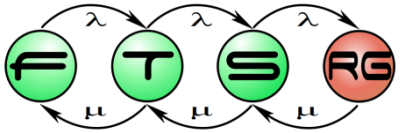
Web Application Development



UI Development Trends

Desktop applications

- Operating system integration
- Rich set of available widgets
- Performance and security

Web applications

- Web 2.0 standards
- No installation required
- Collaboration and communication

What is RAP?

- Component-oriented development
- Based on Eclipse Workbench (+ SWT, JFace)
- Based on Eclipse plug-ins
  - Dependency management
  - Extension points
  - Lifecycle management
  - Modular UI (similar to Web 2.0 mashups)

The Goals of RAP

- Support Java
  - Using the full Java API (compare with GWT)
  - Based on Java-based widget toolkit
- Uses component-oriented model (OSGi)
  - Modular (Eclipse extension points)
  - Can be used without web/Javascript coding

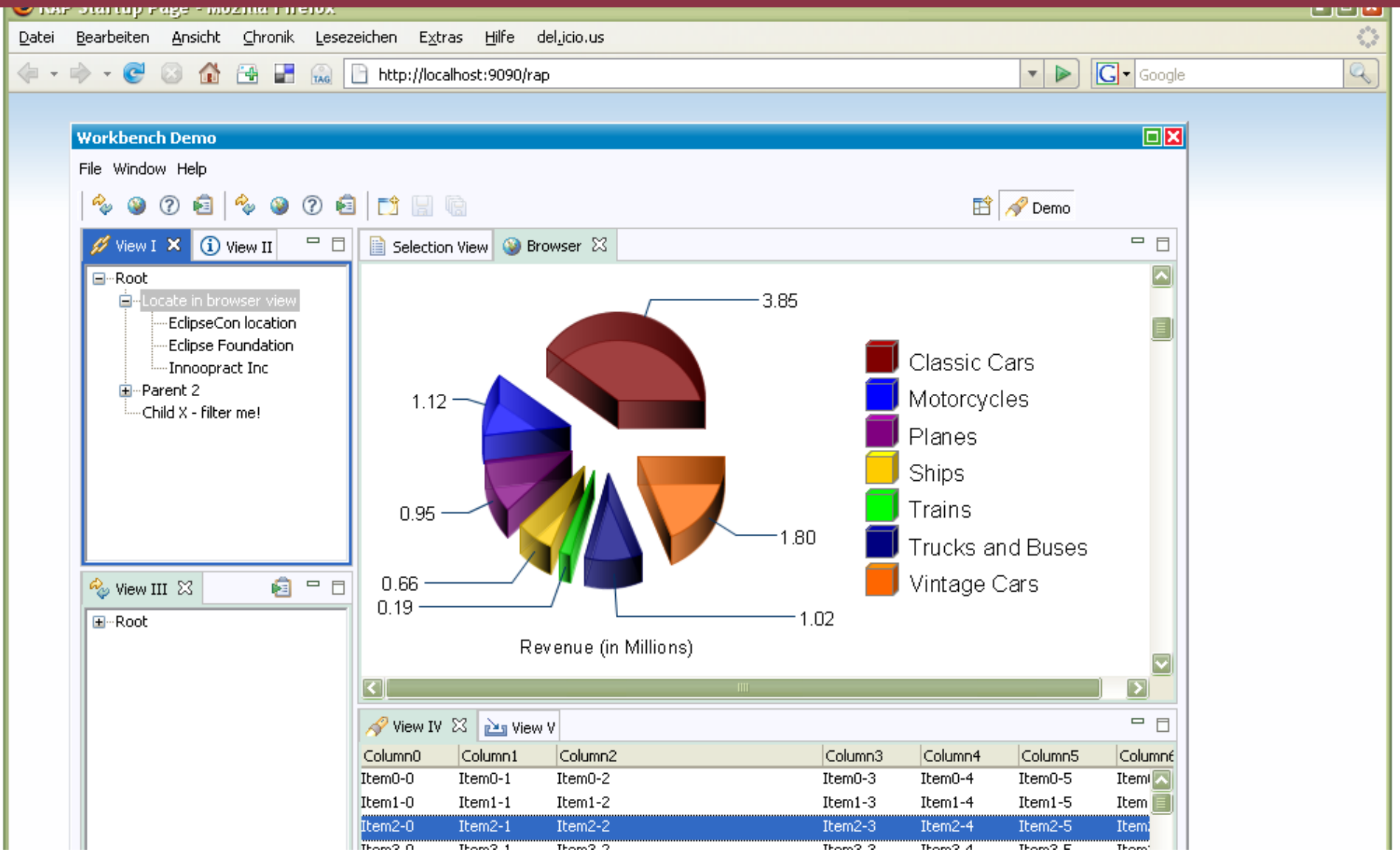
Why RAP?

