On the shoulders of giants

Harnessing the Power of Eclipse Enterprise Ecosystem

amdocs Kirill Grouchnikov

What is he going to talk about?

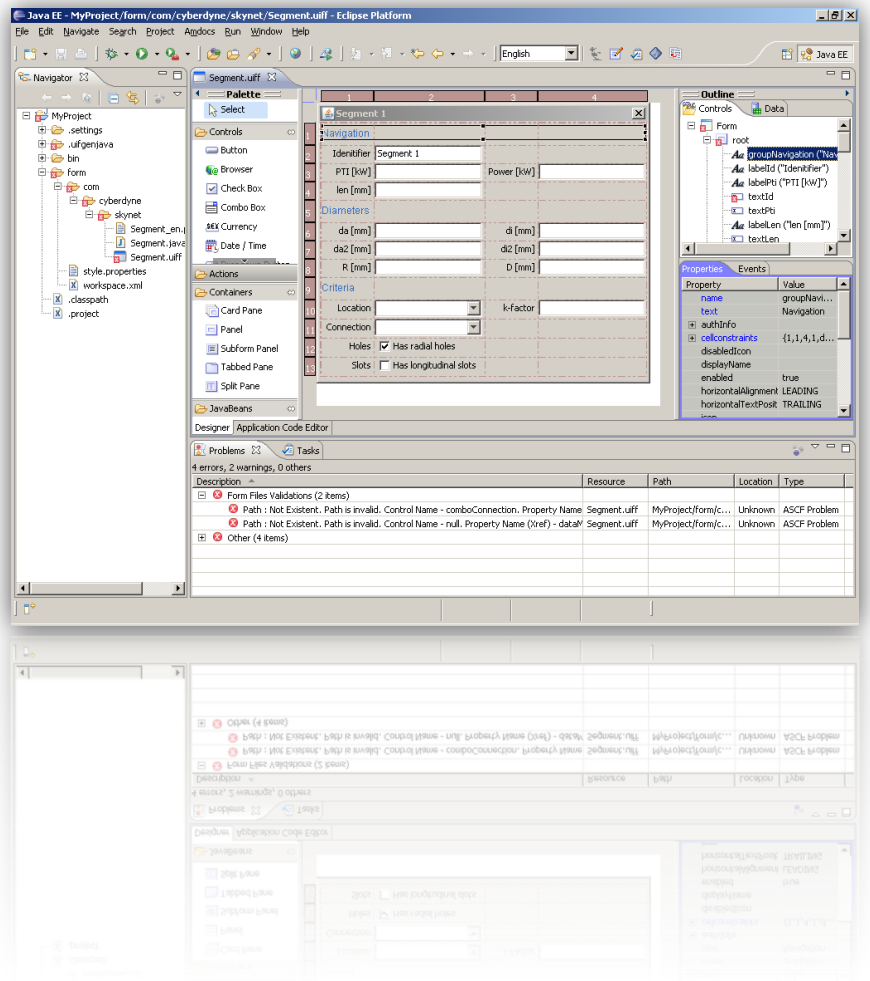
To show you how Eclipse makes you more productive

Why should I care?

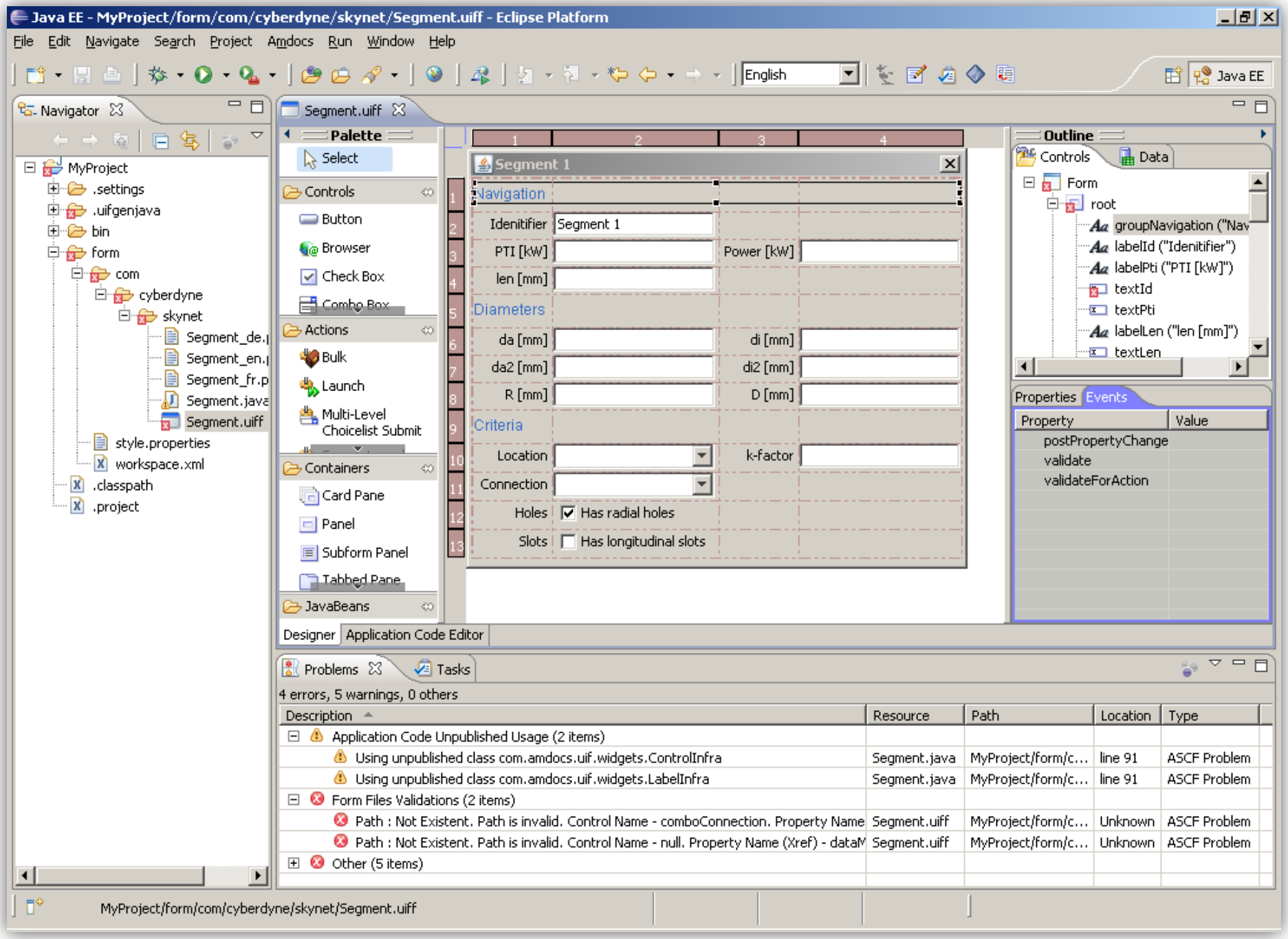
Less code to write, test and support

And then you can retire

What is he going to talk about?

