

Mother of all Projects

Boo(s)ting application development with Eclipse

Jan Engehausen, Sebastian Hähnel

Mother of all projects

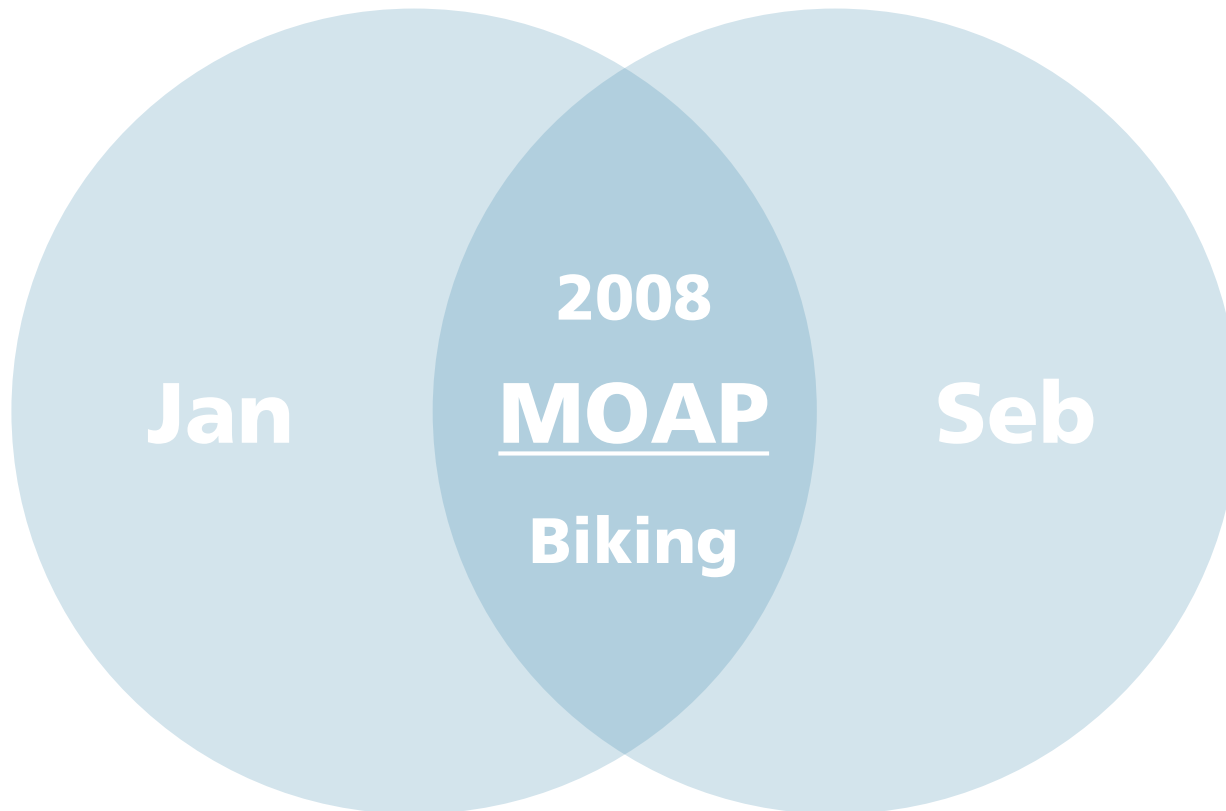


About the presenters

Software Engineers @ UBS

Sebastian Hähnel

Jan Engehausen



Our software developers face a number of challenges



Develop good software

- Development setup
- Clean and secure coding
- Testing and quality assurance

Adhere to standards

- Reference architecture
- Technology governance
- Interaction patterns and their implementation
- Best practices and reusable assets

Integrate with infrastructure

- Continuous build and integration testing
- Release and deployment
- Runtime containers
- Authentication and authorization

This is the *Mother of all Projects*



Our goal is to solve common problems once and share solutions with our developer community!

- We deliver hands-on implementations for common situations since 2009.
- We communicate UBS CTO's reference architectures and technology toolbox to the developers.
- We aim to kick-start application development by *generating a customized component straight into the developers workspace*.
- We work closely with our developer community so that we can continuously improve our examples and see what the developers need.
- We provide technical support for our developers and occasionally set up workshops and training sessions.
- We support UBS CTO with evaluating new technologies and integrating them into our environment.

