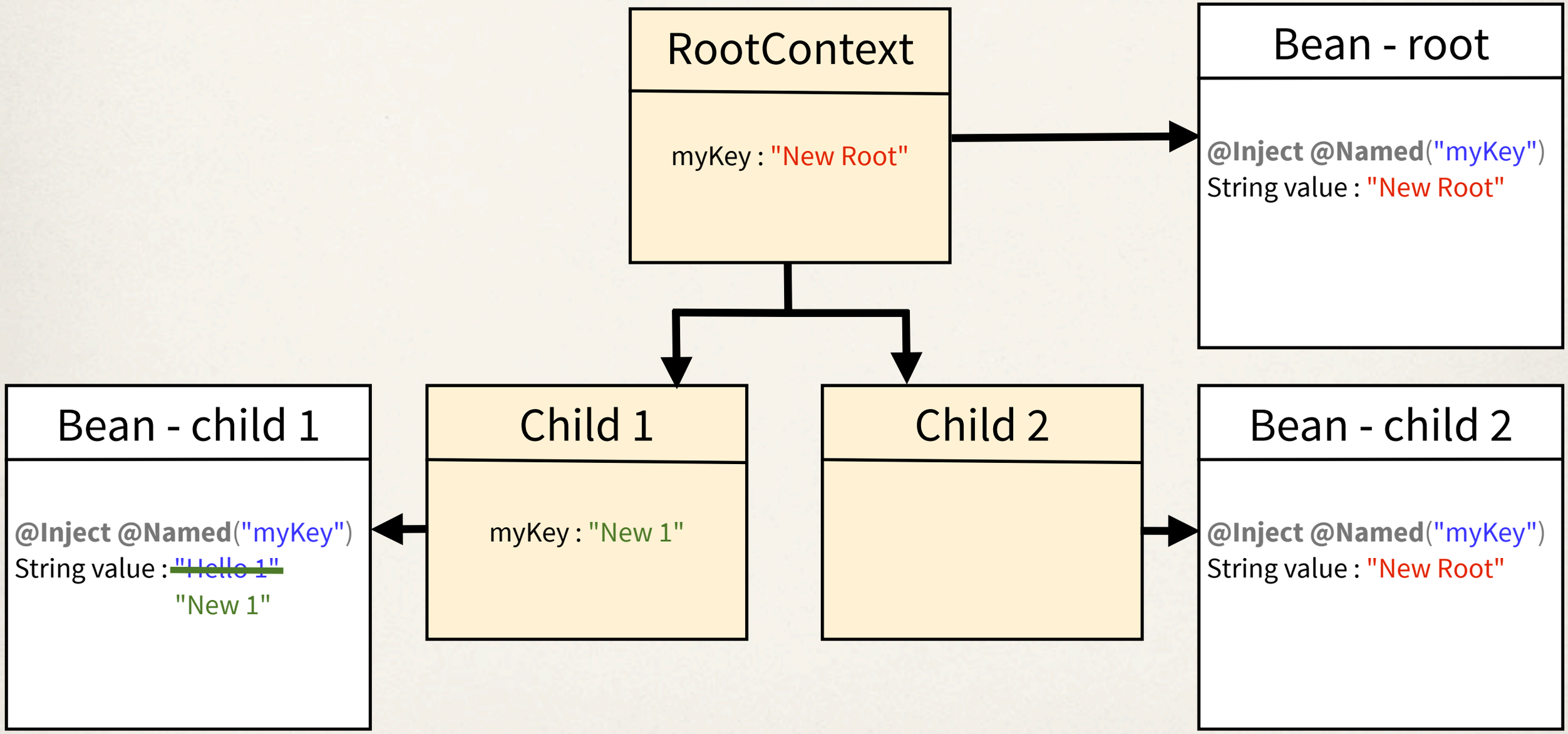
Hierchical contexts



Hierchical contexts



Hierchical contexts



Hierchical contexts

Hierchical contexts

```
BundleContext bundleContext = FrameworkUtil.getBundle(getClass()).getBundleContext();
```

Hierchical contexts

```
BundleContext bundleContext = FrameworkUtil.getBundle(getClass()).getBundleContext();

// Create the root context bound to OSGi-Serviceregistry
IEclipseContext rootContext = EclipseContextFactory.getServiceContext(bundleContext);
rootContext.set("myKey", "Hello World");
```

Hierchical contexts

```
BundleContext bundleContext = FrameworkUtil.getBundle(getClass()).getBundleContext();

// Create the root context bound to OSGi-Serviceregistry
IEclipseContext rootContext = EclipseContextFactory.getServiceContext(bundleContext);
rootContext.set("myKey", "Hello World");

// Create a child contexts
IEclipseContext child1Context = rootContext.createChild("Child 1");
child1Context.set("myKey", "Hello 1");

IEclipseContext child2Context = rootContext.createChild("Child 2");
```

Hierchical contexts

```
BundleContext bundleContext = FrameworkUtil.getBundle(getClass()).getBundleContext();

// Create the root context bound to OSGi-Serviceregistry
IEclipseContext rootContext = EclipseContextFactory.getServiceContext(bundleContext);
rootContext.set("myKey", "Hello World");

// Create a child contexts
IEclipseContext child1Context = rootContext.createChild("Child 1");
child1Context.set("myKey", "Hello 1");

IEclipseContext child2Context = rootContext.createChild("Child 2");

// Create instances of beans bound to different contexts
Bean rootBean = ContextInjectionFactory.make(Bean.class, rootContext);
Bean child1Bean = ContextInjectionFactory.make(Bean.class, child1Context);
Bean child2Bean = ContextInjectionFactory.make(Bean.class, child2Context);
```

Hierchical contexts

```
BundleContext bundleContext = FrameworkUtil.getBundle(getClass()).getBundleContext();

// Create the root context bound to OSGi-Serviceregistry
IEclipseContext rootContext = EclipseContextFactory.getServiceContext(bundleContext);
rootContext.set("myKey", "Hello World");

// Create a child contexts
IEclipseContext child1Context = rootContext.createChild("Child 1");
child1Context.set("myKey", "Hello 1");

IEclipseContext child2Context = rootContext.createChild("Child 2");

// Create instances of beans bound to different contexts
Bean rootBean = ContextInjectionFactory.make(Bean.class, rootContext);
Bean child1Bean = ContextInjectionFactory.make(Bean.class, child1Context);
Bean child2Bean = ContextInjectionFactory.make(Bean.class, child2Context);

// Modify values
rootContext.set("myKey", "New Root");
child1Context.set ("myKey", "New 1");
```

Hierchical contexts

- Summary
  - Values are inherited from Parent to Child
  - on change of value a reinjection in the bean occurs

Hierchical contexts

- Idea of active branches
  - Contexts allow to specify which branch is active
- Important for the UI to e.g. execute actions based on the active parts context

Hierchical contexts

- Idea of active branches
  - Contexts allow to specify which branch is active
  - Important for the UI to e.g. execute actions based on the active parts context

```
IEclipseContext rootContext = ....;
```

Hierchical contexts

- Idea of active branches
  - Contexts allow to specify which branch is active
  - Important for the UI to e.g. execute actions based on the active parts context

```
IEclipseContext rootContext = ....;
IEclipseContext child1Context = rootContext.createChild();
```

Hierchical contexts

- Idea of active branches
  - Contexts allow to specify which branch is active
  - Important for the UI to e.g. execute actions based on the active parts context

```
IEclipseContext rootContext = ....;
IEclipseContext child1Context = rootContext.createChild();
IEclipseContext child2Context = rootContext.createChild();
```

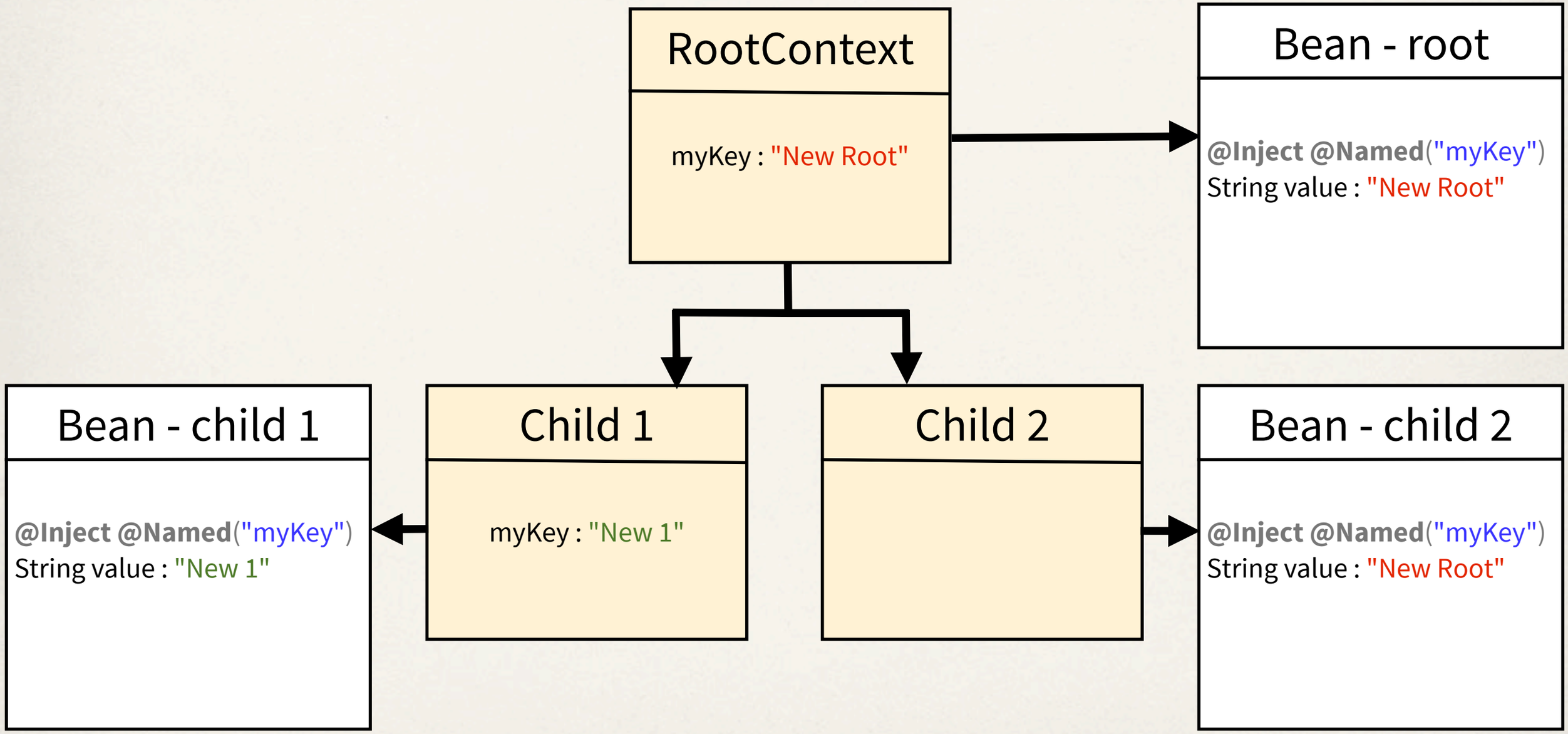
Hierchical contexts

- Idea of active branches
  - Contexts allow to specify which branch is active
  - Important for the UI to e.g. execute actions based on the active parts context