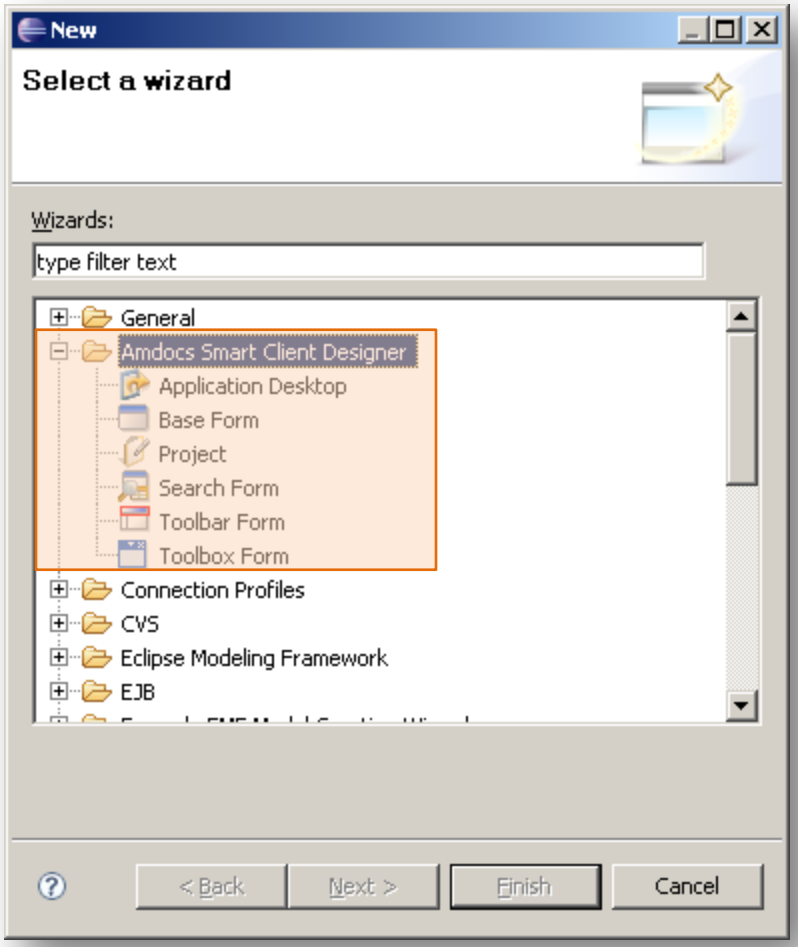


Amdocs Smart Client Designer

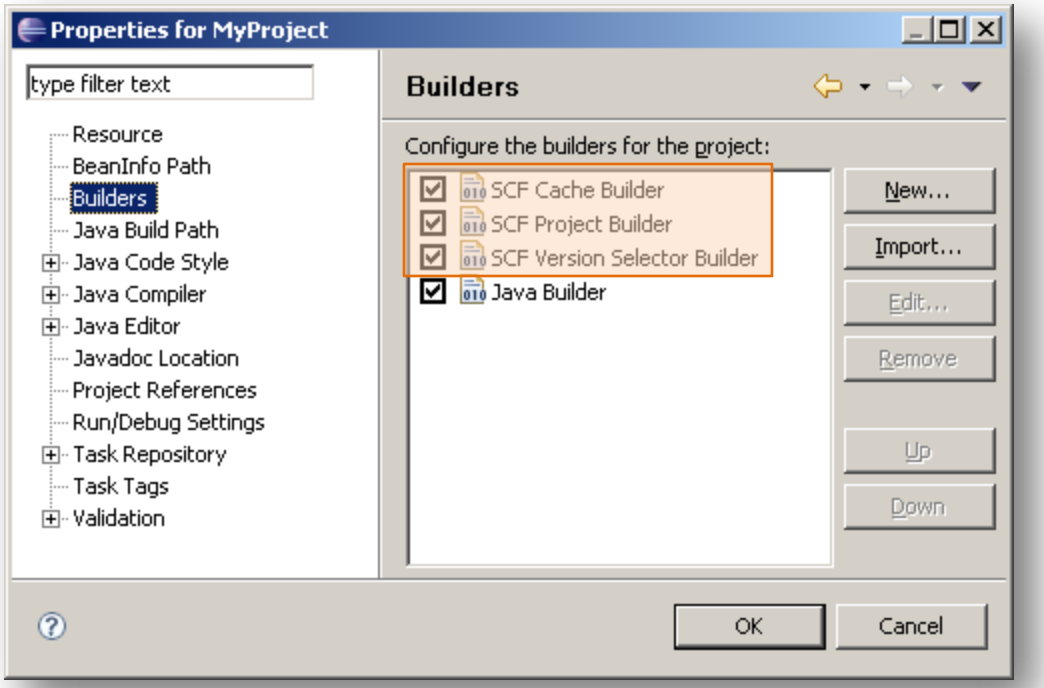

Java EE - MyProject/form/com/cyberdyne/skynet/Segment.uiff - Eclipse Platform
File Edit Navigate Search Project Amdocs Run Window Help
English
Java EE
Navigator
MyProject
.settings
.uifgenjava
bin
form
com
cyberdyne
skynet
Segment_de.
Segment_en.
Segment_fr.p
Segment.java
Segment.uiff
style.properties
workspace.xml
.classpath
.project
Segment.uiff
Palette
Select
Controls
Button
Browser
Check Box
Combo Box
Actions
Bulk
Launch
Multi-Level Choicelist Submit
Containers
Card Pane
Panel
Subform Panel
Tabbed Pane
JavaBeans
Segment 1
Navigation
Idenitifier
Segment 1
PTI [kW]
Power [kW]
len [mm]
Diameters
da [mm]
di [mm]
da2 [mm]
di2 [mm]
R [mm]
D [mm]
Criteria
Location
k-factor
Connection
Holes
Has radial holes
Slots
Has longitudinal slots
Designer
Application Code Editor
Outline
Controls
Data
Form
root
groupNavigation ("Nav
labelId ("Idenitifier")
labelPti ("PTI [kW]")
textId
textPti
labelLen ("len [mm]")
textLen
Properties
Events
Property
Value
postPropertyChange
validate
validateForAction
Problems
Tasks
4 errors, 5 warnings, 0 others
Description
Resource
Path
Location
Type
Application Code Unpublished Usage (2 items)
Using unpublished class com.amdocs.uif.widgets.ControlInfra
Using unpublished class com.amdocs.uif.widgets.LabelInfra
Form Files Validations (2 items)
Path : Not Existent. Path is invalid. Control Name - comboConnection. Property Name
Path : Not Existent. Path is invalid. Control Name - null. Property Name (Xref) - dataM
Other (5 items)
Segment.java
MyProject/form/c...
line 91
ASCF Problem
Segment.uiff
Unknown
MyProject/form/com/cyberdyne/skynet/Segment.uiff