- Skill preservation
  - Eclipse RCP knowledge reusable
- Single sourcing
  - Common technology for client and web application

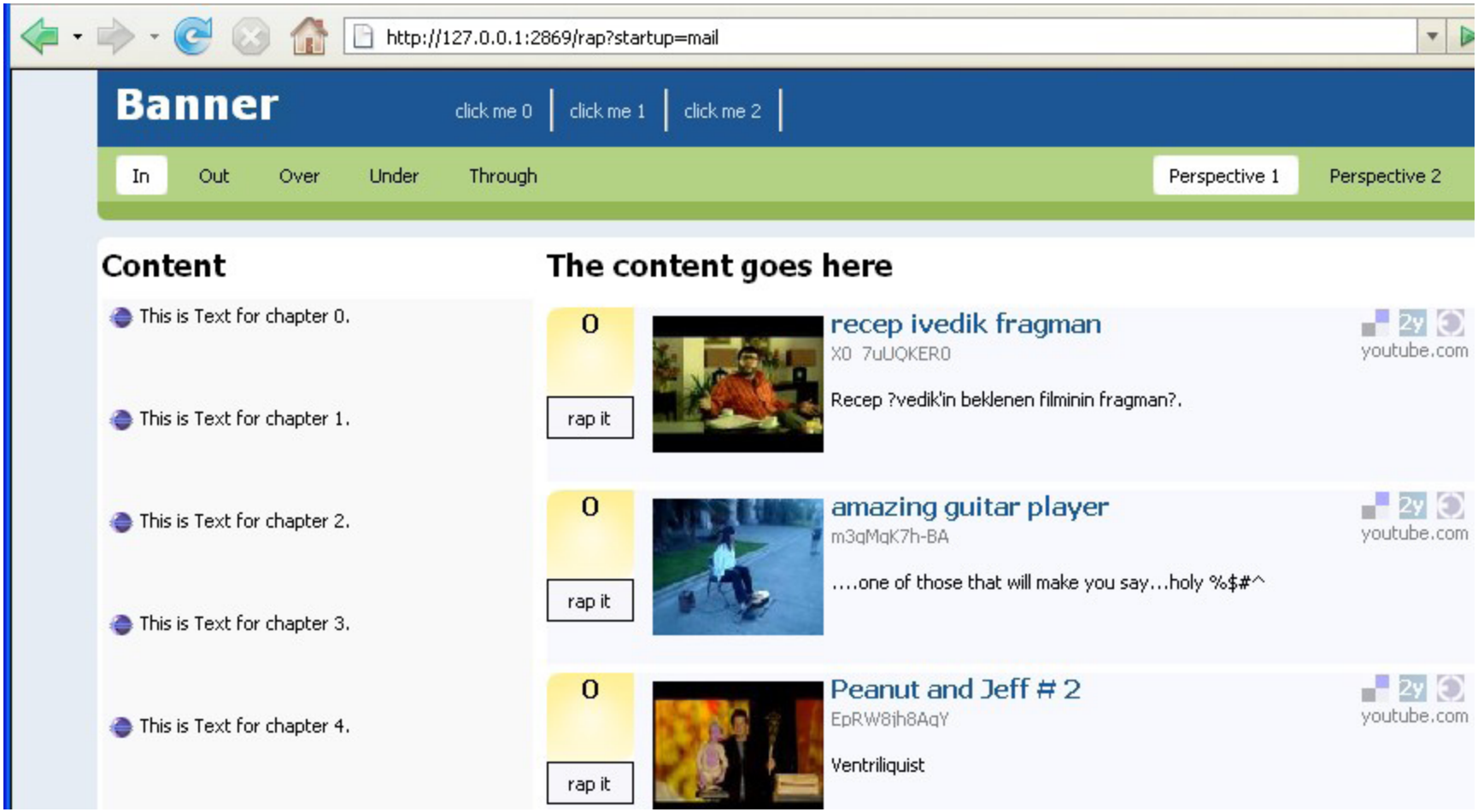
How well does it work?