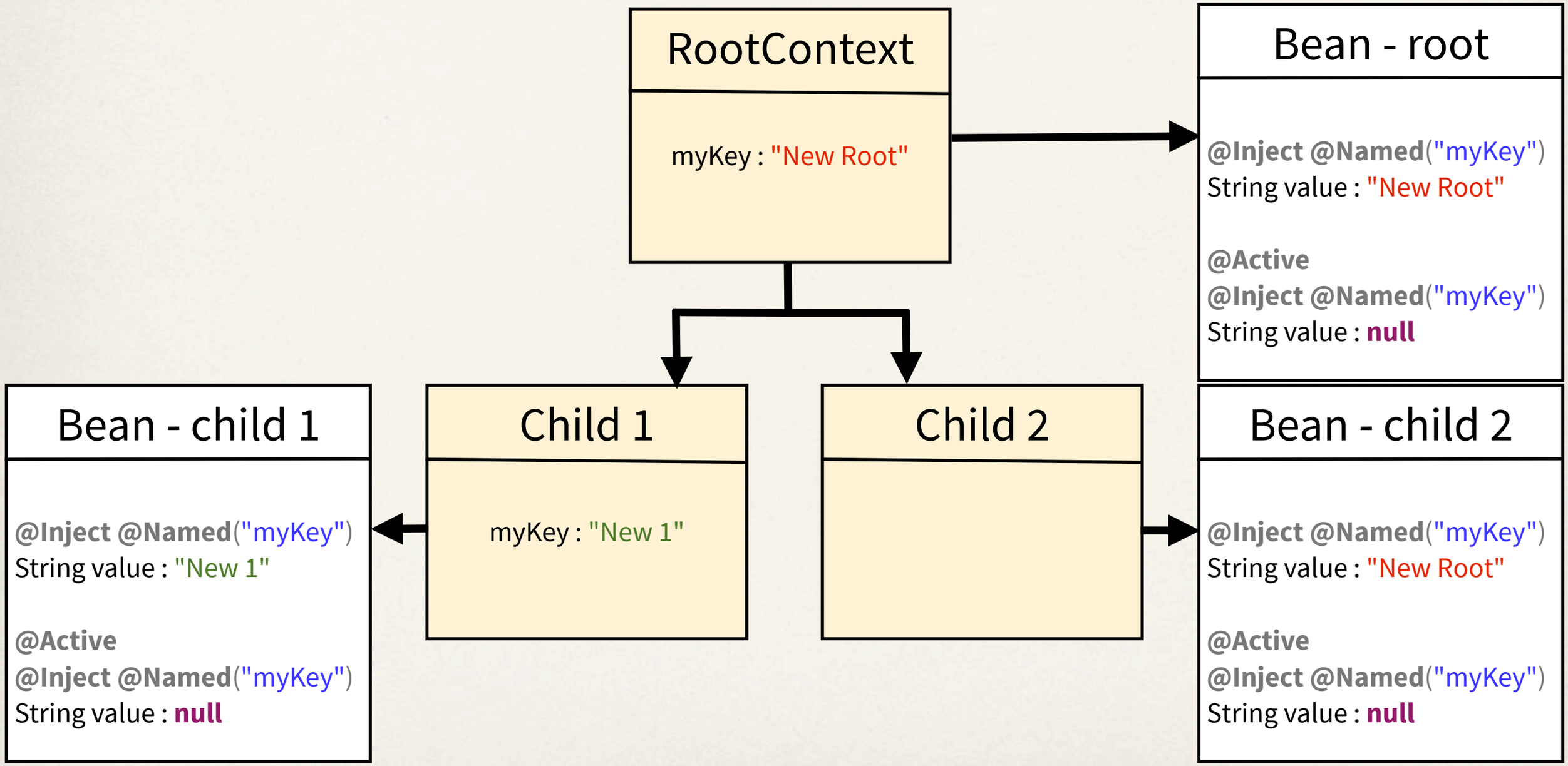
```
IEclipseContext rootContext = ....;
IEclipseContext child1Context = rootContext.createChild();
IEclipseContext child2Context = rootContext.createChild();

child1Context.activateBranch();
```

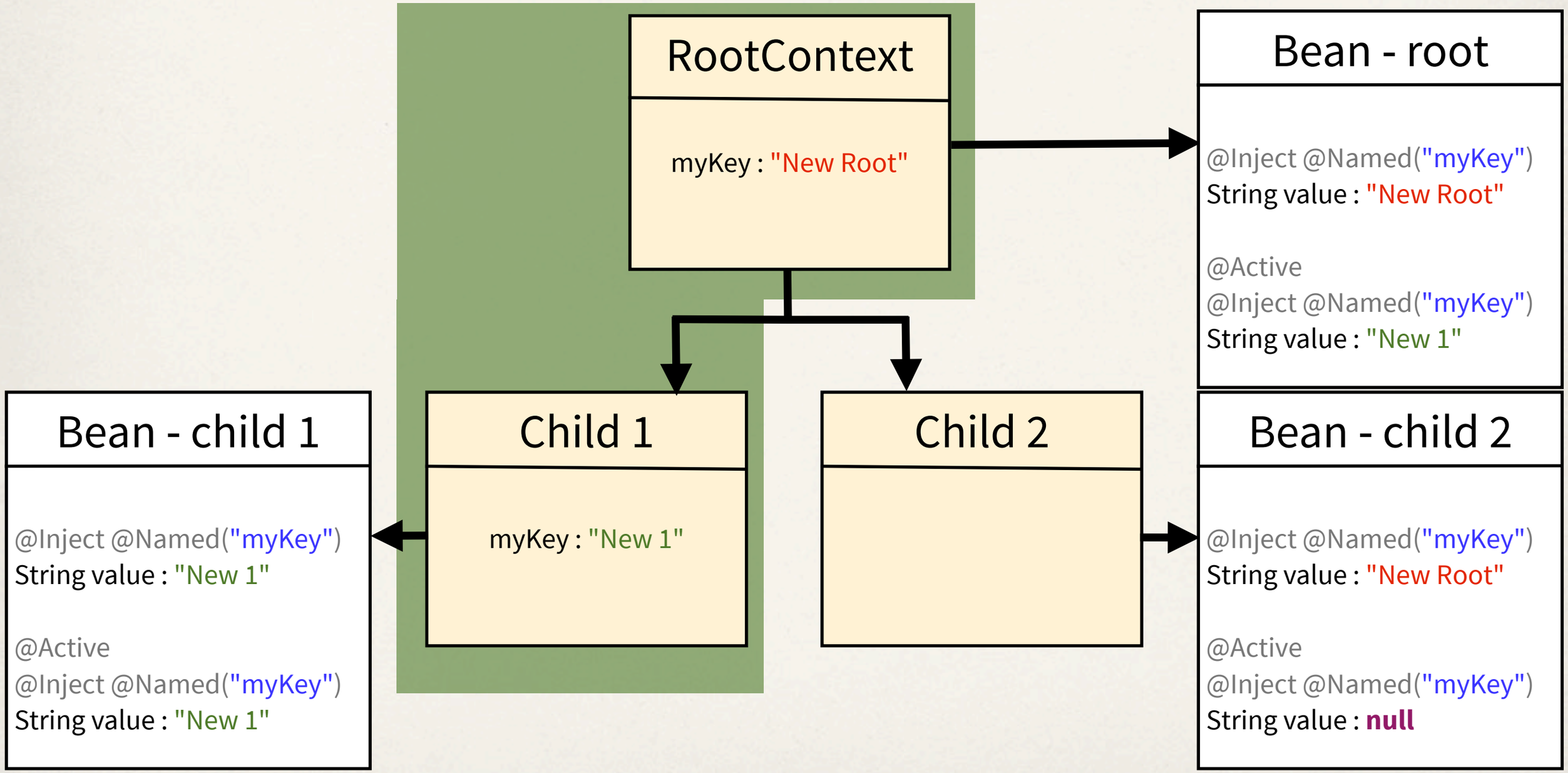
Hierchical contexts



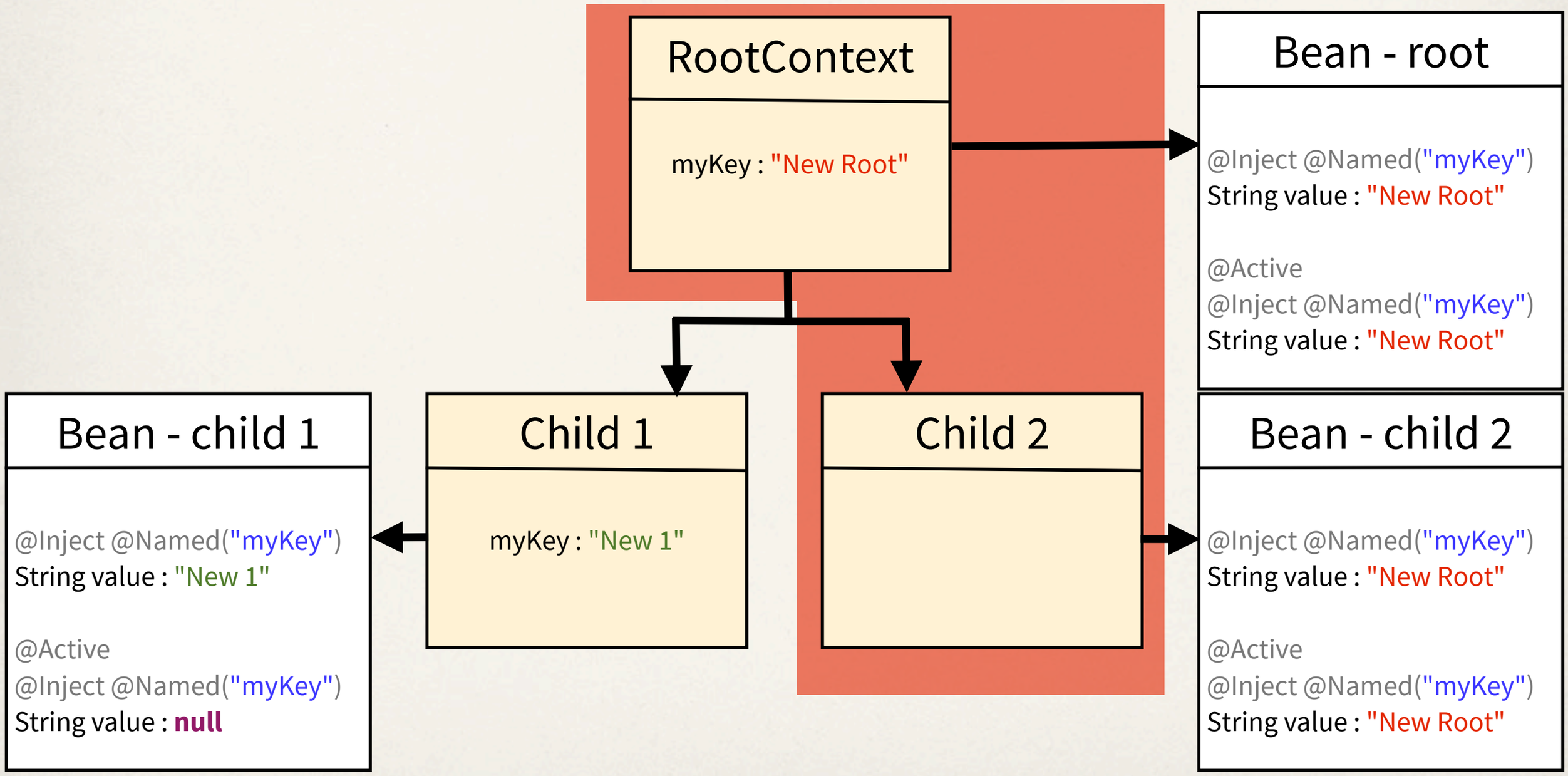
Hierchical contexts



Hierchical contexts



Hierchical contexts



Method calling

- Calling methods
  - Replacement for implementing interfaces
- „Method-Injection“ on request

Method calling

Method calling

```
public class ActionBean {

    @Execute
    private boolean helloWorld(@Named("myKey") String value) {
        //....
        return value.equals("Hello World");
    }
}
```

Method calling

```
public class ActionBean {

	@Execute
	private boolean helloWorld(@Named("myKey") String value) {
		//....
		return value.equals("Hello World");
	}
}
```

```
IEclipseContext rootContext = EclipseContextFactory.getServiceContext(bundleContext);
rootContext.set("myKey", "Hello World");
```