Provides familiar experience

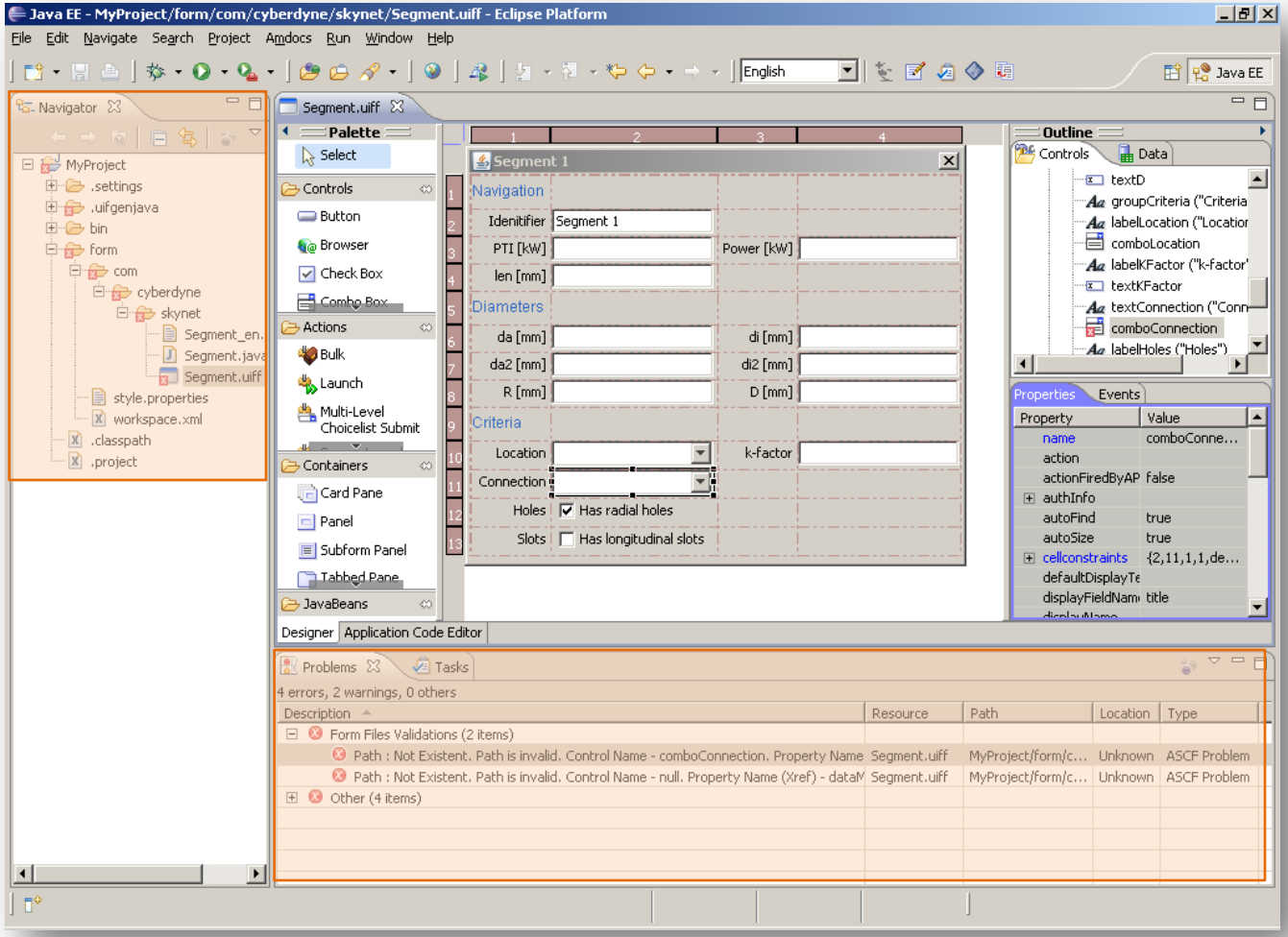
Immersed in Eclipse



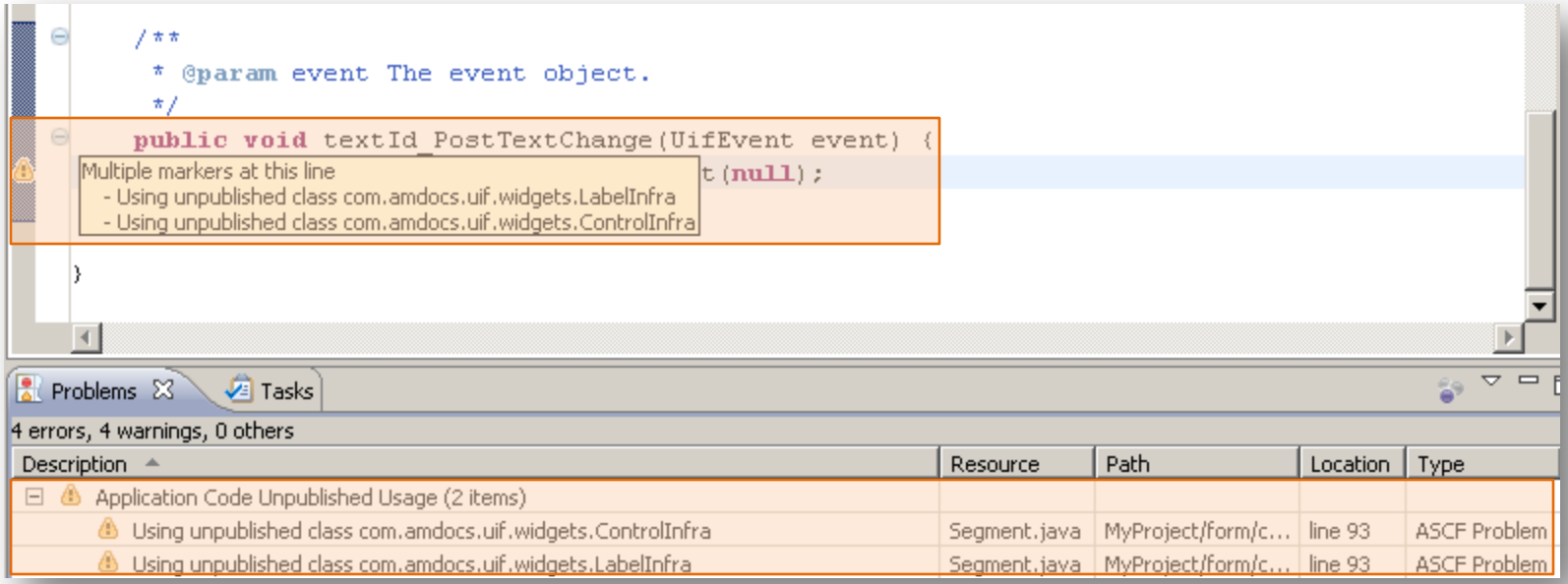
Immersed in Eclipse



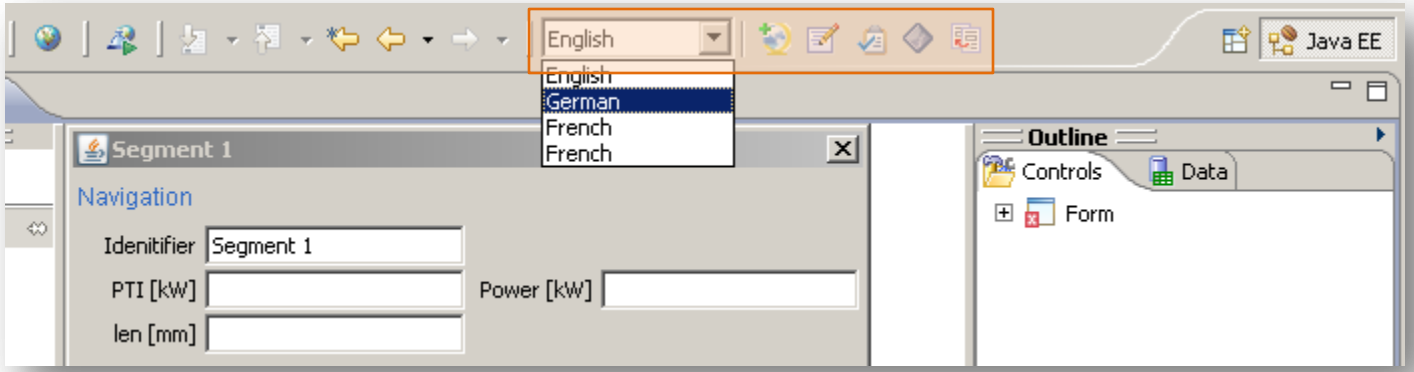
Immersed in Eclipse



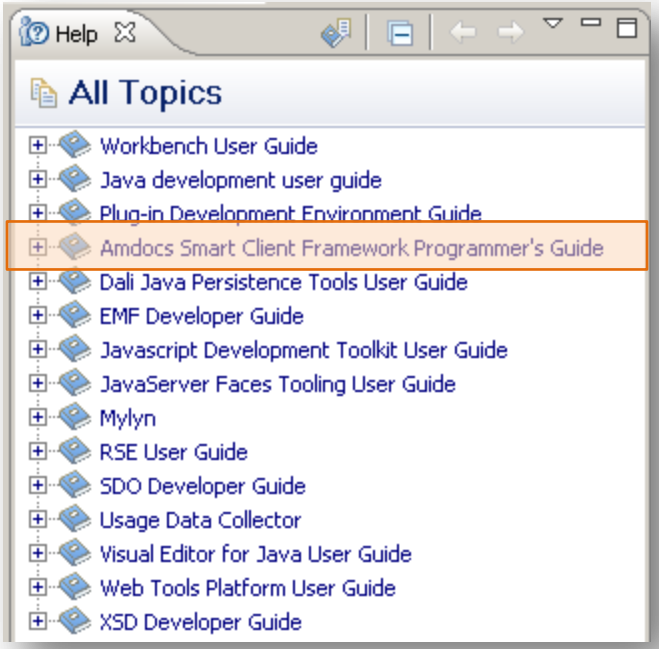
Immersed in Eclipse



Immersed in Eclipse



Immersed in Eclipse



For your users

Provides familiar experience

For your developers

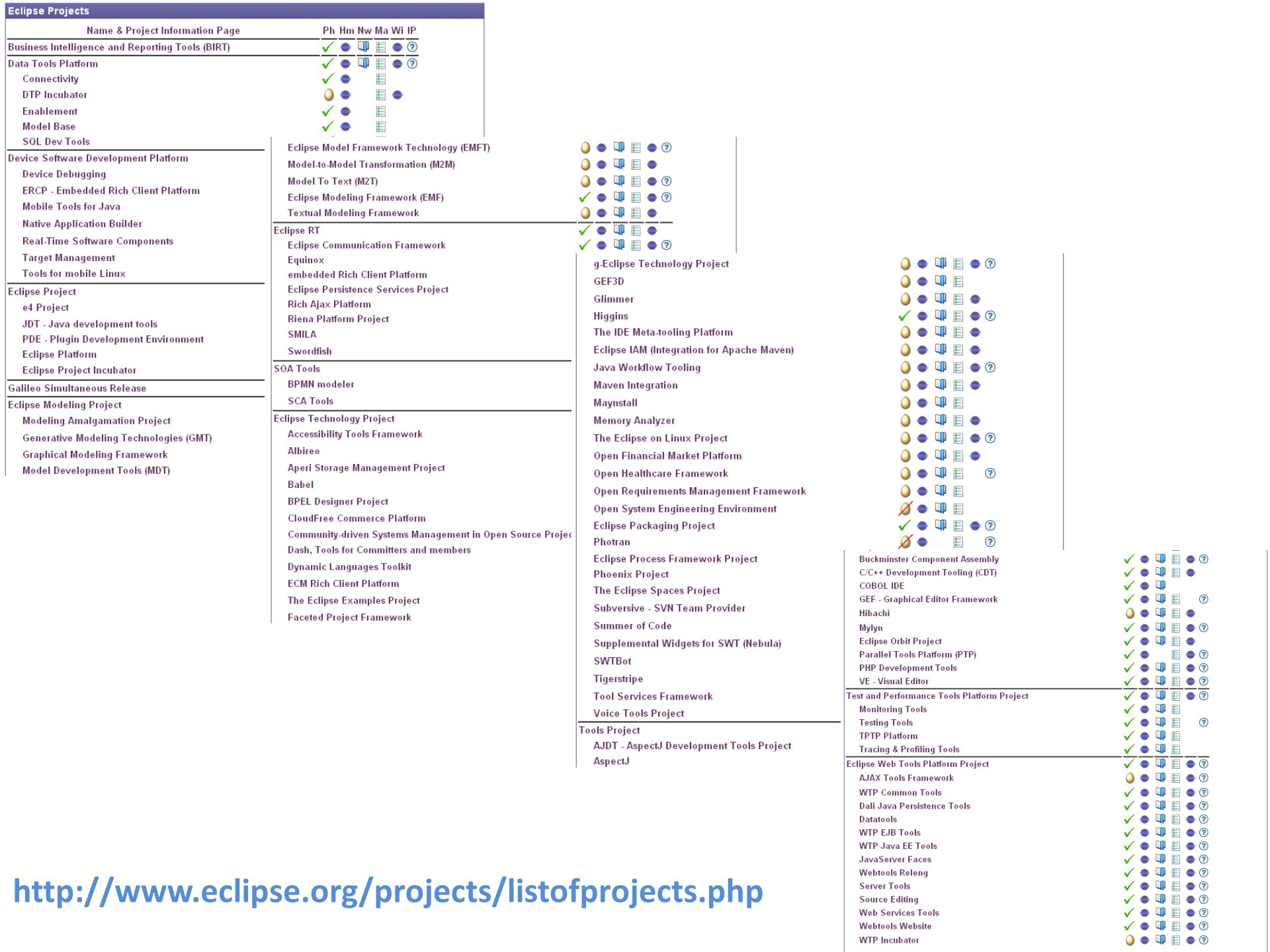
Less code to write, test and support

Especially for enterprise solutions

Eclipse – rich foundation


GMT
DTP
EMF
ECF
WTP
VE
GEF
RCP
JDT
AJDT
TPTP