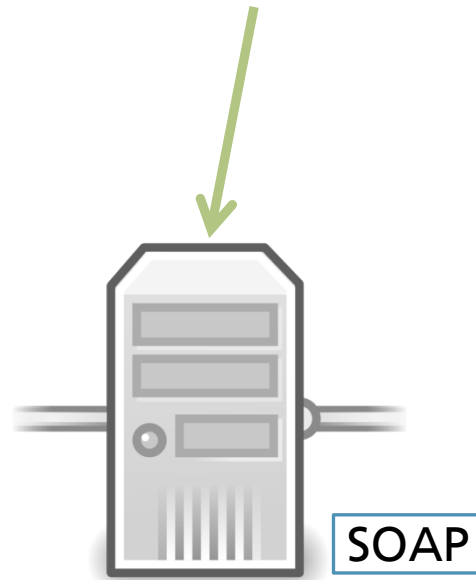
The component templates

Every template implements a reference architecture.

Front-end Template



Web Service Provider Template



Mobile Template



The component templates (continued)

An instantiated template...

- Is a working application
- Implements a number of example use cases
- Demonstrates how to use a number of technologies
- Has a complete build and test setup and is (technically) ready for production
- Contains reference documentation and working guides

Developers use a template...

- To start working on a new component
- As a reference when working on their existing component
- For self-studies

The Eclipse Wizard



The instantiation process

1. In Eclipse, start the wizard for a specific template
2. Select the desired features
3. Type in the new component's name and a few other details
4. Hit "Finish" button

Source: <http://imgur.com/rWu0kFq>

The demo!

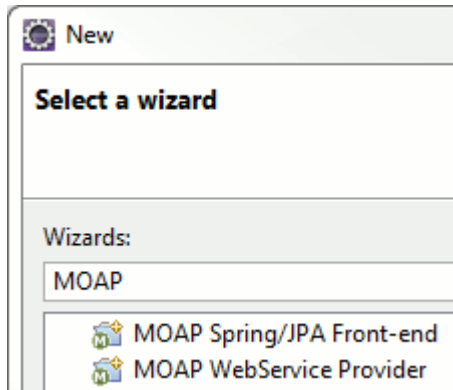
Eclipse Sapphire

The wizard is based on Eclipse Sapphire (<https://www.eclipse.org/sapphire/>)

Sapphire...

- is a UI development framework
- allows developers to focus on the data rather than worry about its presentation
- follows the philosophy of "first the model, then the UI"

Wizard Definition



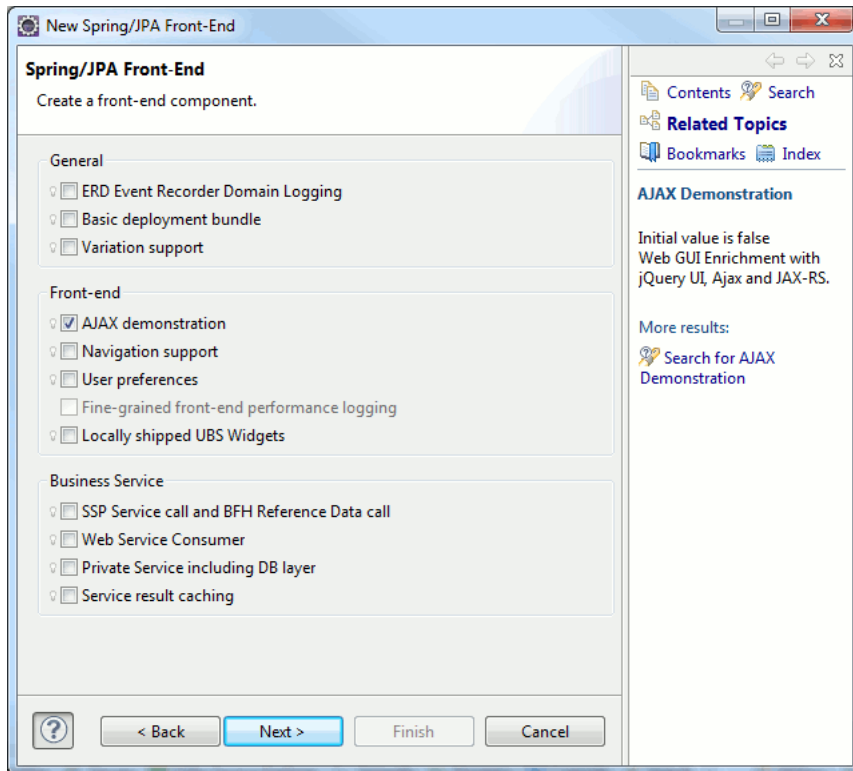
plugin.xml (use Eclipse Extension Point)

```
<plugin>
  <extension point="org.eclipse.ui.newWizards">
    <category name="MOAP Templates" id="MOAP_Category"/>
    <wizard name="MOAP Spring/JPA Front-end" ...>
      <class class="com.foo.wizard.ui.NewJPAMoapWizard">
        <parameter name="wizardTemplate" value="jpa.wizard"/>
        ...
      </class>
    </wizard>
  </extension>
</plugin>
```

