What did we do to solve the problem?

How to create alternate presentations

1. Presentation plugins & APIs
2. Optimize it to work with standard Eclipse plugins

Why worry about the End User UI?

Gotchas. Things to know.
Why worry about the End User UI?

What did we do to solve the problem?

1. Presentation plugins & APIs
2. Optimize it to work with standard Eclipse plugins

How to create alternate presentations

Gotchas. Things to know.
This looks cool to me!!
So many features, this is really powerful.

But where do I even start to work?
How does this solve my problem??

Developers

Non-Developers
Eclipse UIs are inherently complex
He also points out the obvious problem with Eclipse-based process modelling tools: they're not friendly to the business types.
Dated look and Feel.

We’re not in 2007 anymore
What did we do to solve the problem?

10 Minutes

Why worry about the End User UI?

1. Presentation plugins & APIs
2. Optimize it to work with standard Eclipse plugins

How to create alternate presentations

Text

1. Presentation plugins & APIs
2. Optimize it to work with standard Eclipse plugins

Gotchas. Things to know.
Eclipse Based product

Non-Dev Usable UI

Plug-ins in SDK

Presentation Layer
Quick Demo Business UI...
What did we do to solve the problem?

Why worry about the End User UI?

How to create alternate presentations

1. Presentation plugins & APIs
2. Optimize it to work with standard Eclipse plugins

Gotchas. Things to know.
Lesson Learned 1: Capabilities
The cheap way of creating targeted UIs.
Capabilities:

- Prompt the user to enable Capabilities
- Backward Compatibility
- Teamworks Advanced Features
- Teamworks Developer Capability
  - Advanced Data Modeling
  - Advanced Integration
  - Advanced Process Modeling
  - Debug
  - Form Design
  - Pre & Post Assignments
  - Service Modeling
  - Tracking
  - Teamworks Process Analyst Capability

Enable All  Disable All

Description:

Enables Teamworks features that are deprecated and available for reasons of backward compatibility only.
• Works well for filtering UIs for specific Personas.

• Just filters, doesn't expand what you can do.

• You pick what to exclude, not include.

• PS, called “Activities” in code.

http://dev.eclipse.org/viewcvs/index.cgi/platform-ui-home/activities/
Your own Presentation
But also support Standard Eclipse SDK for Java devs
Lesson Learned 2: Use Standard Extensions
Lesson Learned 3: Expose only what you want, where you want it.
Plug-in Proliferation
package com.mmoeller.blog.client;

import com.google.gwt.core.clientEntryPoint;

/**
 * Entry point classes define <code>onModuleLoad()</code>.
 *<p>
 * public class Blog implements EntryPoint {
 */

/**
 * This is the entry point method.
 */

public void onModuleLoad()

}
package com.mmoeller.blog.client;

import com.google.gwt.core.clientEntryPoint;

/**
 * Entry point classes define <code>onModuleLoad()</code>.
 */

public class Blog implements EntryPoint {

    /**
     * This is the entry point method.
     */
    public void onModuleLoad() {

    }

}
Custom Presentation Layer

Filters UI
Views through extra extensions

Eclipse UI

Platform

JDT ★★★

3rd Party X

3rd Party Y

Your Product A

Your Product B
Example: Toolbar
More or less randomly placed action-groups.

User defined list of perspectives, growing.

“File” based global actions

App Defined list of Perspectives

Global & Navigation actions + Search
Extension point filters
- Makes exposed
- Gives additional data
Type, Images, advanced view implementation
Lesson Learned 4: Build your own custom Presentation
The two parts of Presentation
**PresentationFactory**
controls the graphics around the views, building the Tool and Status bars

**Application**
More content related, but controls the base layout, default menus and similar.
Presentation Classes

WorkbenchPresentationFactory
- createStandaloneViewPresentation
- createView
- createEditor
- createCoolbarManager
- createStatusLineManager

ICoolbarManager
- addItem
  ...

IStatusLineManager
- addItem
  ...