WST
JEM
RAP

</acronyms>

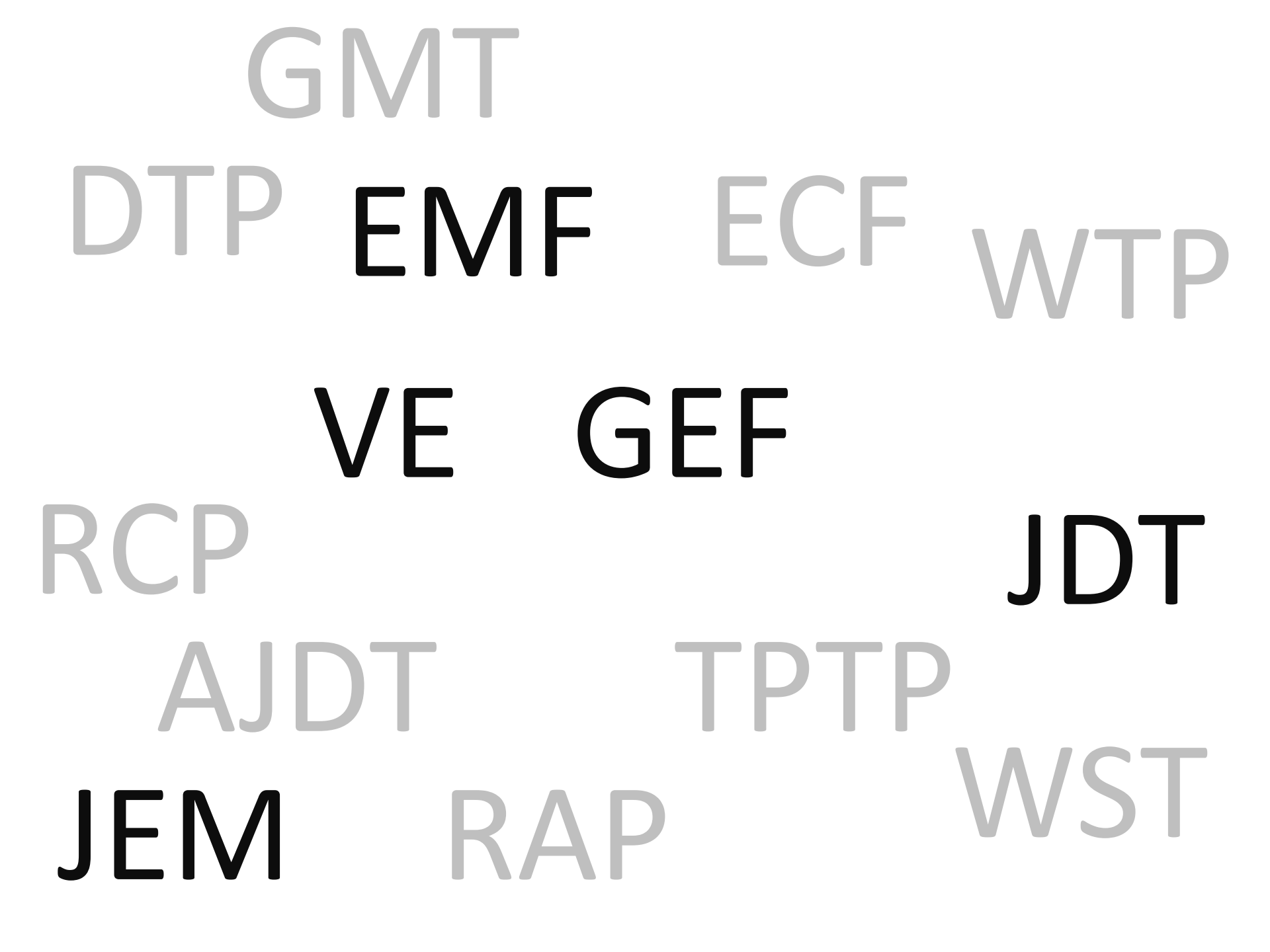


http://www.eclipse.org/projects/listofprojects.php

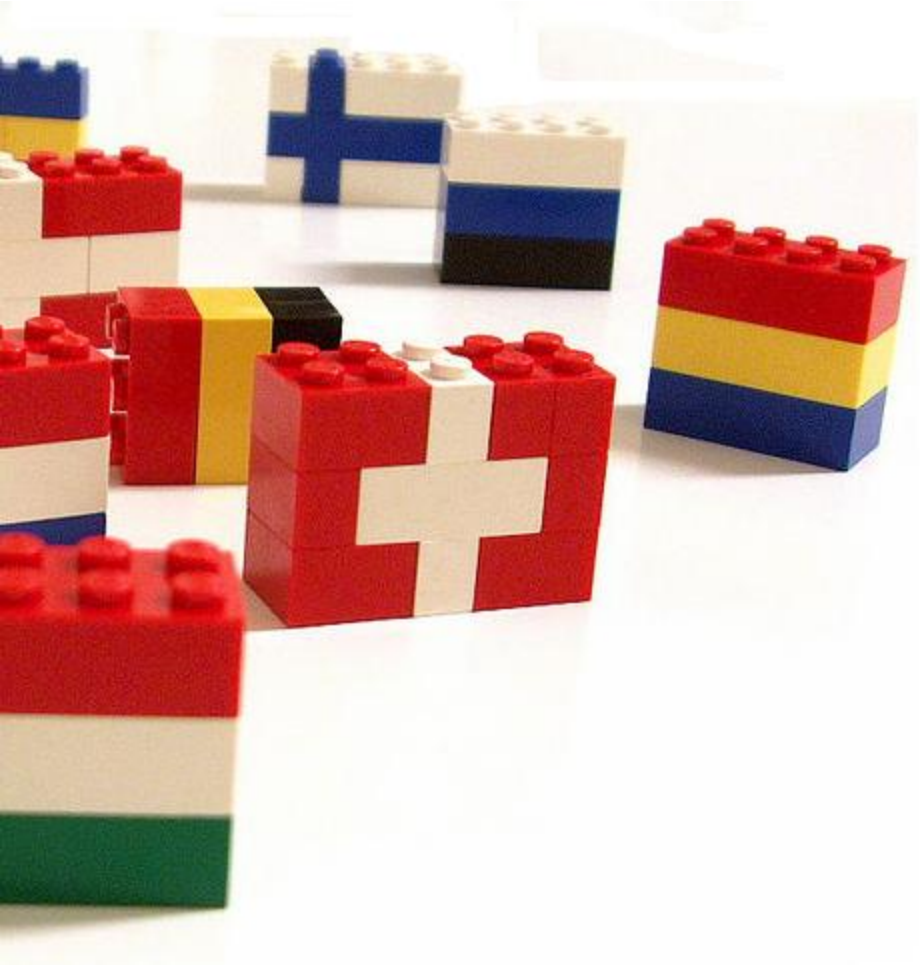
Community health - example

http://www.eclipse.org/modeling/emf/project-info/team.php

- Open Canarias
- RedHat
- Zeligsoft
- Puzzle ITC
- Geensys
- Embarcadero
- Okidoo
- IBM
- Oracle
- ES-Computersysteme
- Elver
- FernUniversitaet
- University of Brock


GMT
DTP
EMF
ECF
WTP
VE
GEF
RCP
JDT
AJDT
TPTP
WST
JEM
RAP

Our building blocks



- EMF – persistence
- JEM / VE – UI builder
- GEF – editors
- JDT – Java source trees