- Eclipse/RCP code…
  - 70-90% reusable
  - RAP implements a(n increasing) subset of RCP
  - But incompatibilities
  - **Important:** multi-user support

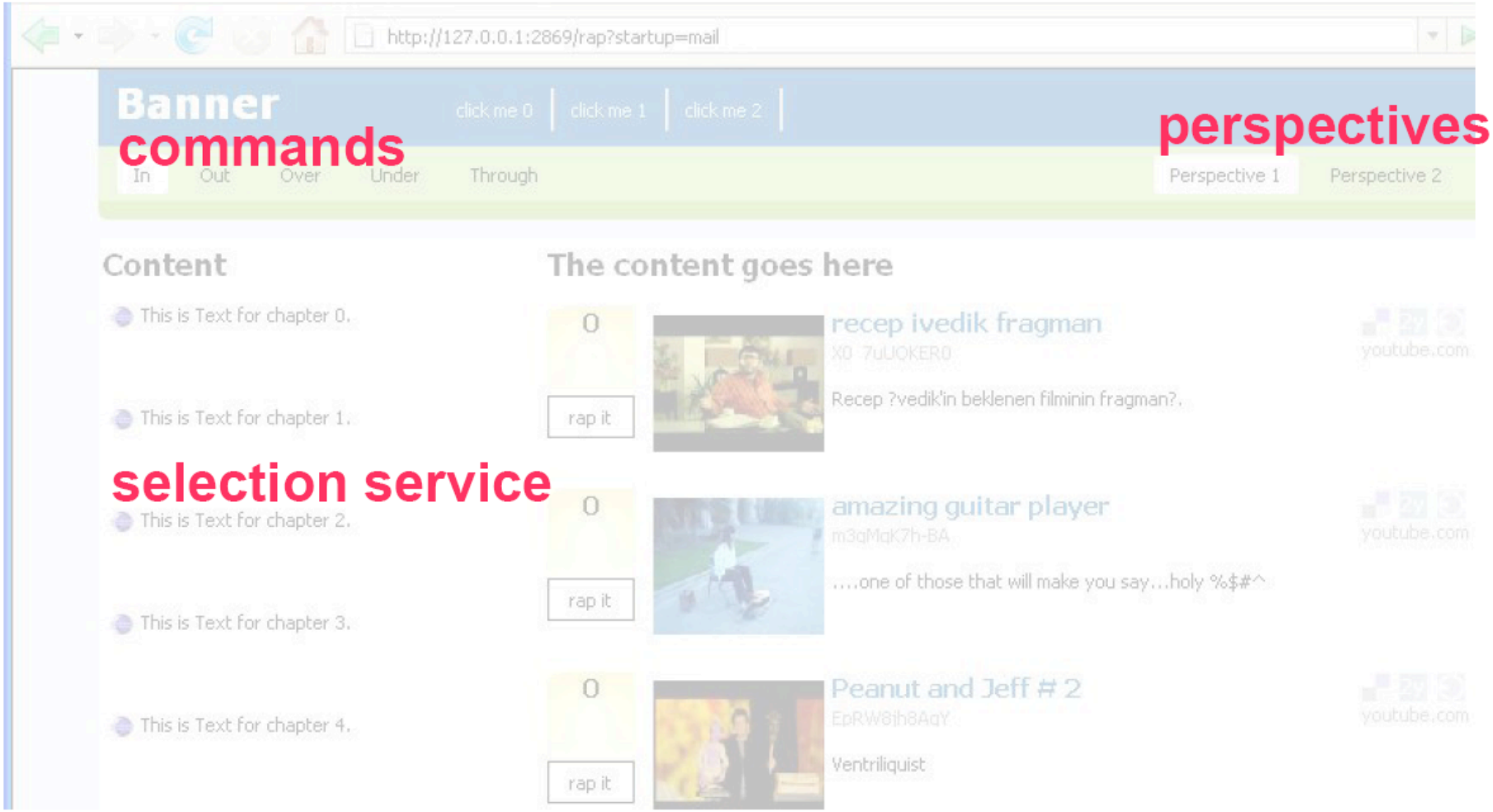
Sample application



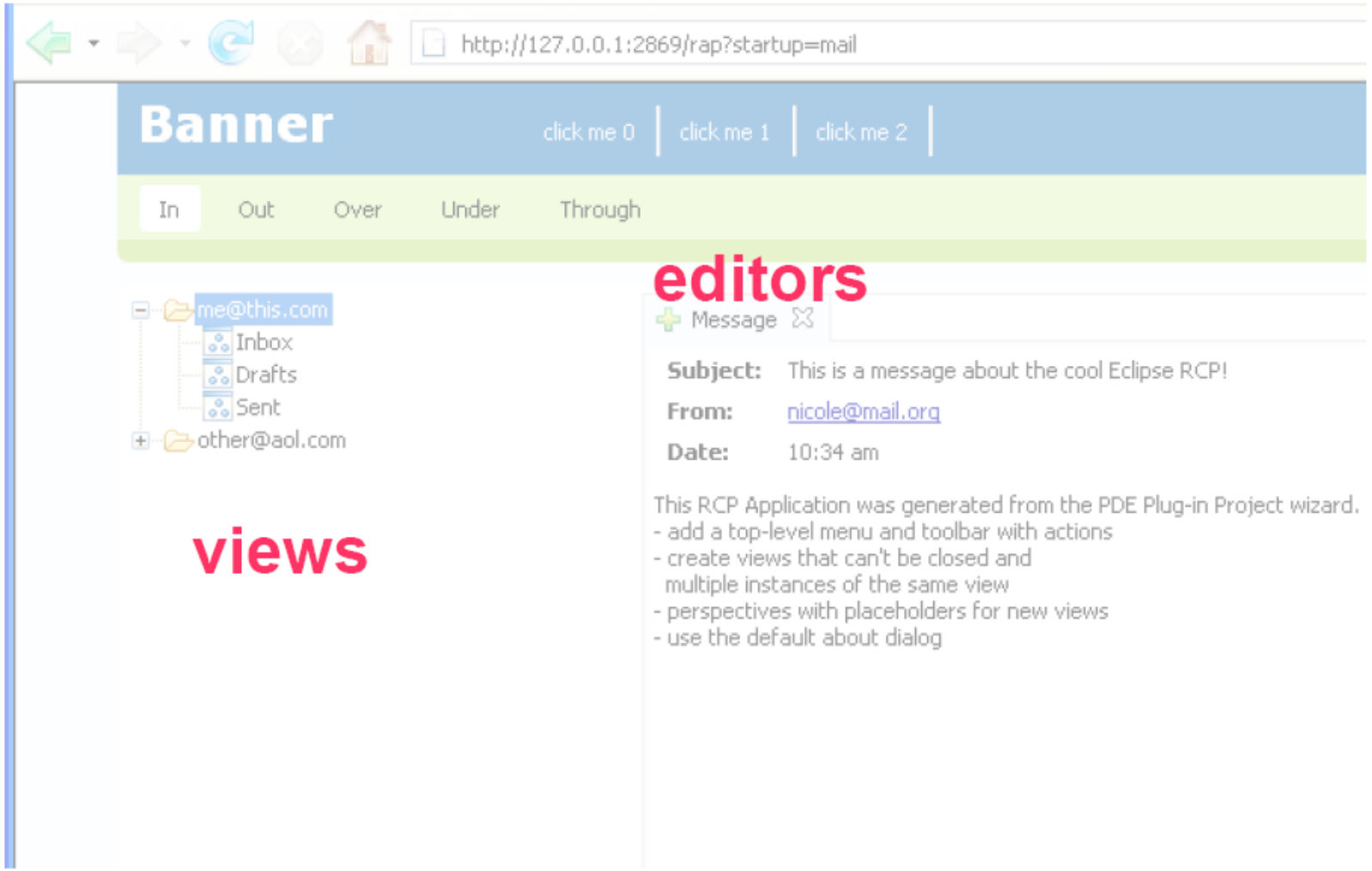
Not only RCP appearance...