Method calling

```
public class ActionBean {

	@Execute
	private boolean helloWorld(@Named("myKey") String value) {
		//....
		return value.equals("Hello World");
	}
}
```

```
IEclipseContext rootContext = EclipseContextFactory.getServiceContext(bundleContext);
rootContext.set("myKey", "Hello World");

IEclipseContext child1Context = rootContext.createChild("Child 1");
child1Context.set("myKey", "Hello 1");
```

Method calling

```
public class ActionBean {

	@Execute
	private boolean helloWorld(@Named("myKey") String value) {
		//....
		return value.equals("Hello World");
	}
}
```

```
IEclipseContext rootContext = EclipseContextFactory.getServiceContext(bundleContext);
rootContext.set("myKey", "Hello World");

IEclipseContext child1Context = rootContext.createChild("Child 1");
child1Context.set("myKey", "Hello 1");

ActionBean actionBean = ContextInjectionFactory.make(ActionBean.class, rootContext);
boolean success = (Boolean)ContextInjectionFactory.invoke(actionBean, Execute.class, rootContext); // returns true
boolean success = (Boolean)ContextInjectionFactory.invoke(actionBean, Execute.class, child1Context); // returns false
```

Hierchical contexts

- Hands-on 1
  - Go to project context.core
  - Fix TODOs
    a) Application.java
    b) ColorSelectorUI.java
    c) ColorChooserAction.java
    d) ColorItem.java
    e) PreviewItem.java

Off to Sopot

Application Model

- Model is defined using .ecore
- Model does not only hold UI-Informations (e.g. Commands and Handlers)

- Different Element Types

UI-Elements	Mix-ins	NONE-UI-Elements
MWindow	MContext	MCommand
MPerspective	MContribution	MHandler
MPart	MUILabel	MBindingTable
....	....	....

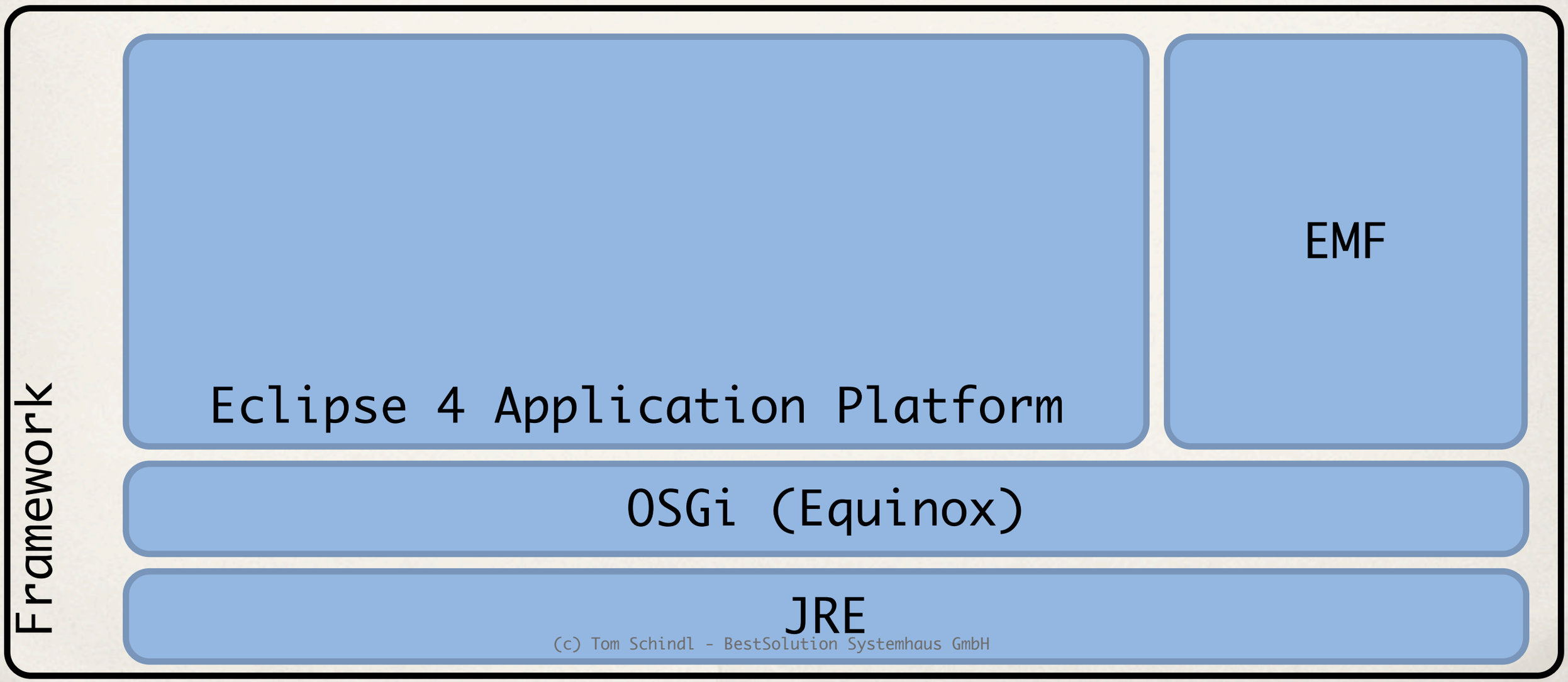
Application Model

- MContext-Mixin (e.g. MWindow, MPerspective, MPart)
  - Defines the context hierarchy of the workbench
  - A context has to be created & set before renderered
- MContribution-Mixin
  - Contributions of external code (bundleclass://$bundleId/$fqnClassName)
  - Instance creation through IContributionFactory