Putting the pixels on canvas

XML

↓ EMF

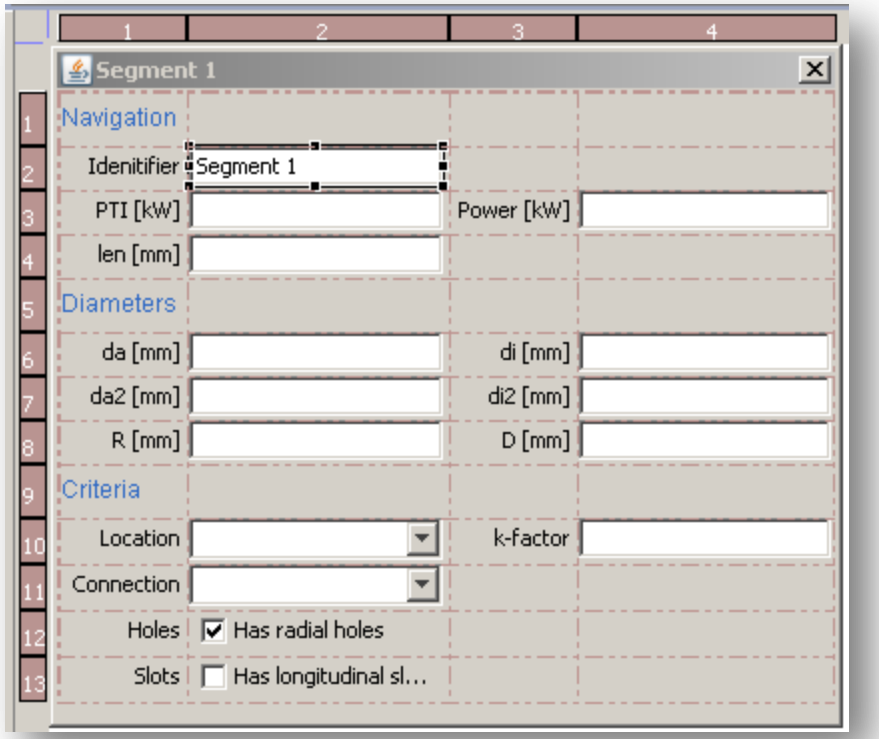
Java model

↓ JEM

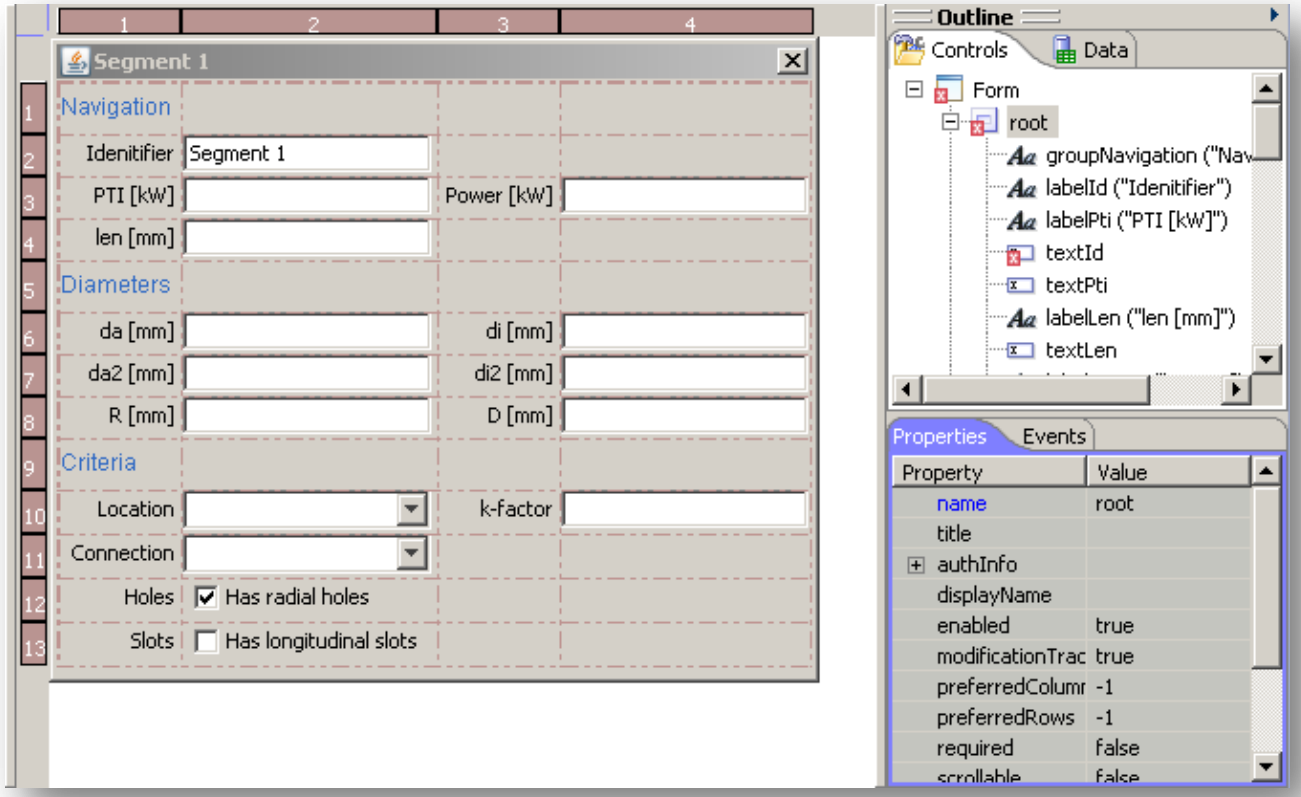
Swing frame

↓ GEF

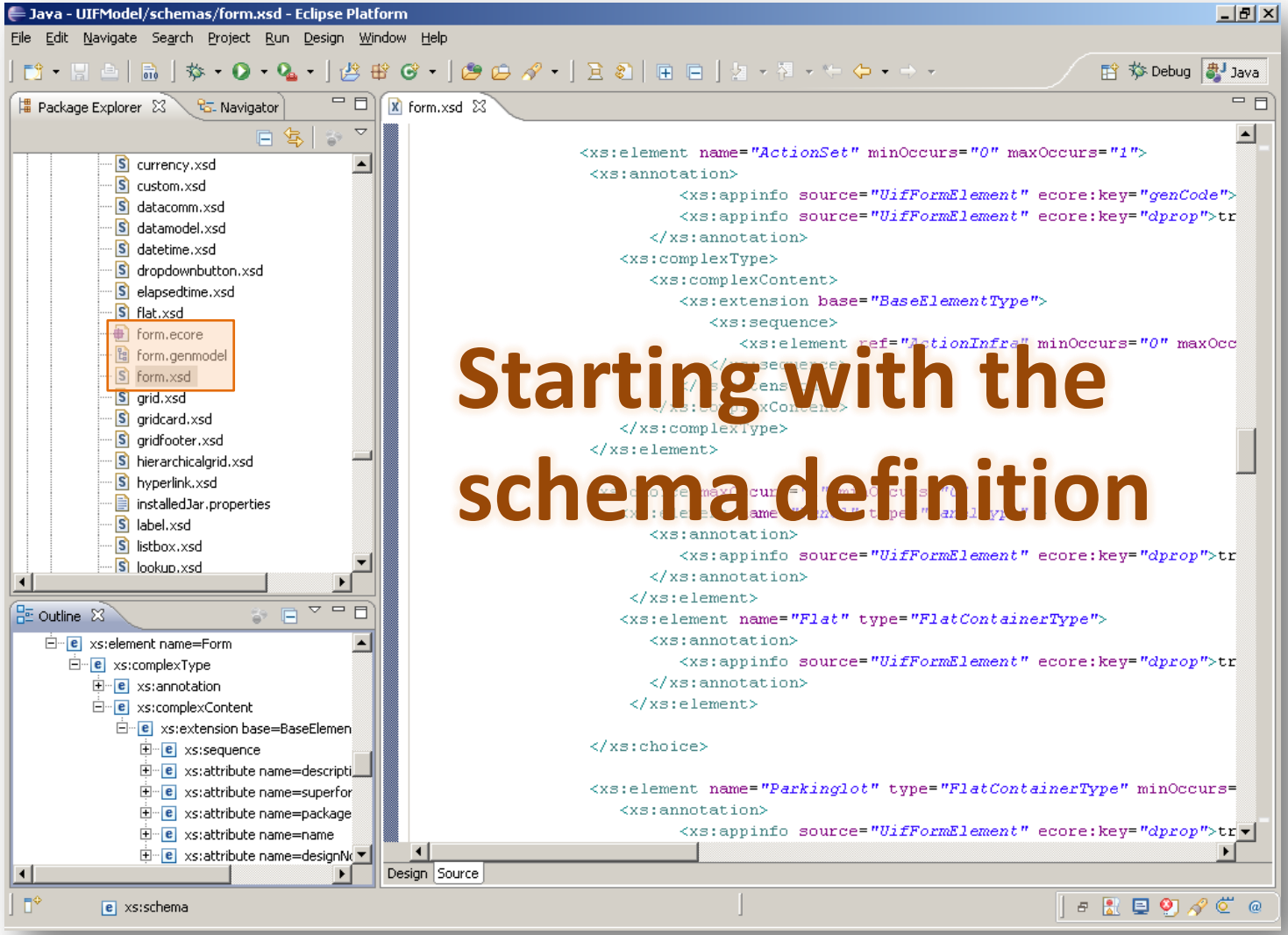
Canvas



EMF as the persistence layer

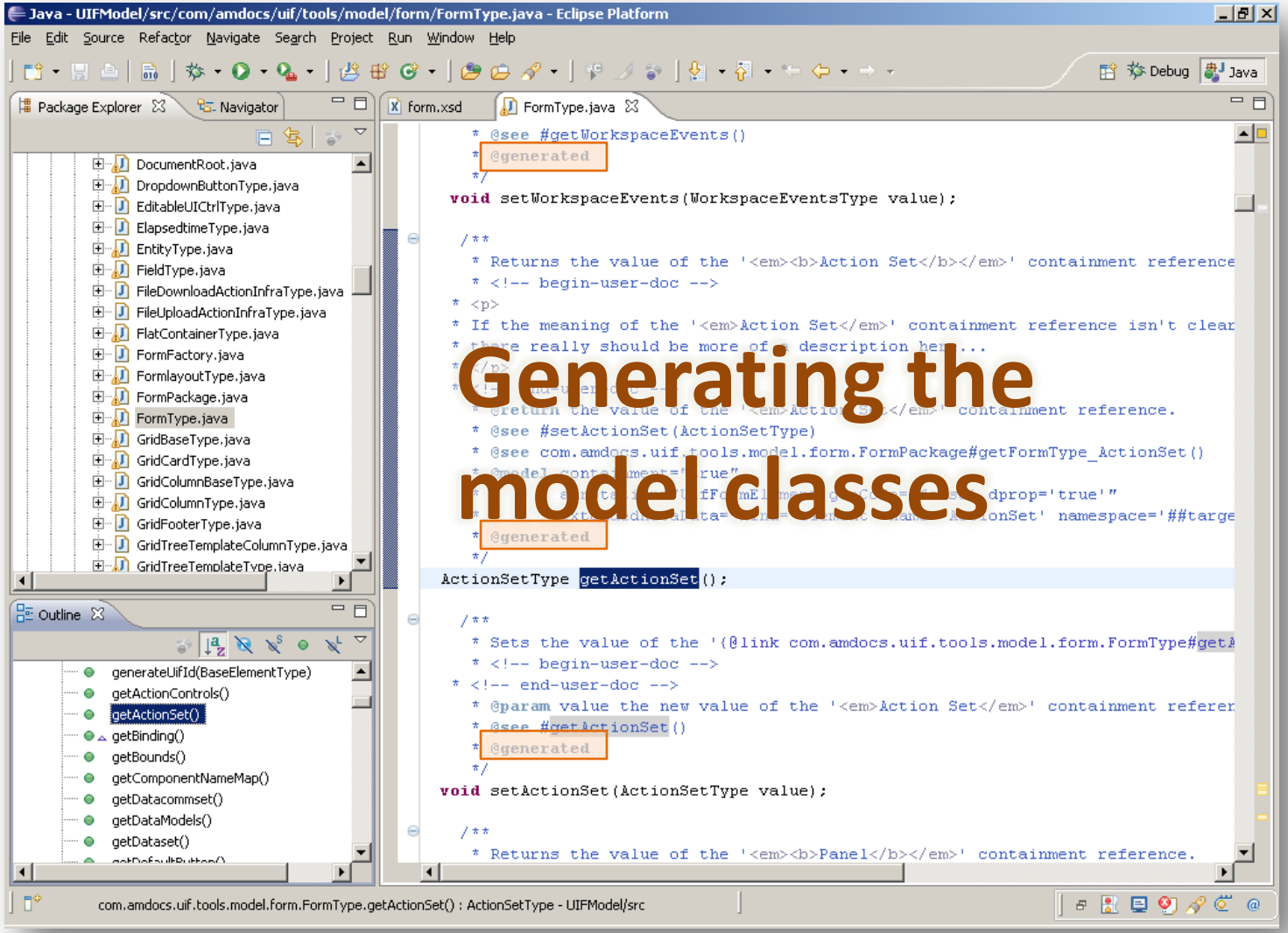


EMF as the persistence layer



Starting with the schema definition

EMF as the persistence layer



Generating the model classes

EMF as the persistence layer

Segment.uiff

```
<?xml version="1.0" encoding="UTF-8"?>
<Form xmlns="http://" uifid="Form" bounds="0,
  <WorkspaceEvents uifid="uifId27123309482240
  <ActionSet uifid="uifId11233094090535"/>
  <Panel uifid="uifId21233094090535" name="ro
    <AuthInfo uifid="uifId812330941709 8"/>
    <FormLayout uifid="uifId3123309409054 5" c
```