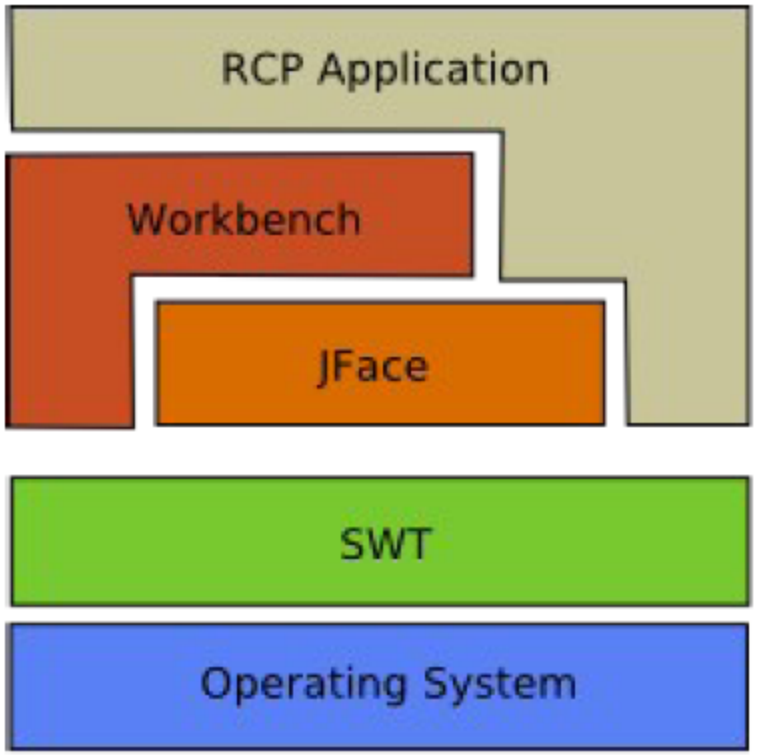
But Still an Eclipse Workbench

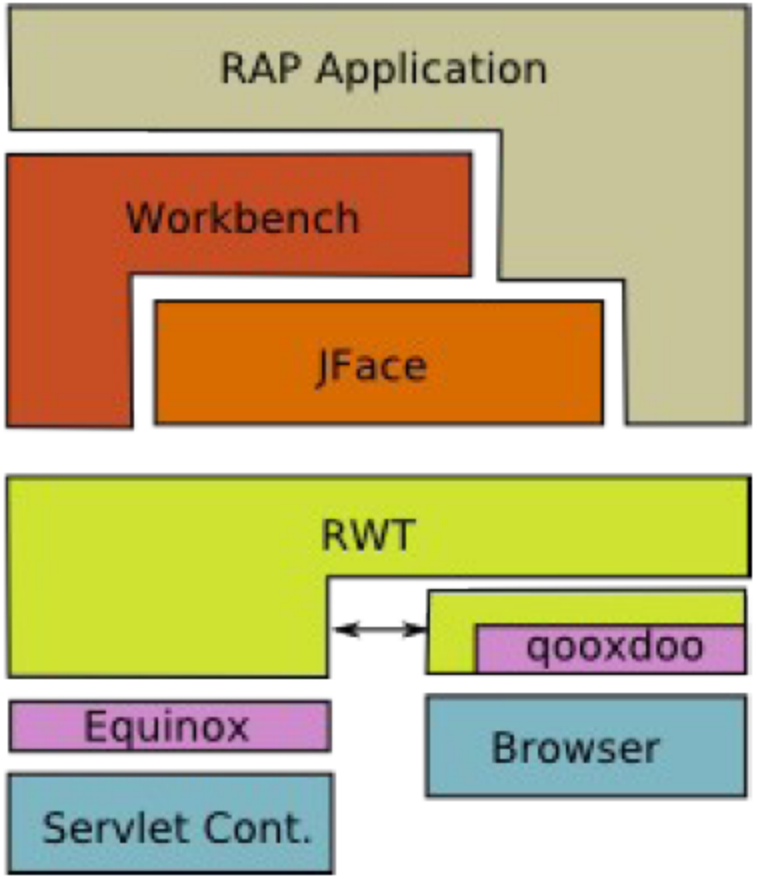


Views, Editors



Architecture





Architecture

- RWT (RAP Widget Toolkit)
  - Web-based implementation of SWT
  - Uses qooxdoo Javascript library
- Most SWT-based library works (mostly)

- Important difference: multiuser behaviour
  - Application bundles are shared between sessions

Internals

- Server centric
  - Entire business logic in server
- Browser
  - Javascript based rendering
  - Since RAP 2.0 uses a JSON for communication
- User event results in state change
  - → evaluated by the server
  - State delta traverses on the network