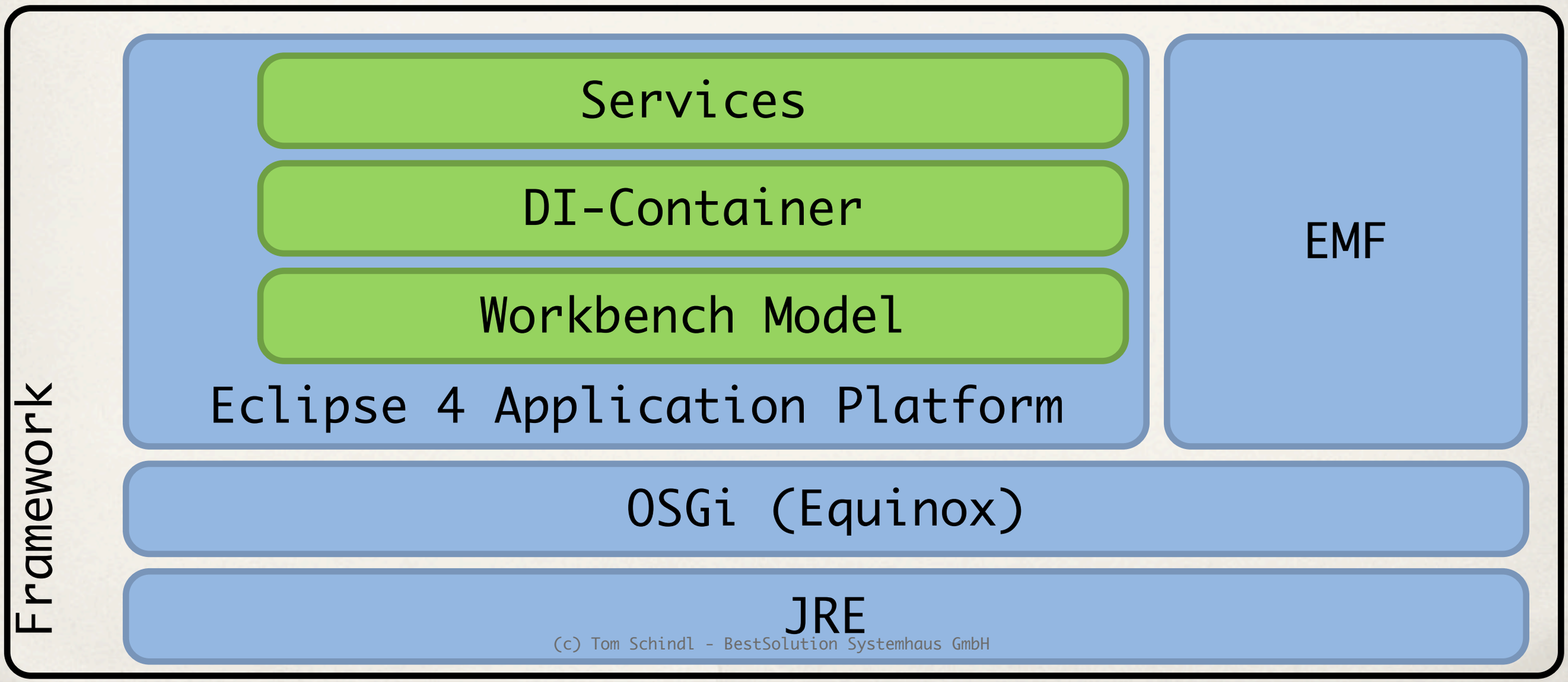
Off to Sopot

Rendering concept

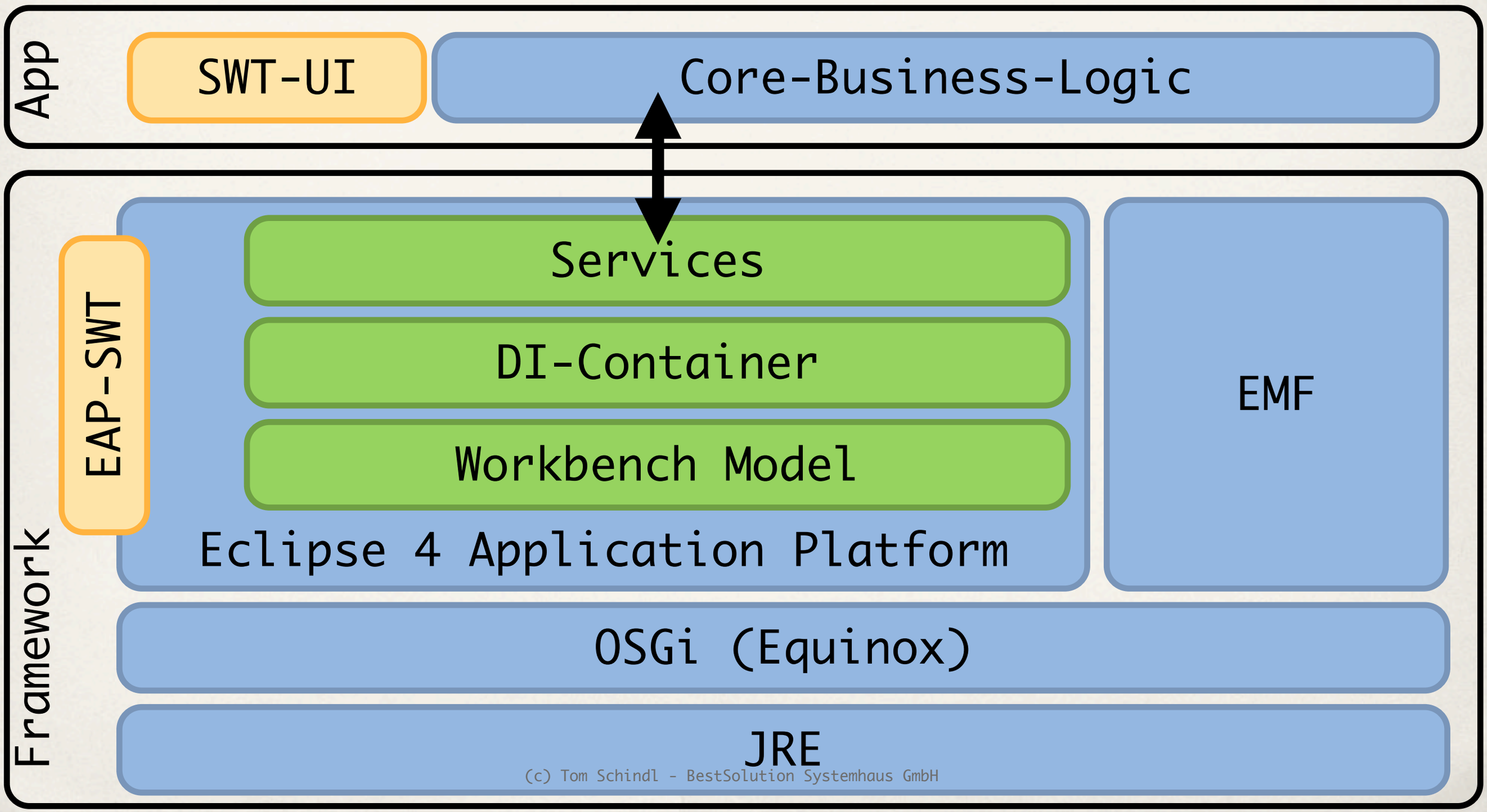
Rendering concept



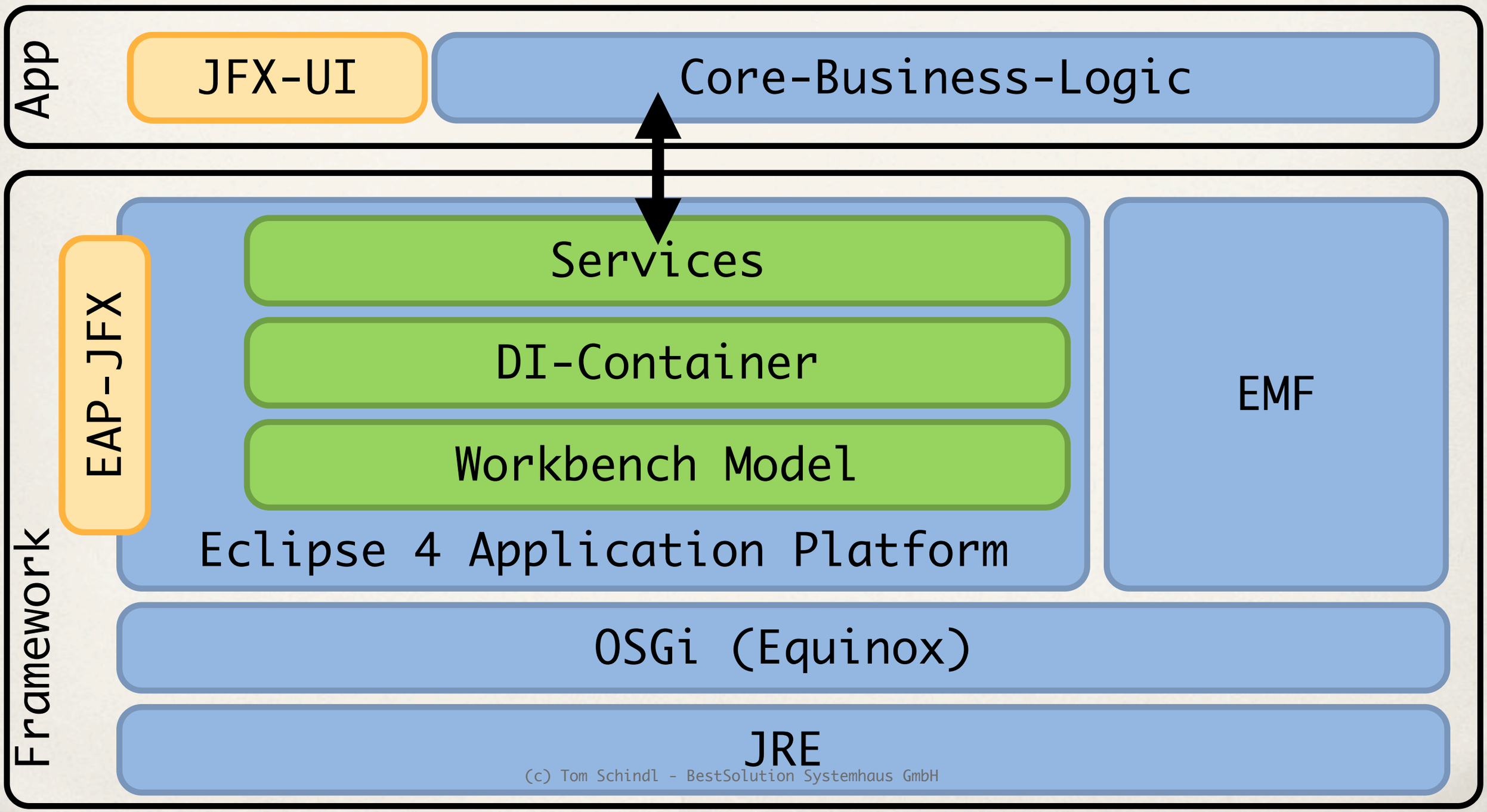
Rendering concept



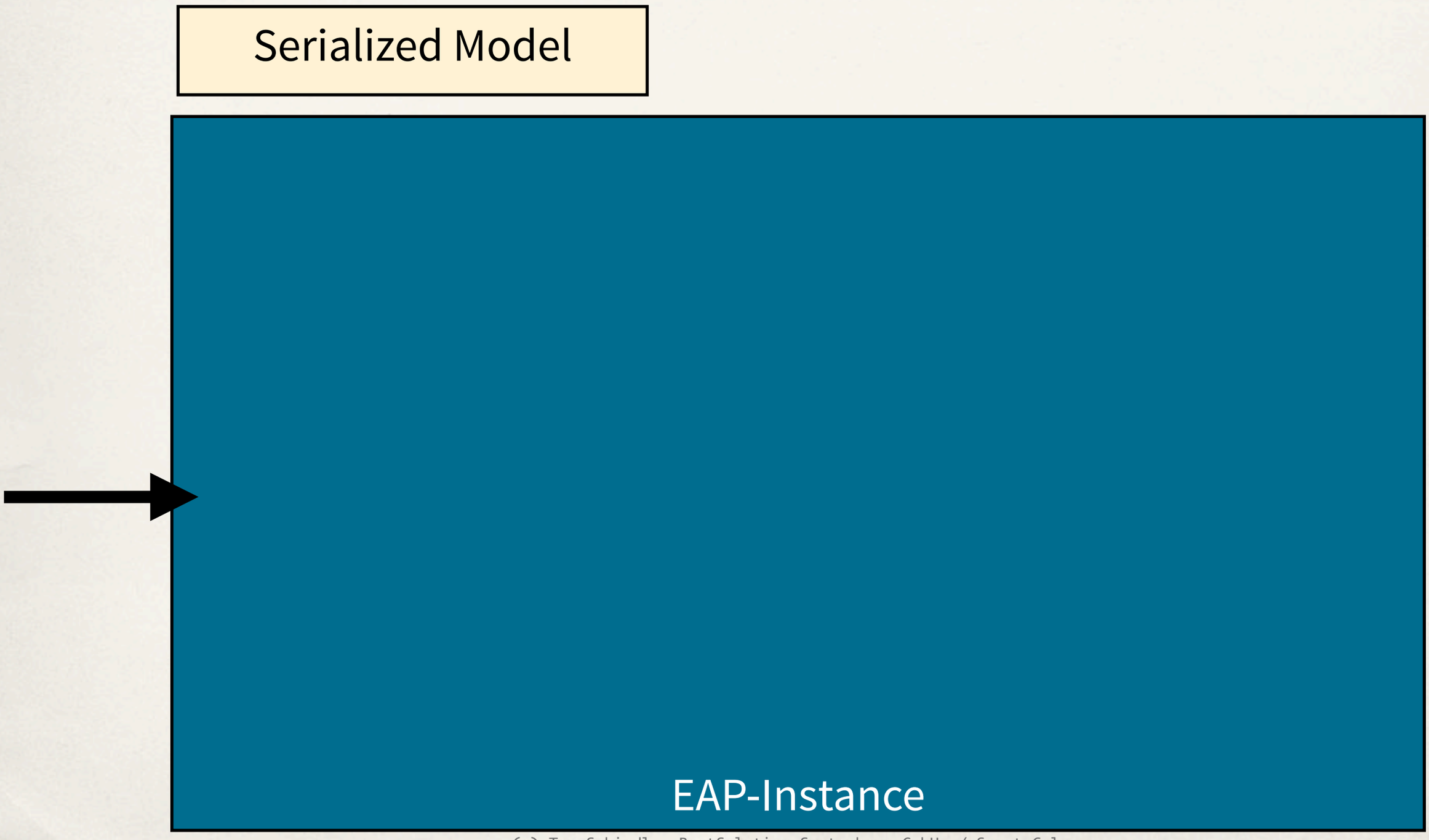
Rendering concept



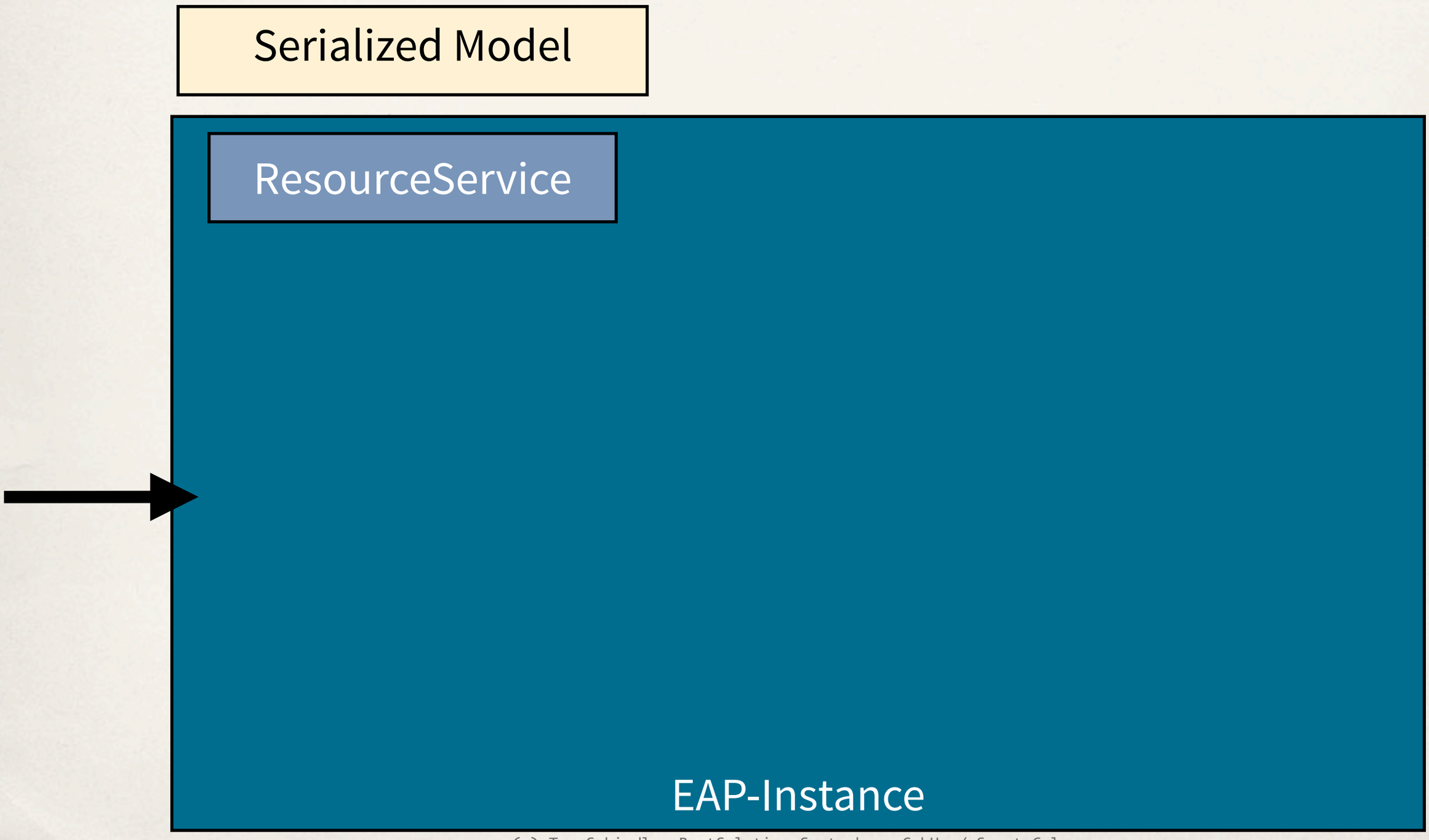
Rendering concept



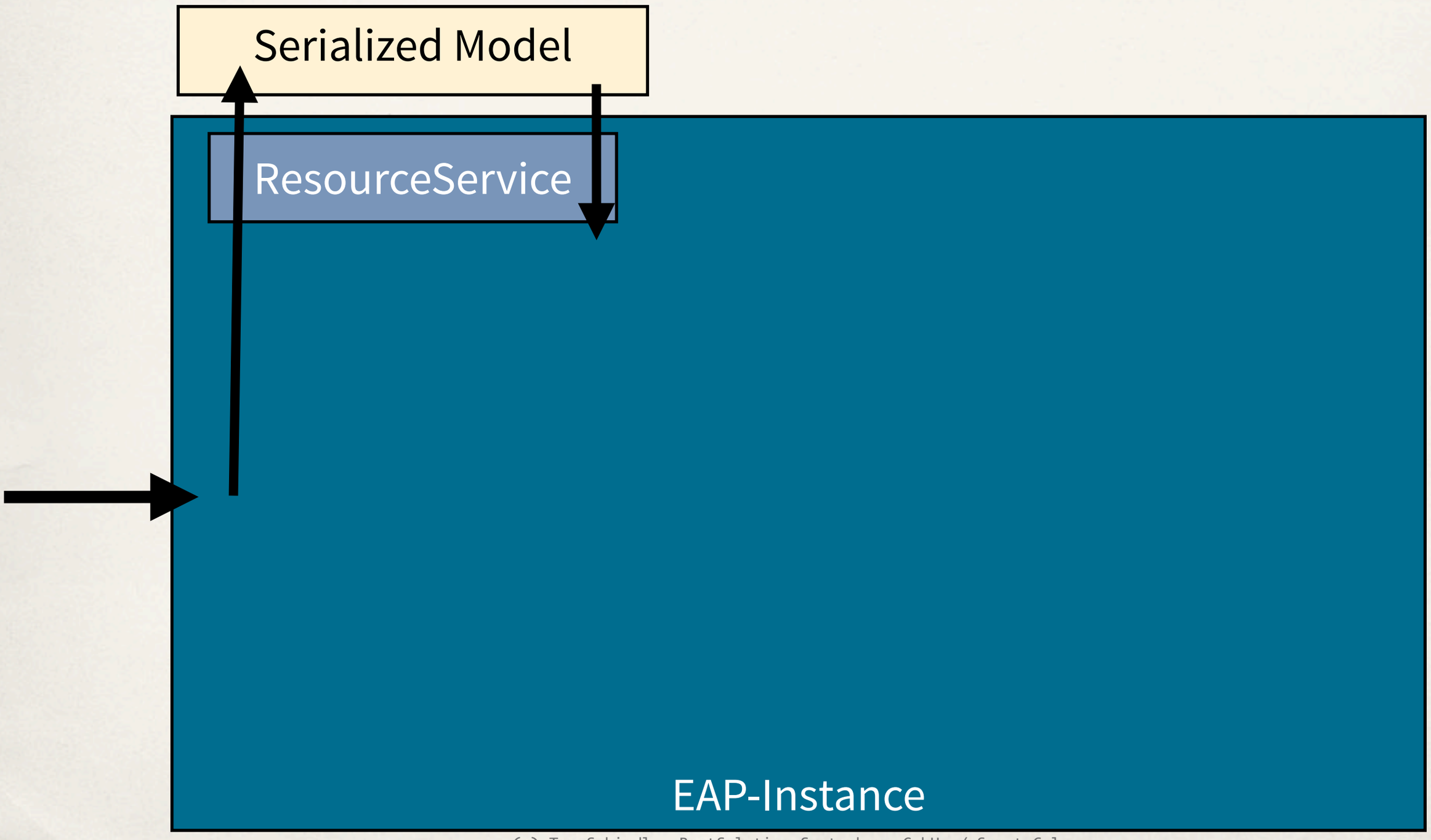
Rendering concept



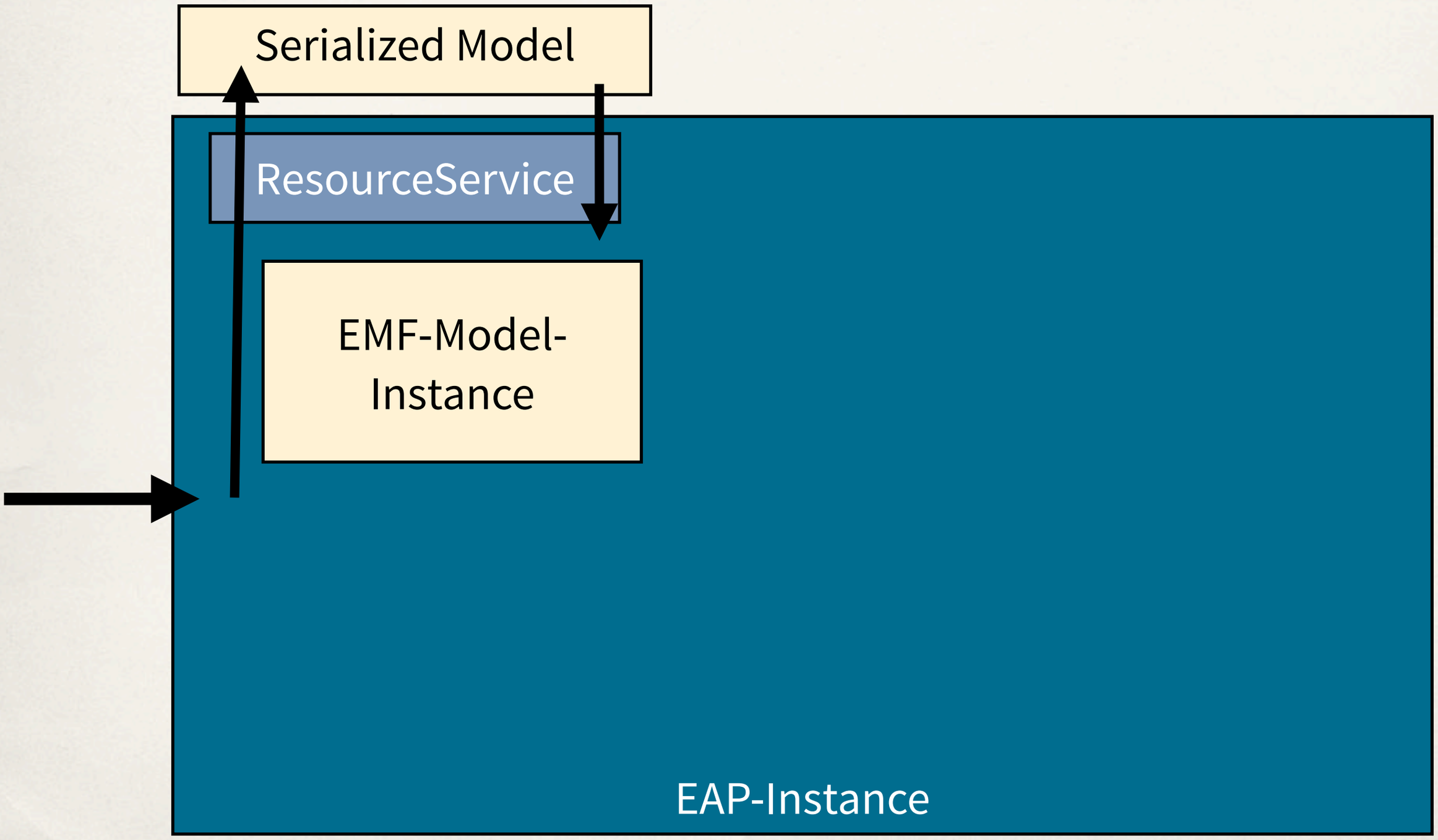
Rendering concept