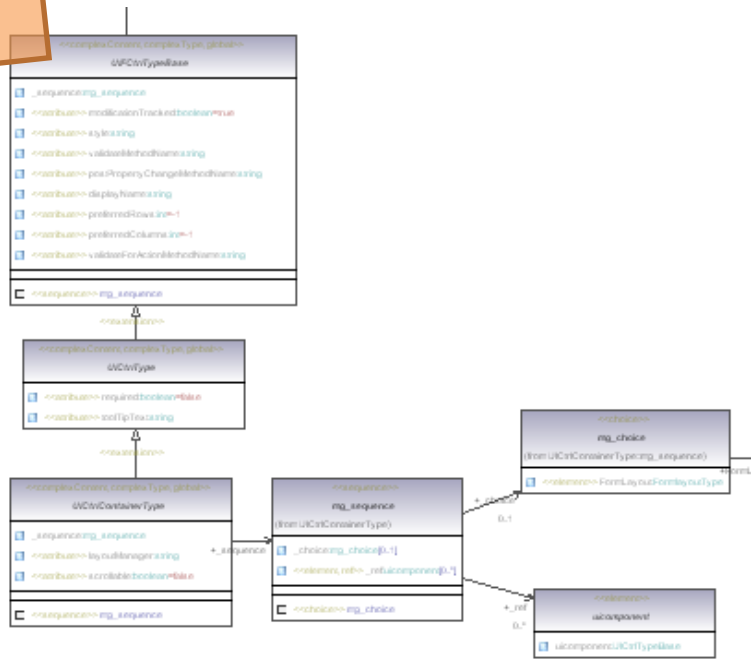
```
Resource formModelResource = null;
ResourceSet resourceSet = new ResourceSetImpl();
resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()
        .put(Resource.Factory.Registry.DEFAULT_EXTENSION,
                new FormResourceFactoryImpl());

resourceSet.getPackageRegistry().put(FormPackage.eNS_URI,
        FormPackage.eINSTANCE);

try {
    URI inputURI = URI.createPlatformResourceURI(modelFile
            .getFullPath().toString(), false);
    formModelResource = resourceSet.createResource(inputURI);
    formModelResource.load(getLoadOptions());

} catch (Exception e) {
    e.printStackTrace();
}
return formModelResource;
```

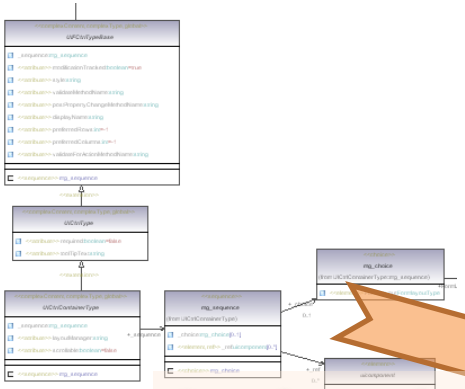
xml -> java model



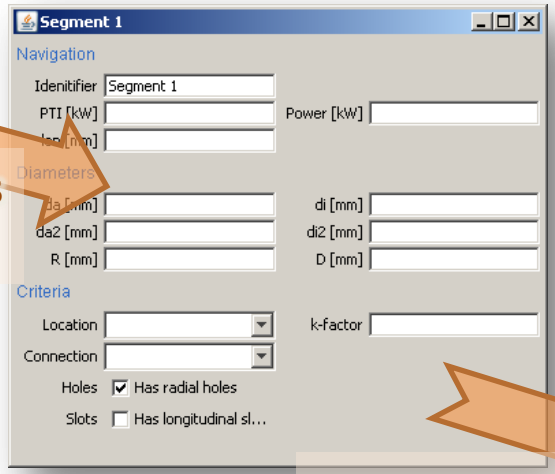
xml ↔ java
notifications
validations

JEM proxies

Java model



Offscreen Swing JFrame



Creation and updates via JEM proxies

GEF graphical edit parts

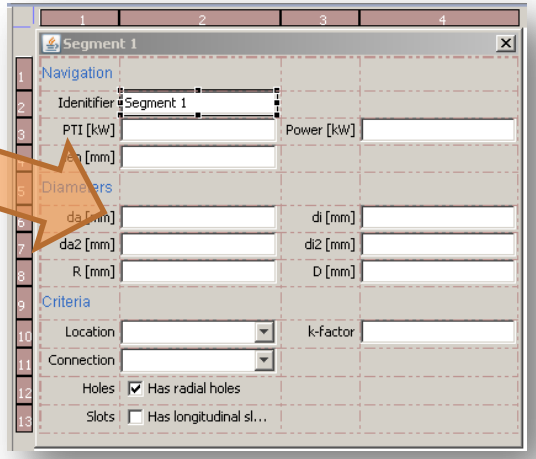
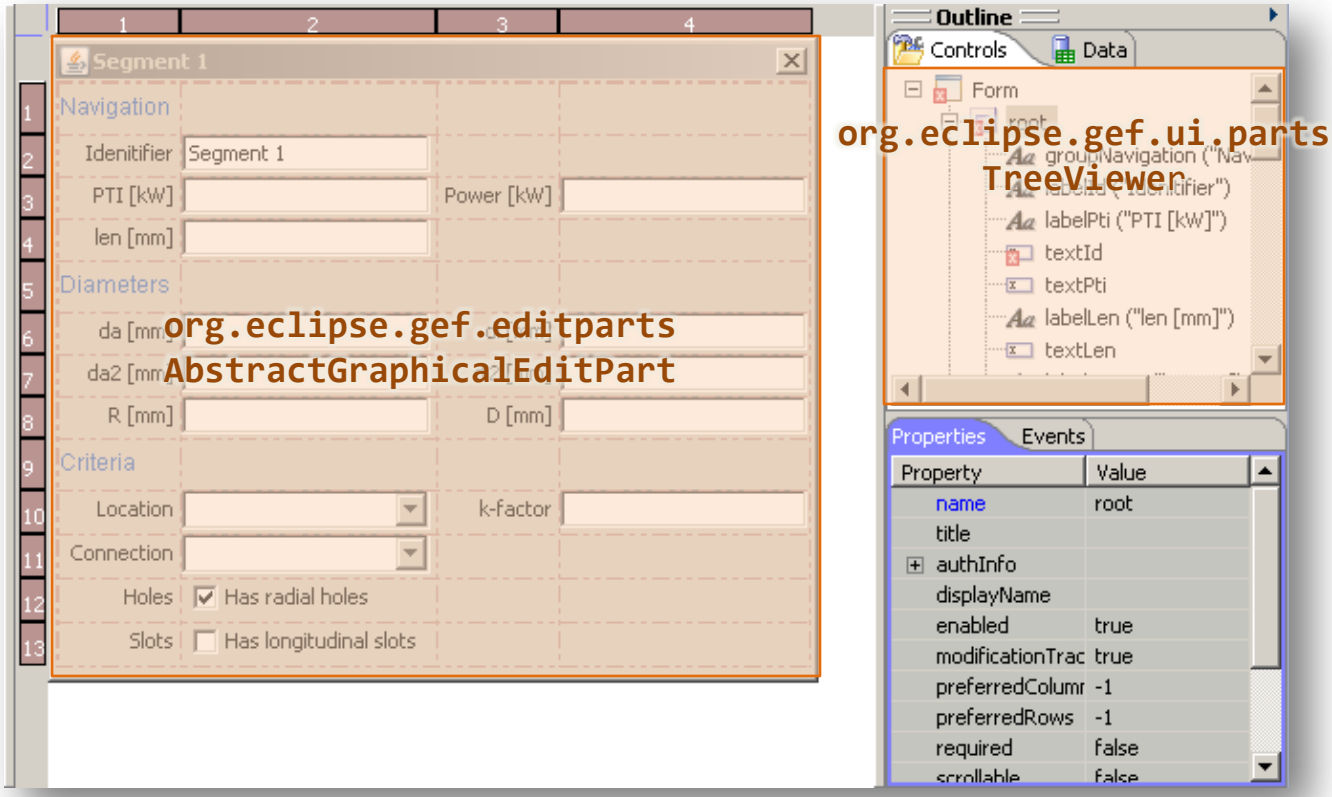