For the developer…

org.eclipse.rap.demo *DemoTreeViewPart.java

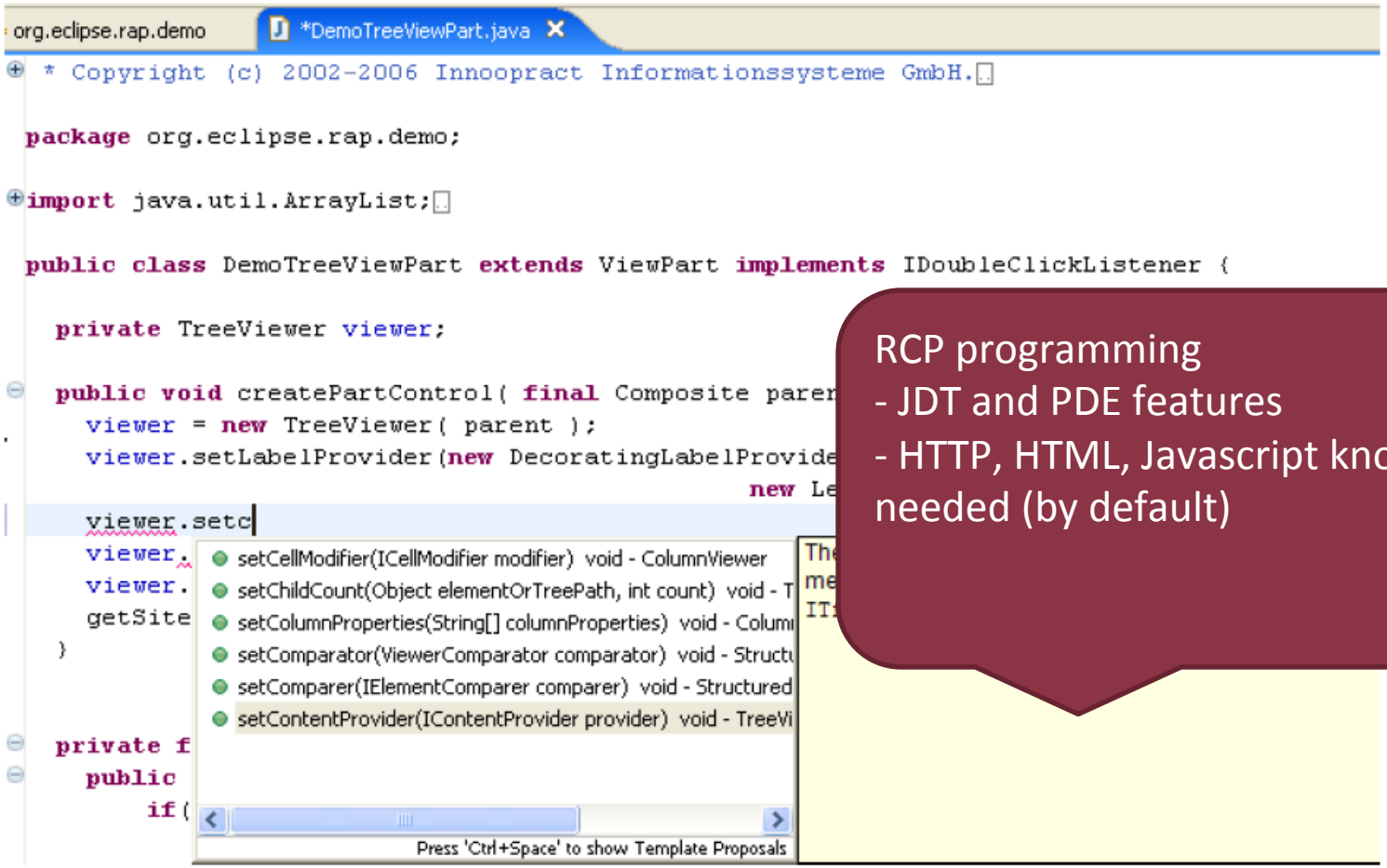
```
 * Copyright (c) 2002-2006 Innoopract Informationssysteme GmbH.

package org.eclipse.rap.demo;

import java.util.ArrayList;

public class DemoTreeViewPart extends ViewPart implements IDoubleClickListener {

  private TreeViewer viewer;

  public void createPartControl( final Composite parent ) {
    viewer = new TreeViewer( parent );
    viewer.setLabelProvider(new DecoratingLabelProvider(new LabelProvider(),
                                                        new LeafStarLabelDecorator()));
    viewer.setc
    viewer.
    viewer.
    getSite
  }

  private f
    public
      if(
```

setCellModifier(ICellModifier modifier) void - ColumnViewer
setChildCount(Object elementOrTreePath, int count) void - T
setColumnProperties(String[] columnProperties) void - Colum
setComparator(ViewerComparator comparator) void - Structu
setComparer(IElementComparer comparer) void - Structured
setContentProvider(IContentProvider provider) void - TreeVi