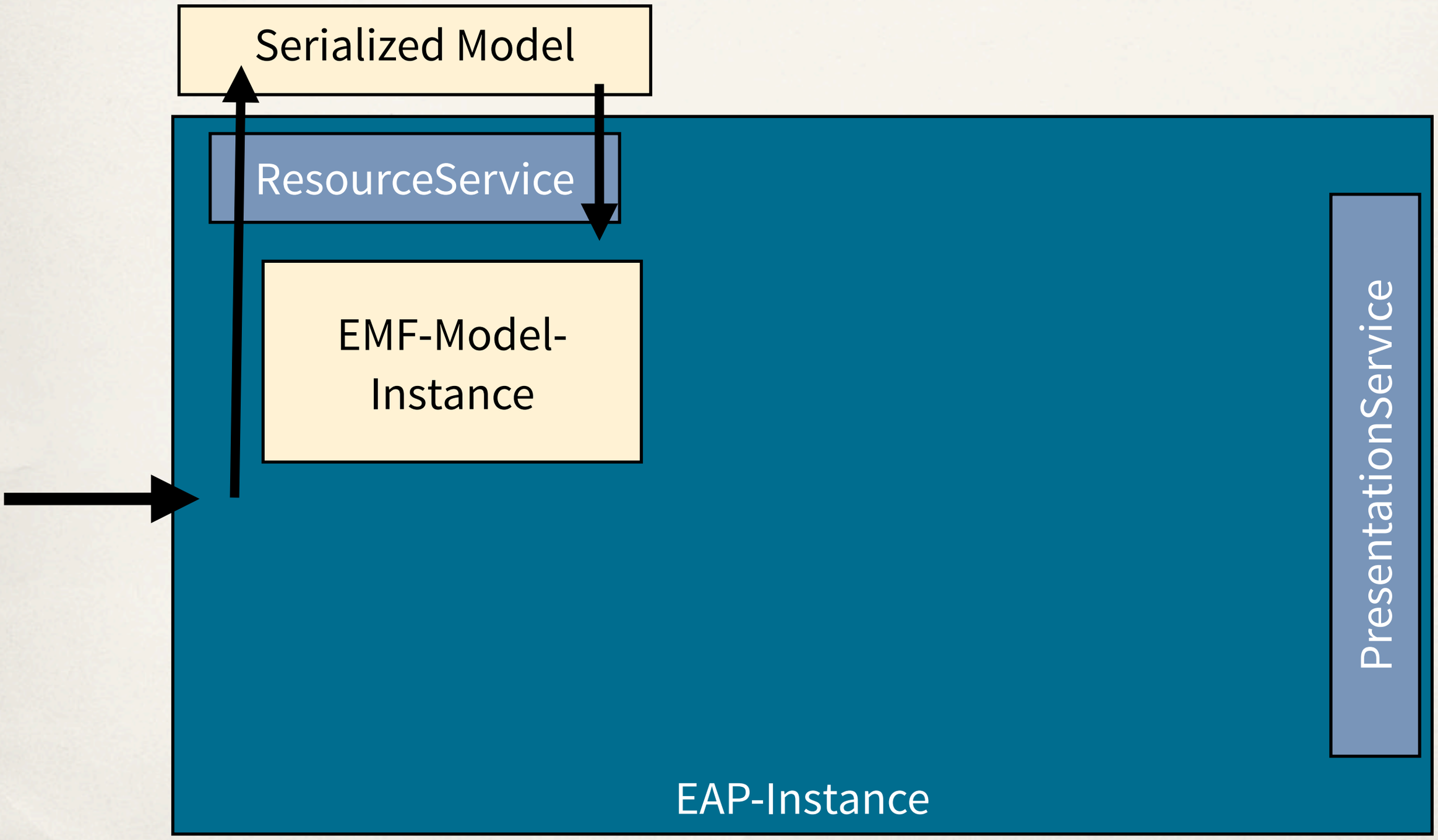
Rendering concept



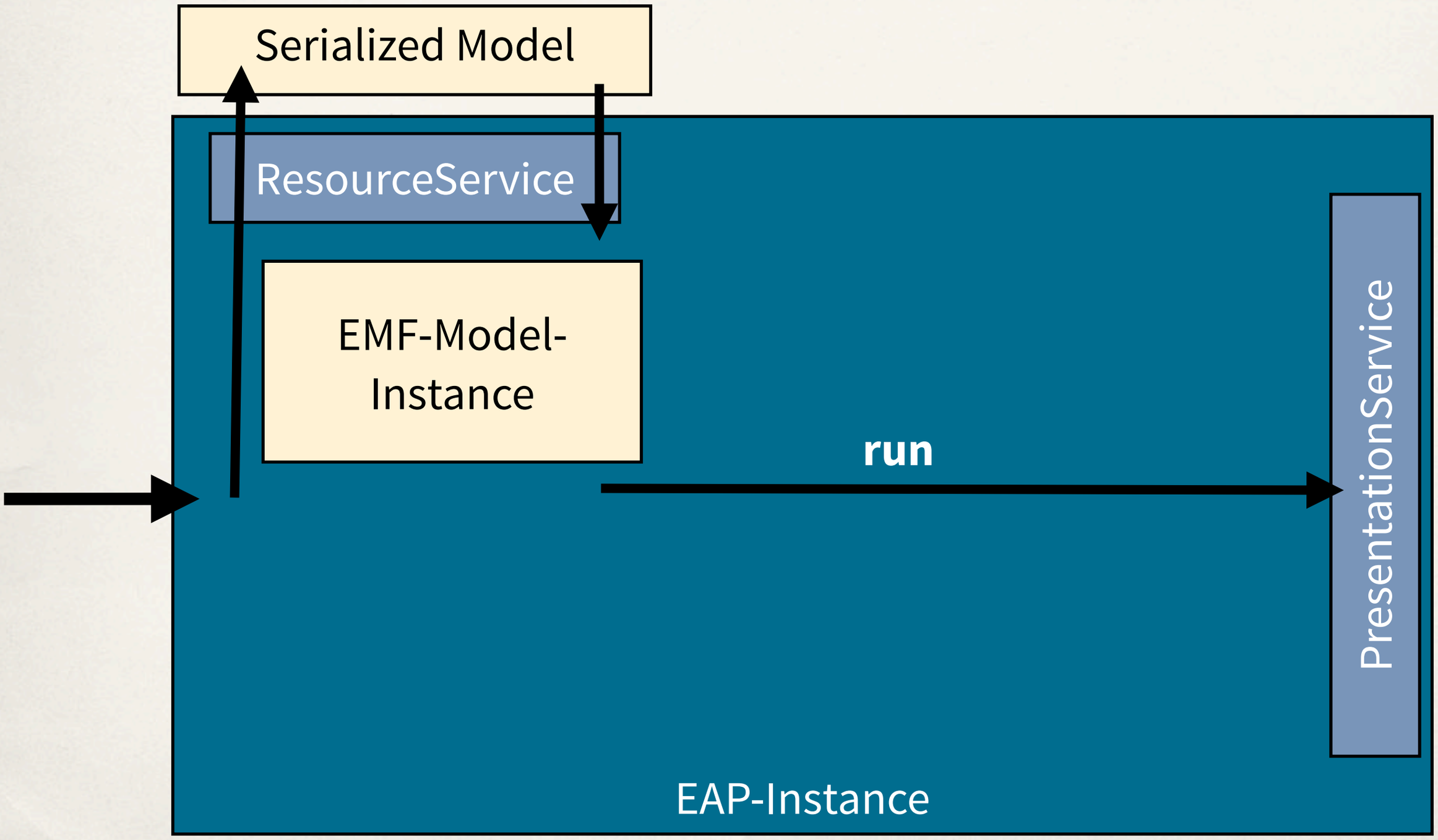
Rendering concept



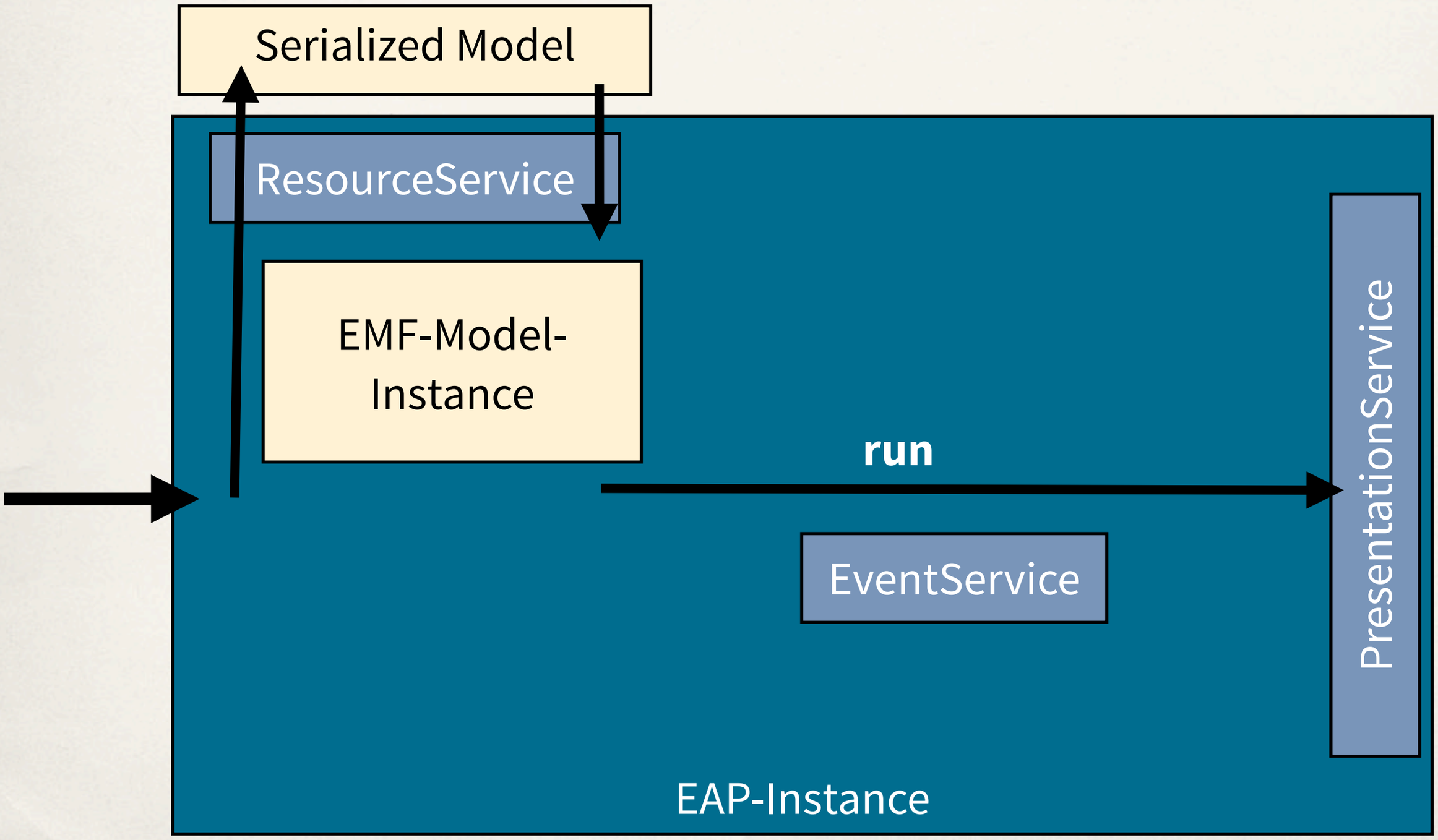
Rendering concept



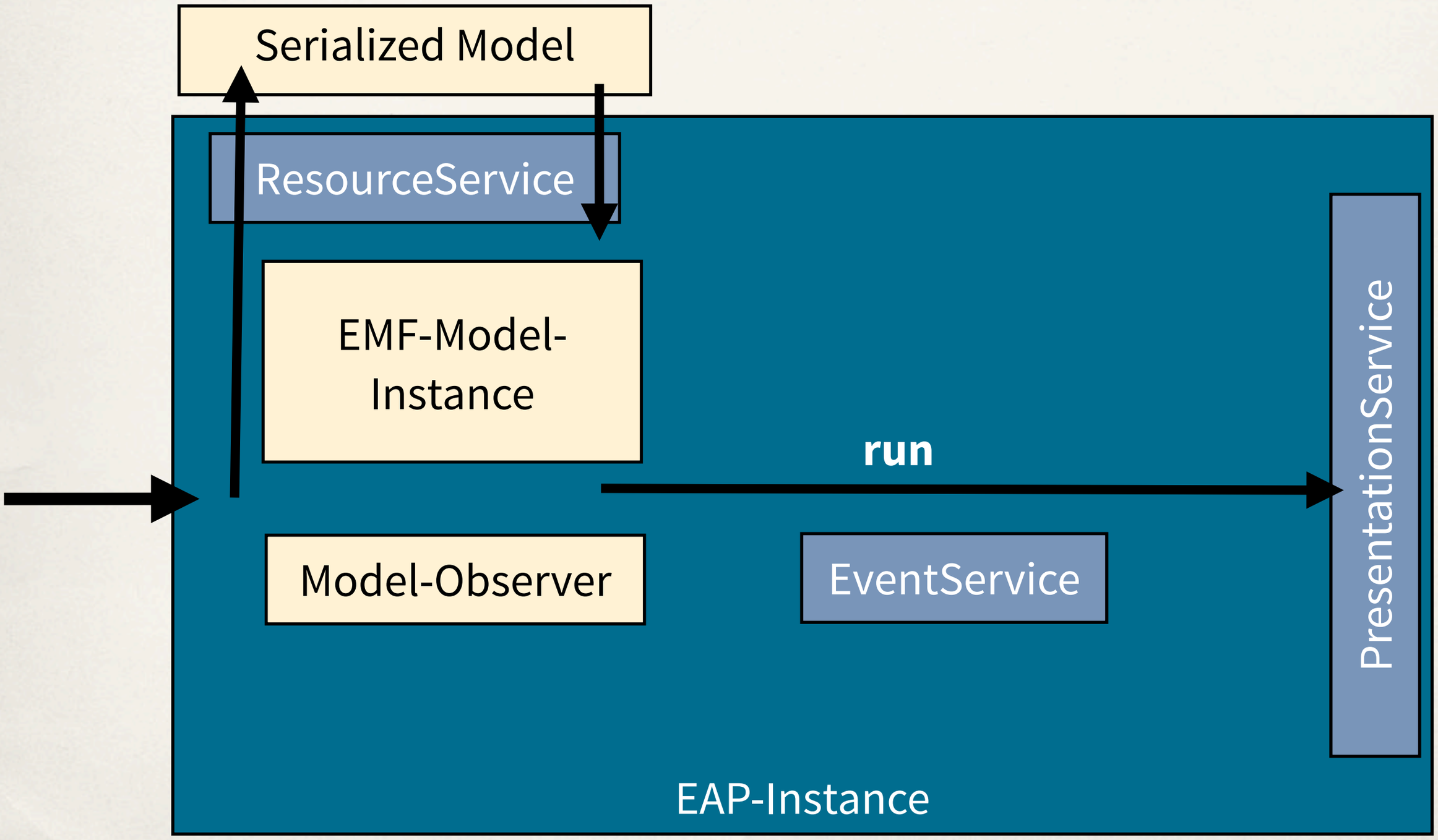
Rendering concept



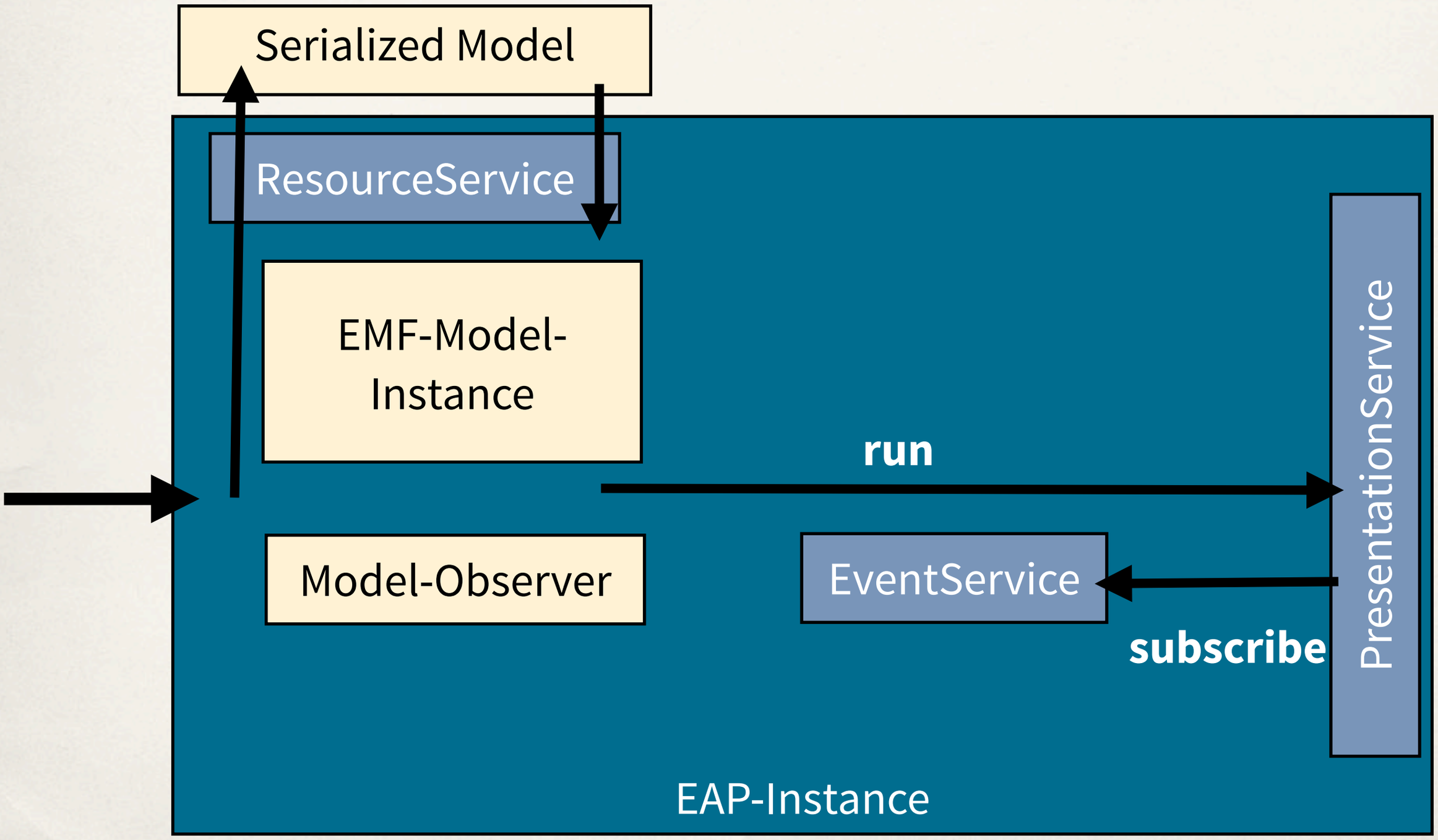
Rendering concept



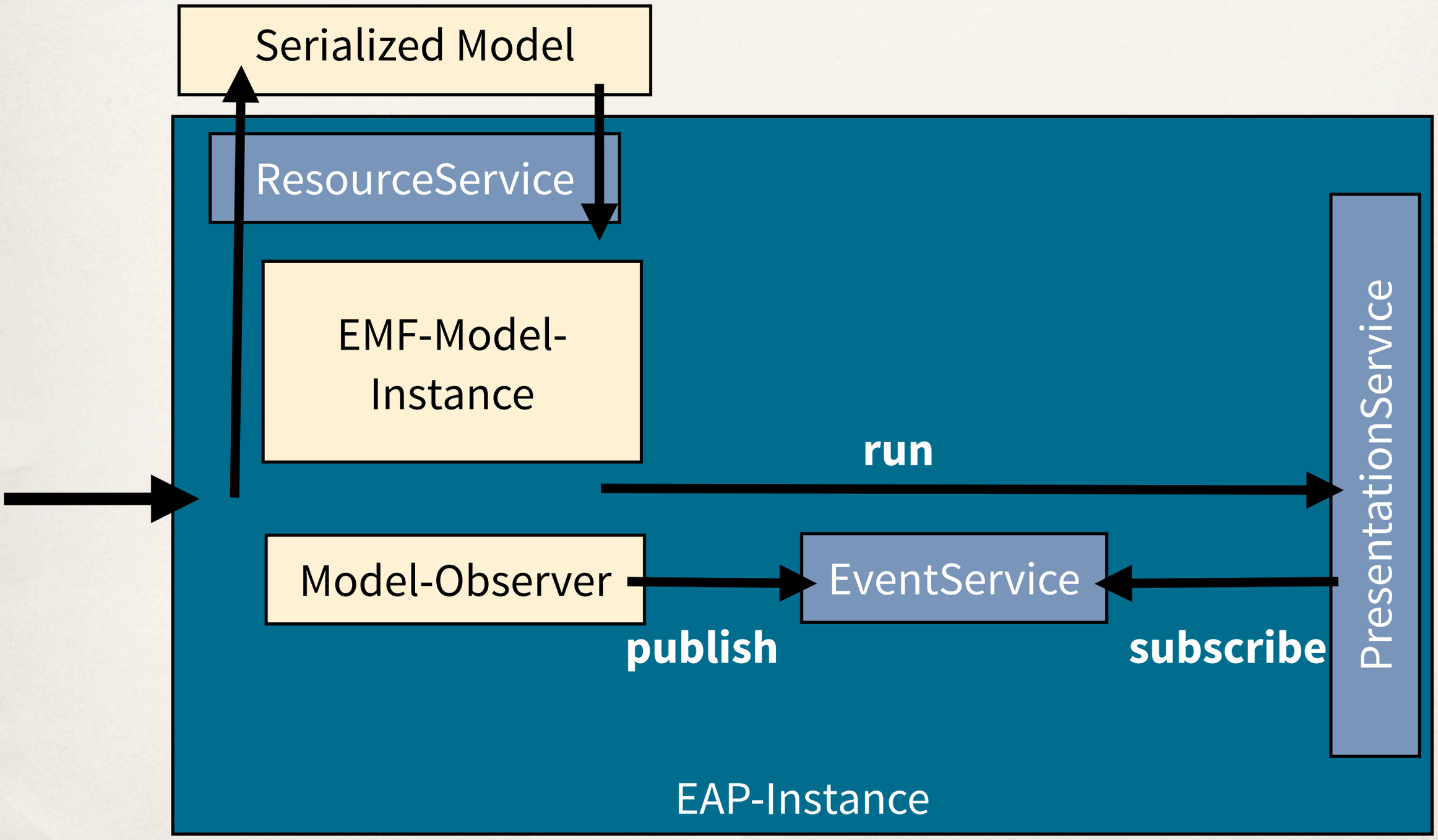
Rendering concept



Rendering concept



Rendering concept



Rendering concept

- Presentation-Service API (IPresentatioEngine - 5 methods only!)

- Important methods
  #run ... running the application
  #createGui ... create UI-Element from model-element
  #removeGui ... destroying UI-Element

- SWT-Default:
  org.eclipse.e4.ui.internal.workbench.swt.PartRenderingEngine

Rendering concept

- Model changes are delivered through the IEventBroker

- Important Event-Fields (in UIEvents.EventTags)
  EventTags.ELEMENT ... the element modified
  EventTags.NEW_VALUE ... the new value
  EventTags.OLD_VALUE ... the old value

Rendering concept

- Replacing with your own

```
<extension
    id="product"
    point="org.eclipse.core.runtime.products">
  <product
    application="org.eclipse.e4.ui.workbench.swt.E4Application"
    name="simple.swt.app">
   <!-- ... -->
   <property
      name="presentationURI"
      value="bundleclass://simple.swt.renderer.solution/simple.swt.renderer.SimpleRenderingEngine">
   </property>
  <!-- ... -->
 <product
```