StackPresentation
- createControl
- selectPart
package teamworks.ae.product;

import org.eclipse.jface.action.IContributionManager;

public class TWPresentationFactory extends WorkbenchPresentationFactory_38 {

    public TWPresentationFactory() {
    }

    @Override
    public Control createStatusLineControl(IStatusLineManager statusLine, Composite parent) {
        return ((TWStatusLineManager)statusLine).createControl(parent);
    }

    @Override
    public IStatusLineManager createStatusLineManager() {
        return new TWStatusLineManager();
    }

    public StackPresentation createStandaloneViewPresentation(Composite parent, IStackPresentationSite site, boolean showTitle) {
        return new TWStandalonePartPresentation(parent, site);
    }

    @Override
    public StackPresentation createViewPresentation(Composite parent, IStackPresentationSite site) {
        return new TWViewPartPresentation(parent, site);
    }

    @Override
    public StackPresentation createEditorPresentation(Composite parent, IStackPresentationSite site) {
        return new TWEditorPartPresentation(parent, site);
    }
}
Presentation Classes

WorkbenchPresentationFactory
- createStandaloneViewPresentation
- createView
- createEditor
- createCoolbarManager
- createStatusLineManager

StackPresentation
- createControl
- selectPart

IStatusLineManager
- addItem
- ....

ICoolbarManager
- addItem
- ....
The 3 StackPresentations

Standalone View

Editor

View

View

View
Other Presentation Parts

Toolbar (Coolbar)

StatusBar (StatusBarManager)
Setup:
plugin_customization.ini
org.eclipse.ui/presentationFactoryId=teamworks.ae.product.presFactory
org.eclipse.ui/defaultPerspectiveId=teamworks.perspective.library
Application Classes

IApplication
  start
  stop

WorkbenchAdvisor
  createWorkbenchWindowAdvisor
  ....

WorkbenchWindowAdvisor
  createActionBarAdvisor
  preOpenWindow
  postWindowcreate

ActionBarAdvisor
  makeActions
  fillMenu
  fillCoolbar
ActionBarAdvisor Class
package teamwork.ae.product;

import org.eclipse.jface.action.GroupMarker;

/**
 * An action bar advisor is responsible for creating, adding, and disposing of the
 * actions added to a workbench window. Each window will be populated with
 * new actions.
 */

public class ApplicationActionBarAdvisor extends ActionBarAdvisor {

    // Actions - important to allocate these only in makeActions, and then use them
    // in the fill methods. This ensures that the actions aren't recreated
    // when fillActionBars is called with FILL_PROXY.
    private IWorkbenchAction exitAction;
    private IWorkbenchAction aboutAction;
    private IWorkbenchAction newWindowAction;
    private IWorkbenchAction openPerspective;
    private IWorkbenchAction newAction;
    private IWorkbenchAction saveAction;
    private IWorkbenchAction importAction;
    private IWorkbenchAction openPreferencesAction;
    private IWorkbenchAction helpContentsAction;
    private IWorkbenchAction helpSearchAction;
    private IWorkbenchAction dynamicHelpAction;
    private IWorkbenchAction showView;

    public ApplicationActionBarAdvisor(IActionBarConfigurer configurator) {
        super(configurator);
    }

    protected void makeActions(final IWorkbenchWindow window) {
        // Creates the actions and registers them.
        // Registering is needed to ensure that key bindings work.
    }
}
ActionBarAdvisor
Class
Why worry about the End User UI?

What did we do to solve the problem?

How to create alternate presentations

1. Presentation plugins & APIs
2. Optimize it to work with standard Eclipse plugins

~ 5 Minutes
Understand the difference between Application and Presentation...
Perspective class is hard to skin.
and hard to replace
Learn Reflection!

Nothing is really private.
Learn Reflection!
Nothing is really private.
Actions and ToolItem
Create a very custom toolbar is a lot of work
Why worry about the End User UI?

What did we do to solve the problem?

How to create alternate presentations

1. Presentation plugins & APIs
2. Optimize it to work with standard Eclipse plugins

Gotchas. Things to know.
• Doesn’t have to look or feel like Eclipse even if you support JDK

• Presentation is fairly easy

• Own your UI, don’t let plug-ins rule your world
Talk: Deep integration of GWT/Javascript and Eclipse 3.3 using XULRunner and JavaXPCOM

By: Bryan Campbell

When: Wednesday, 10:20 in Room 209/210
Designing Business & Technical UIs

Morten Moeller
Lead Product & UX Designer
Lombardi
mmoeller@lombardi.com