Press 'Ctrl+Space' to show Template Proposals

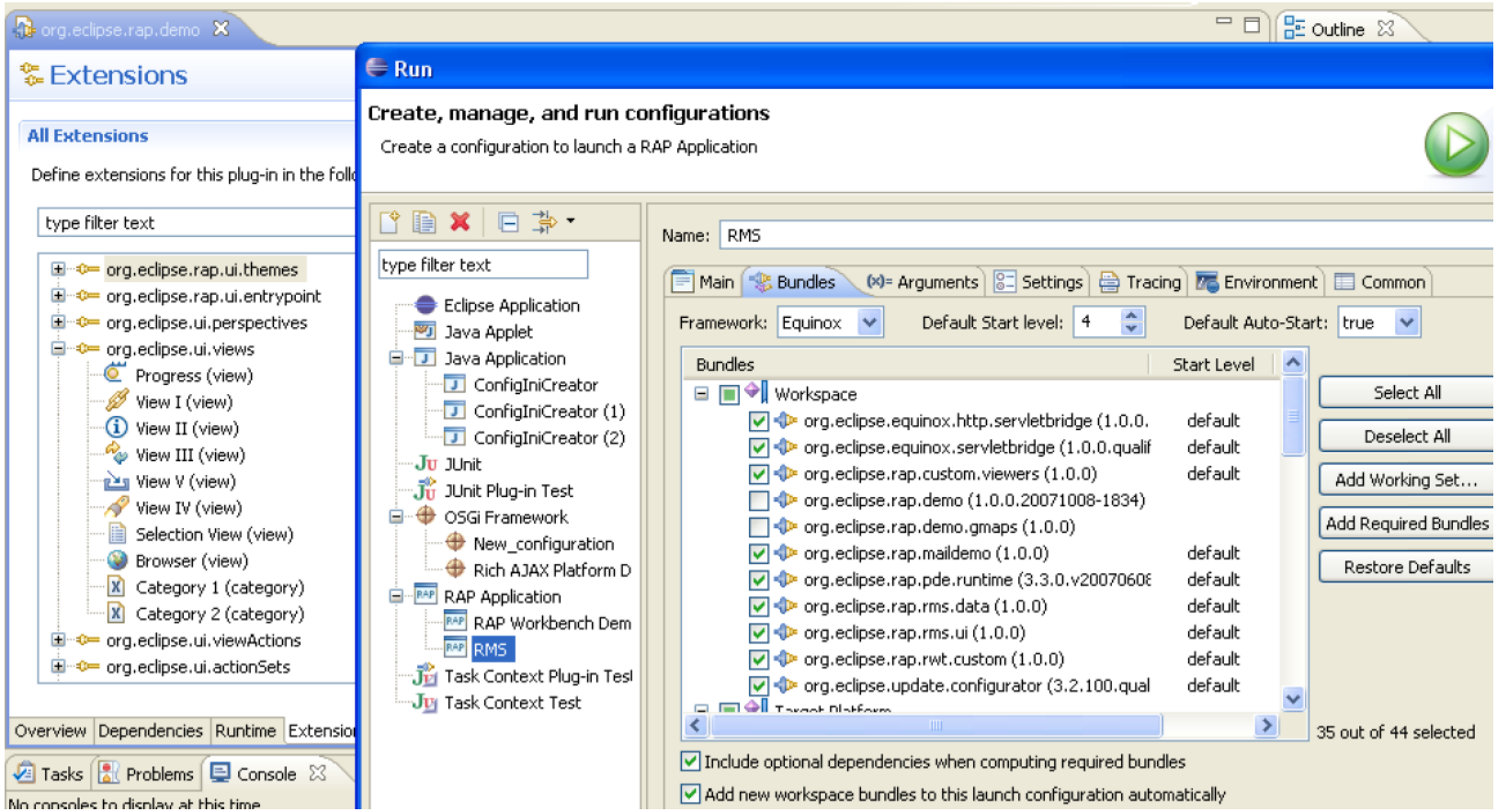
The `AbstractTreeViewer` implementation of this method checks to ensure that the content provider is an `ITreeContentProvider`.