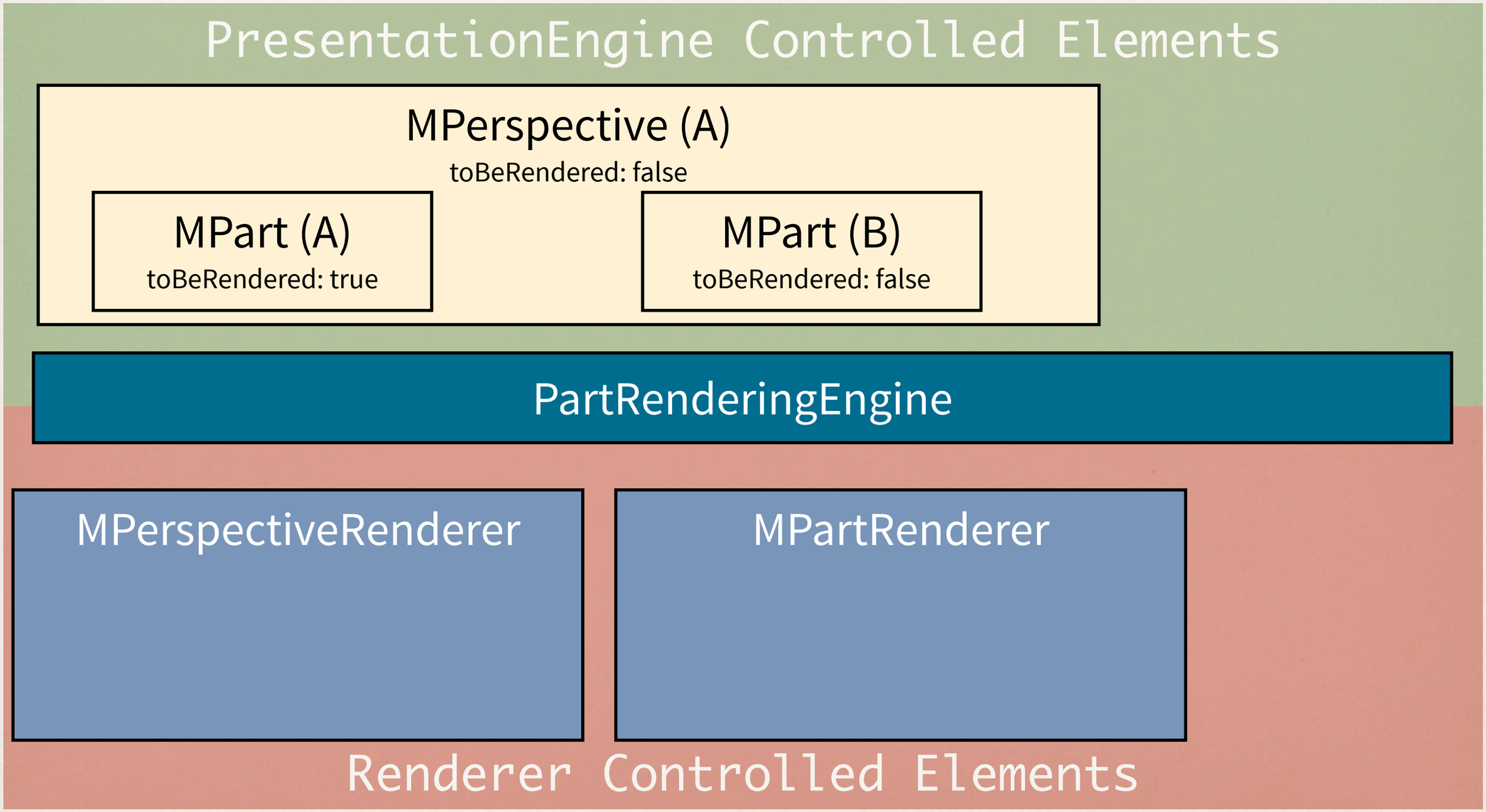
Rendering concept

- Hands-on

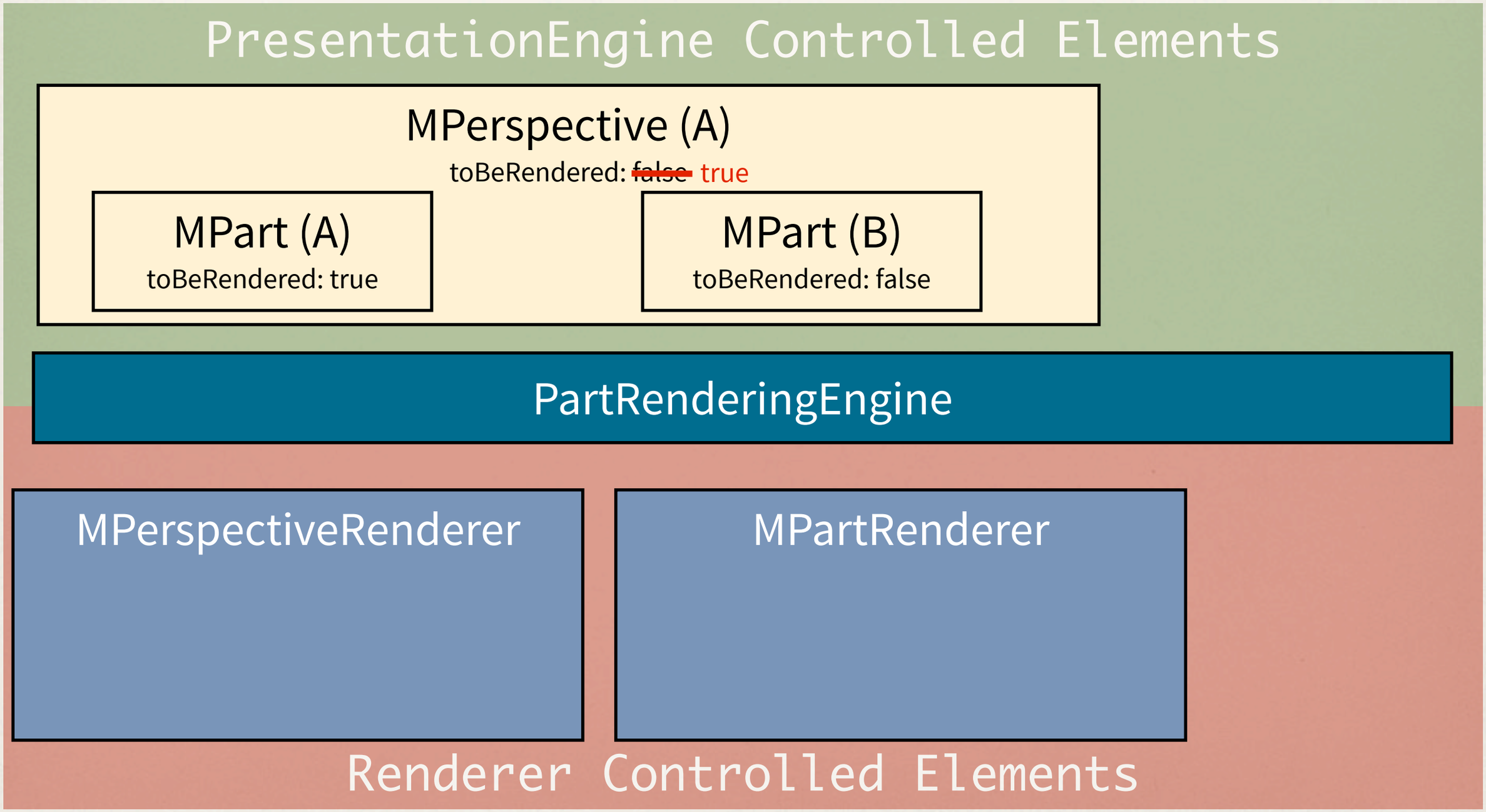
- Go to project simple.swt.renderer

- Fix TODOs in SimpleRenderingEngine
  a) run
  b) createMainWindow
  c) createPart
  d) *implement* handleHeightChanged

Rendering concept



Rendering concept



Rendering concept

PresentationEngine Controlled Elements

MPart (B)
toBeRendered: false

PartRenderingEngine

PerspectiveRenderer

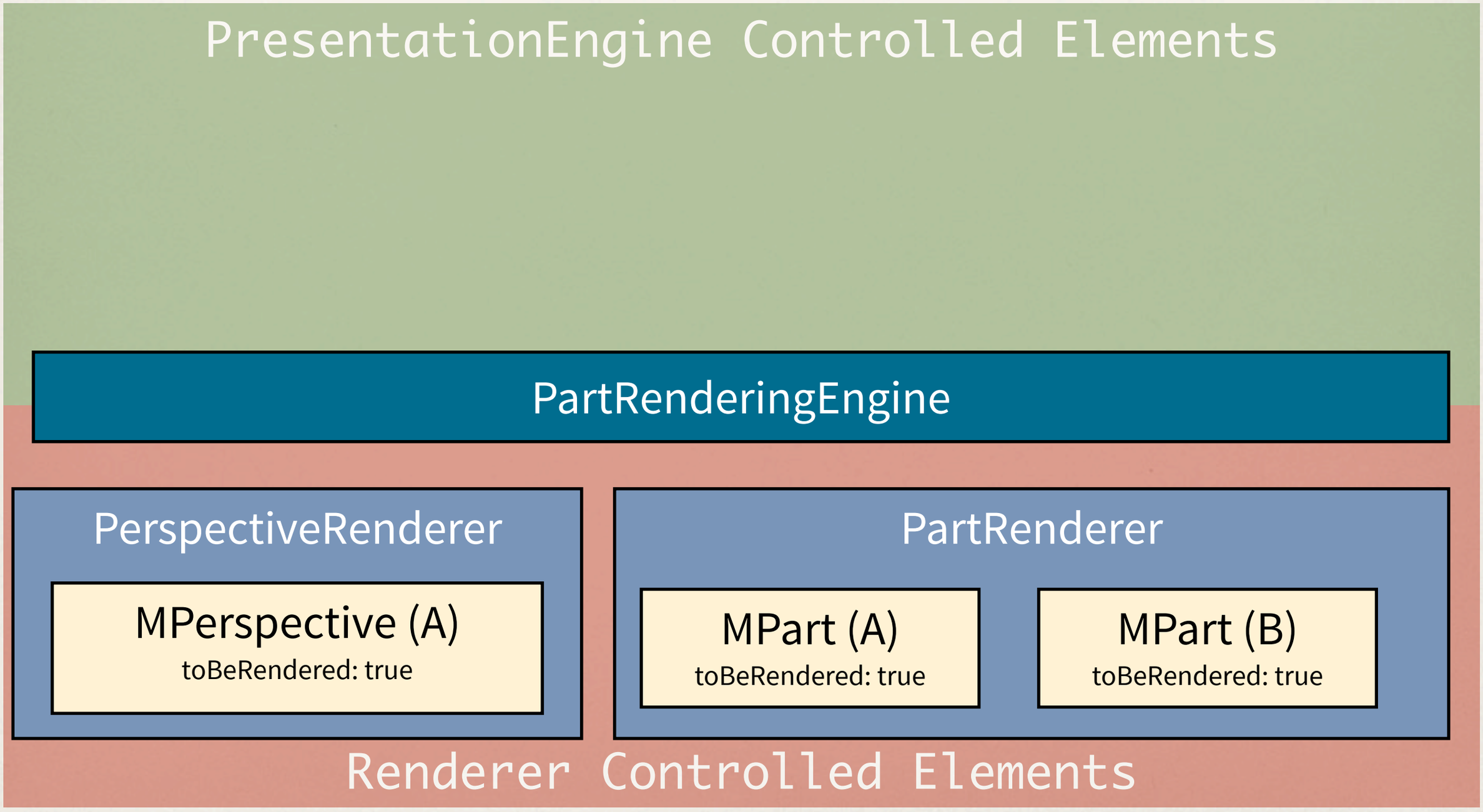
MPerspective (A)
toBeRendered: true

PartRenderer

MPart (A)
toBeRendered: true

Renderer Controlled Elements

Rendering concept



Rendering concept

- PartRenderingEngine is the IPresentationEngine
- AbstractPartRenderer used the create the real widgets and keep them insync while toBeRendered=true
- IRendererFactory used to create renderers
  #getRenderer(MUIElement,Object) ... method to find renderer for element
  - Can be replaced in plugin.xml

```
<property
    name="rendererFactoryUri"
    value="bundleclass://swt.e4.renderer/renderer.CustomRendererFactory">
</property>
```

Rendering concept

- Hands-on
  - Go to project swt.e4.renderer

- Fix TODOs in
  a) FancyTrimWindowRenderer
  b) PGroupContributedPartRenderer
  c) CustomRendererFactory

Off to Sopot

JavaFX Rendering concept

- Similar to SWT-Concept with Engine + Renderers
- Renderers themselves are „widget-toolkit agnostic“
  - Communicate with the „real“ widget through a small facade
  - Use DI for the communication with the widget

JavaFX Rendering concept

Toolkit agnostic

Toolkit specific

JavaFX Rendering concept

Toolkit agnostic

MWindow

SWTWindowRenderer

Toolkit specific

JavaFX Rendering concept

Toolkit agnostic

MWindow

MWindow

WindowRenderer

WWindow

SWTWindowRenderer

FXWindow

SwingWindow

Toolkit specific

JavaFX Rendering concept

```
public interface WWindow<N> extends WWidget<MWindow> {

	public void setMainMenu(WLayoutedWidget<MMenu> menuWidget);
	public void setTopTrim(WLayoutedWidget<MTrimBar> trimBar);
	// ... left, right, bottom

	public void addChild(WLayoutedWidget<MWindowElement> widget);
	public void addChild(int idx, WLayoutedWidget<MWindowElement> widget);
	public void removeChild(WLayoutedWidget<MWindowElement> widget);

	public void show();
	public void close();
}
```

```
public class DefWindowRenderer extends BaseWindowRenderer<Stage> {
	@Override
	protected Class<? extends WWindow<Stage>> getWidgetClass(MWindow window) {
		return WWindowImpl.class;
	 }
}
```

JavaFX Rendering concept

- Hands-on
  - Go to project javafx.e4.renderer
  - Fix TODOs in
    a) CustomRendererFactory
    b) CustomStageRenderer

JavaFX Rendering concept

- Extra Hands-on (extra fun win, mac and linux32)
  - Run development/install-fx.launch and point it to your JDK-Directory
  - Mac: e.g. /Library/Java/JavaVirtualMachines/jdk1.7.0_17.jdk
  - Windows: e.g. C:/Program Files/Java/jdk1.7.0_17
  - Linux: e.g. ~/bin/jdk1.7.0_17

JavaFX Rendering concept

- Extra Hands-on (extra fun win, mac and linux32)

- Run development/install-fxosgi.launch
- Run development/javafx.e4.app.releng.launch
- Run ant javafx.e4.app.releng/build_$os_$arch.xml