Java code (create wizard, load its definition)

```
public class NewJPAMoapWizard extends NewMoapWizard<CreateJPAMoapOp> {
  public final static String ID = "com.foo.wizard.NewJPAMoapWizard";
  public NewJPAMoapWizard() {
    super(modelElement,
      DefinitionLoader
        .context(CreateJPAMoapOp.class)
        .sdef("Moap")
        .wizard("jpa.wizard")
    );
  }
}
```

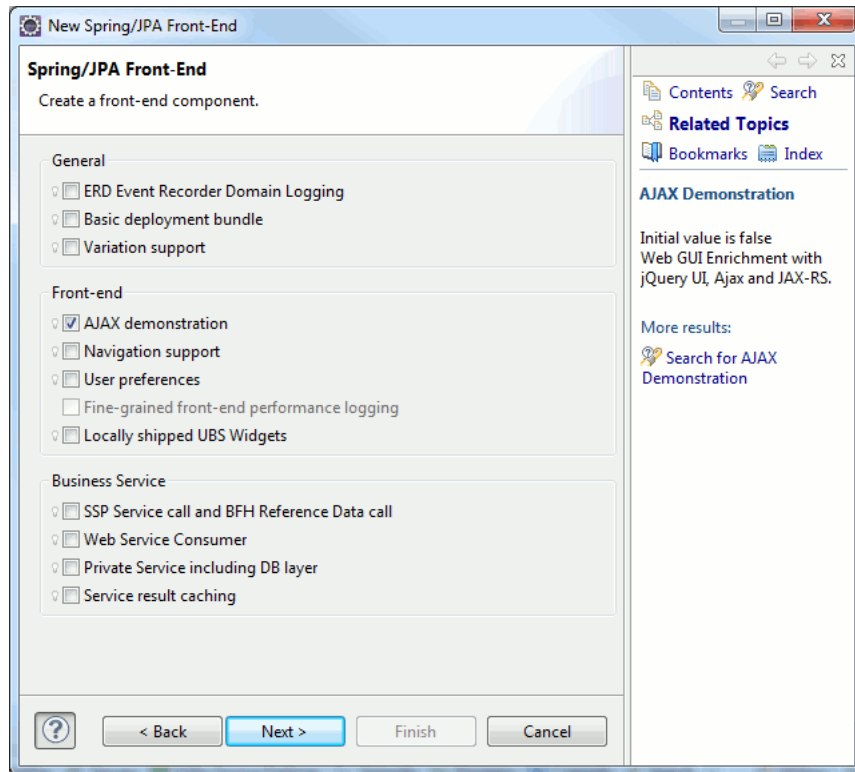
Page Definition



Moap.sdef (Sapphire definition)

```
<definition>
...
<wizard>
  <id>jpa.wizard</id>
  <element-type>CreateJPAMoapOp</element-type>
  <label>New Spring/JPA Front-end</label>
</page>
<page>
  <id>jpa.options</id>
  <label>Spring/JPA Front-end</label>
  ...
<content>
  <with>
    <path>JPATemplateOptions</path>
  </with>
  <case>
    <content>
      <include>jpa.template.options</include>
    </content>
  </case>
  ...
</content>
</definition>
```