image changes

GEF - everything is an *EditPart*



All together now

XML

↓ EMF

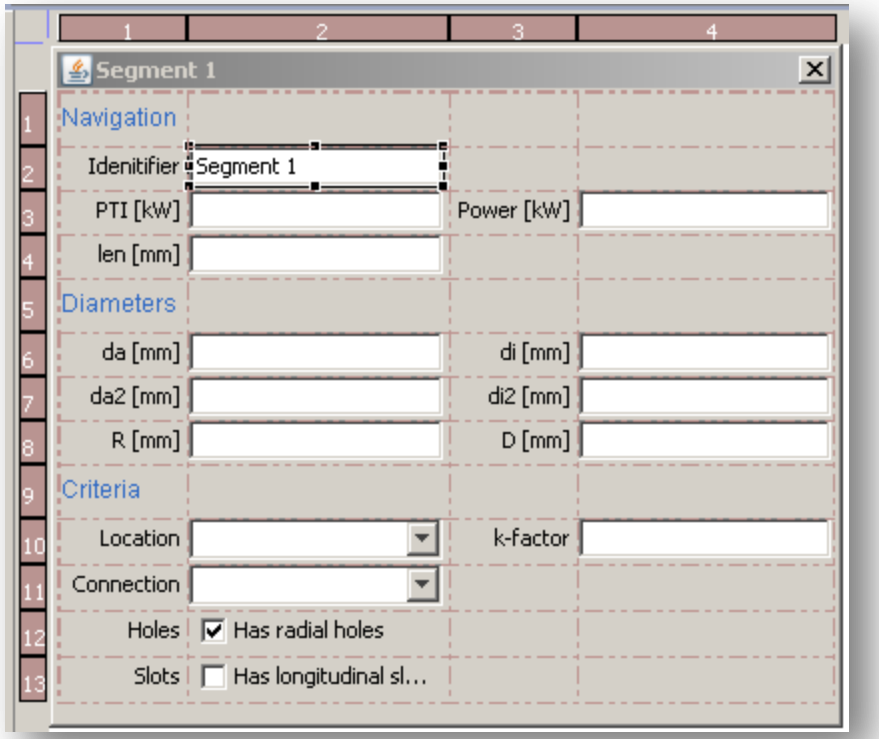
Java model

↓ JEM

Swing frame

↓ GEF

Canvas



Java-based warnings

Java class change

↓ core

Builder invocation

↓ JDT

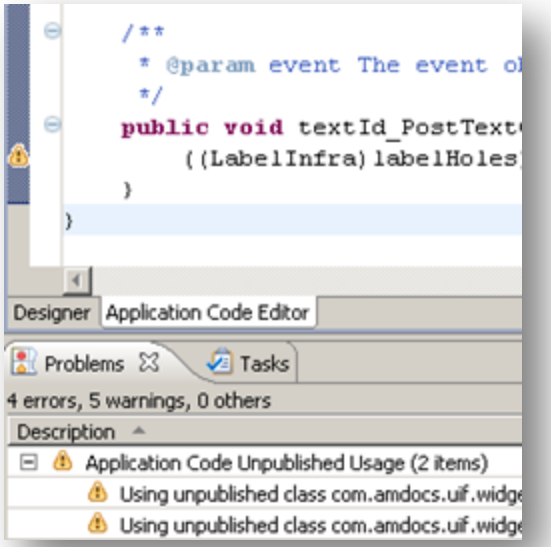
Java syntax trees

↓

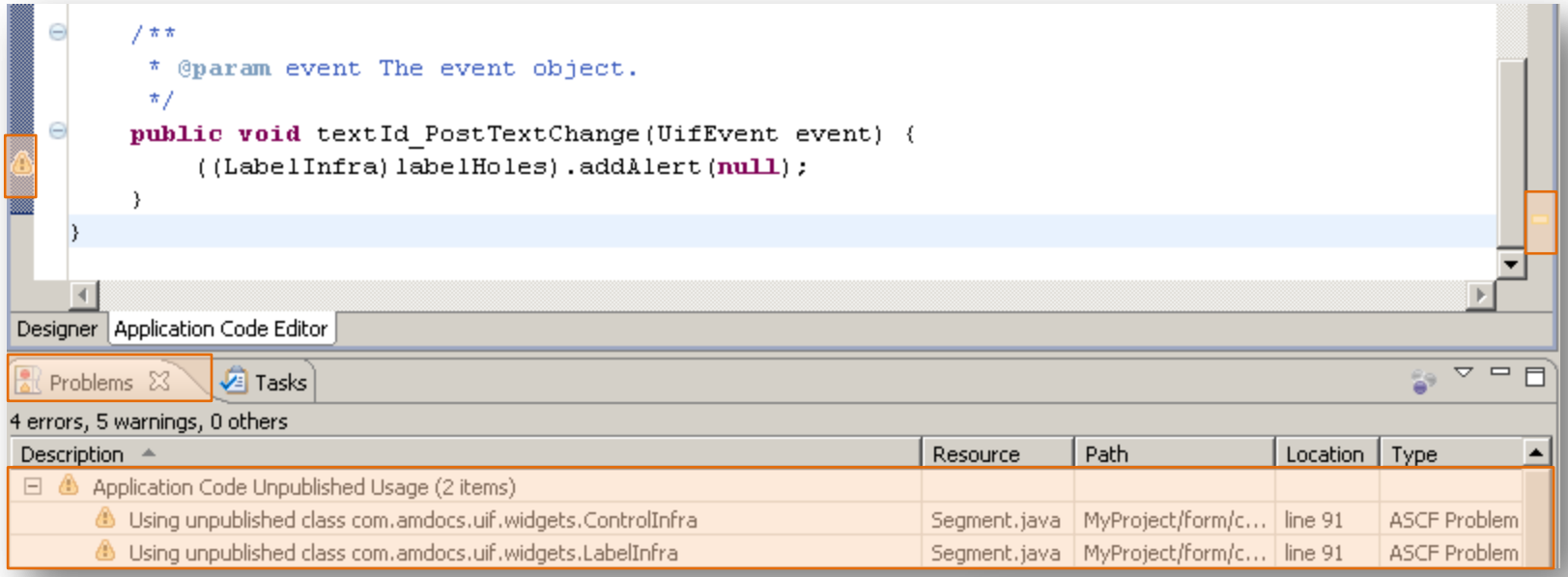
Marker creation

↓ core

UI update

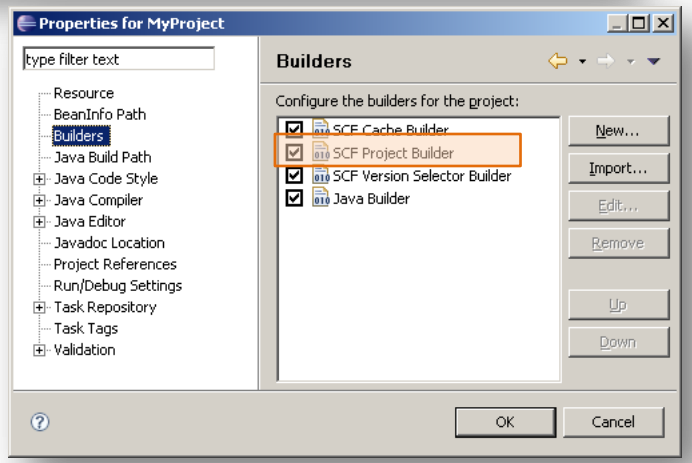


Flagging API usage violations



org.eclipse.jdt.core.dom.ASTParser

```
public void createASTs(ICompilationUnit[] compilationUnits,
                       String[] bindingKeys,
                       ASTRequestor requestor,
                       IProgressMonitor monitor)
```



All together now

Java class change

↓ core

Builder invocation

↓ JDT

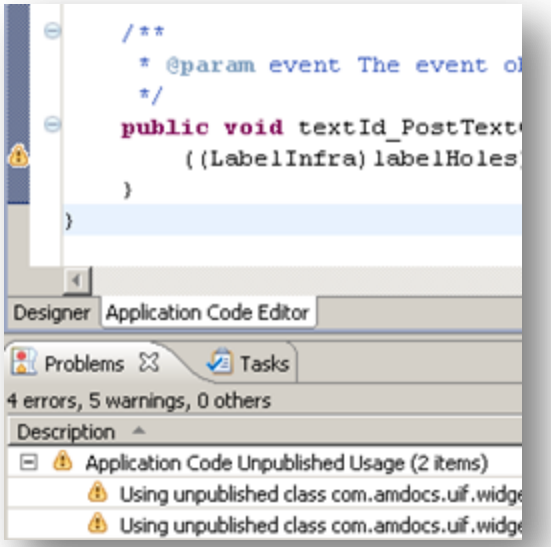
Java syntax trees

↓

Marker creation

↓ core

UI update



And those are just the big pieces


markers
builders
actions
monitors
natures
commands
wizards
help
dialogs
validations
views
jobs
editors
perspectives


markers
builders
actions
monitors
natures
commands
wizards
help
dialogs
validations
views
jobs
editors
perspectives

Asynchronous

Background

Queuing

Cancellable

Progress

big pieces

+

small pieces

=

the Eclipse way

The only rule of Eclipse way

don’t fight it

=

provide familiar experience to your users

Our experience

- Reuse as much plumbing as possible
- Reuse as many UI parts as possible
- Use very few synchronous UI operations
- Reuse as many UI flows as possible
- Decompose into plugins
- Build for extensibility
- Don’t depend on internal code

</dudeTalking>

Q & A

http://www.eclipse.org/documentation/

kirillg@amdocs.com

www.pushing-pixels.org

Thank you

Image credit: **cemre** at http://flickr.com/photos/f/637025/ under NC-SA 2.0