Property Definition



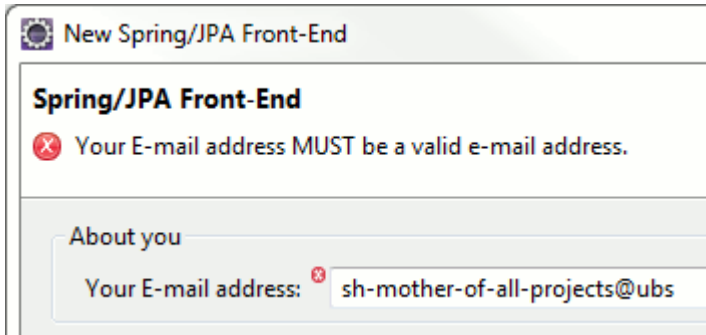
Moap.sdef (Sapphire definition)

```
...  
<composite>  
  <id>jpa.template.options</id>  
  <content>  
...  
  <group>  
    <label>Front-end</label>  
    <content>  
      <property-editor>  
        <property>Ajax</property>  
      </property-editor>  
...  
  ...
```

Java code (define property)

```
public interface TemplateBase extends Element { ...  
    @Fact(statement = "Web GUI Enrichment with jQuery UI, Ajax and JAX-RS.")  
    @Label(standard = "AJAX demonstration")  
    @Type(base = Boolean.class)  
    @InitialValue(text = "false")  
    ValueProperty PROP_AJAX = new ValueProperty(TYPE, "Ajax");
```

Property Validation Rules



New Spring/JPA Front-End

Spring/JPA Front-End

✖ Your E-mail address MUST be a valid e-mail address.

About you

Your E-mail address: ✖ sh-mother-of-all-projects@ubs

Java code (*define property*)

```
public interface TemplateBase extends Element {
    ...
    @Fact(statement = "Enter your E-mail address; must not be empty.")
    @Required
    @Label(standard = "Your E-mail address")
    @InitialValue(text = Constants.YOUR_EMAIL)
    @Service(impl = EmailValidationService.class)
    ValueProperty PROP_EMAIL_ADDRESS =
        new ValueProperty(TYPE, "emailAddress");
}
```

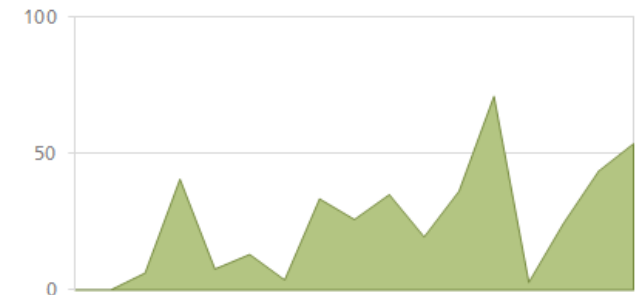
Java code (*validate property value*)

```
public class EmailValidationService extends ValidationService {
    private static final String eMailRegExp = "[\\w\\.%+-]+@[\\w-]+\\.([\\w\\.]+)";
    public Status compute() {
        final Value<?> value = context(Value.class);
        final String text = value.text();
        if (text != null && !text.matches(eMailRegExp)) {
            ...
            return Status.createErrorStatus(msg);
        } else {
            return Status.createOkStatus();
        }
    }
}
```

The Wizard: Conclusion

- The application developer can choose between command line template instantiation and the wizard.

We got good feedback on the wizard and see that developers use it often.



Adoption of the wizard since its creation in summer 2013

- The Eclipse Sapphire experience
 - We had little knowledge about developing Eclipse plugins.
 - Developing the wizard took only a few weeks. It was easy!
- The wizard has saved us a lot of time during our manual tests before a release.

Contact information

sh-mother-of-all-projects@ubs.com
UBS AG

UBS Intranet
<http://goto/it-moap>

UBS AG
Postfach
8098 Zürich

www.ubs.com