For the developer…

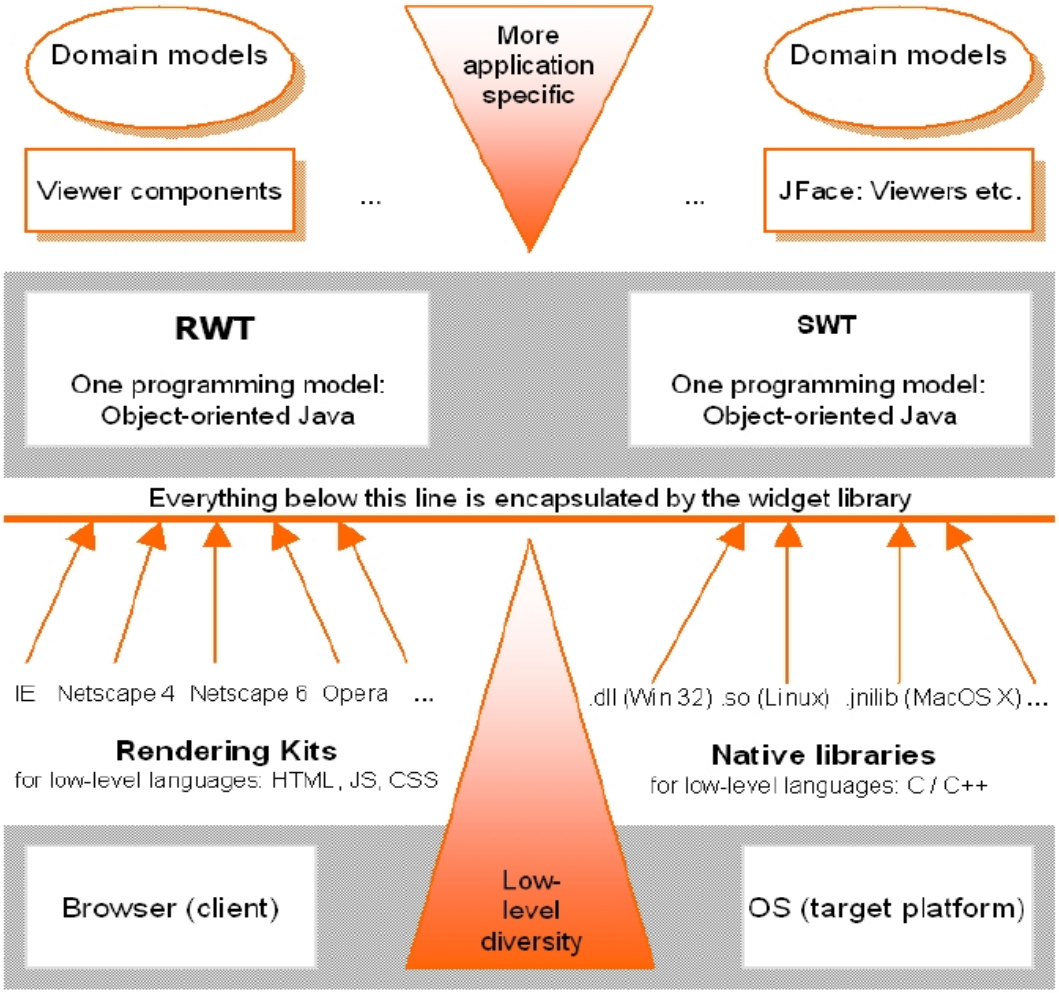


RCP programming
- JDT and PDE features
- HTTP, HTML, Javascript knowledge not needed (by default)

RAP Application Execution / Debug

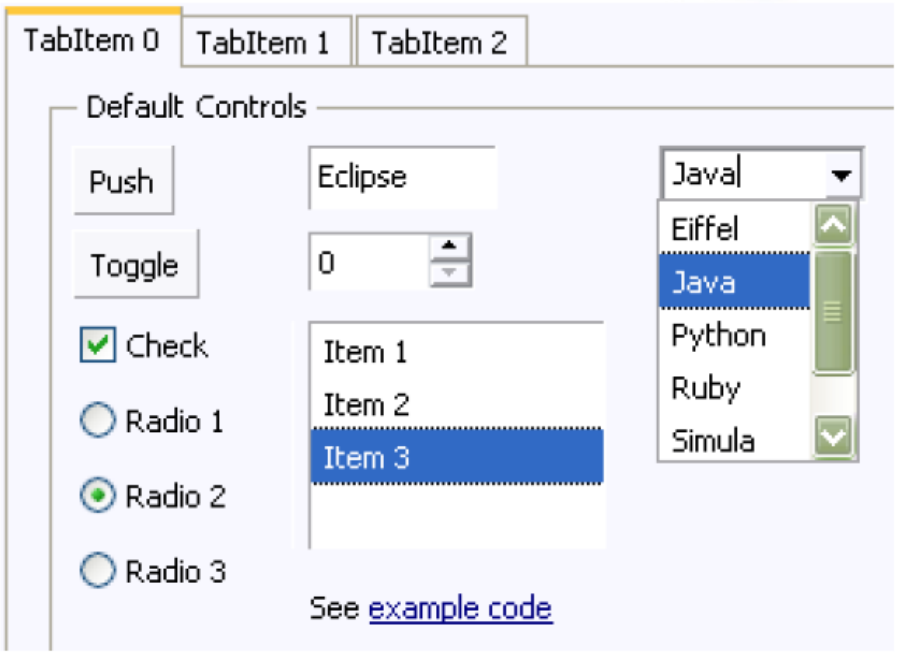


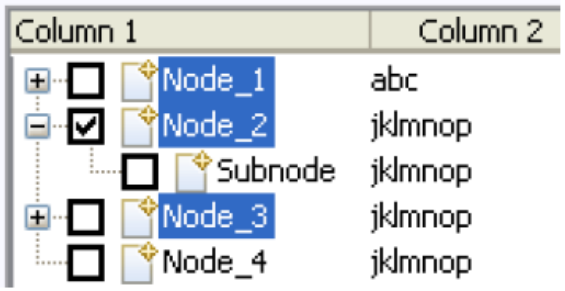
RWT Architecture

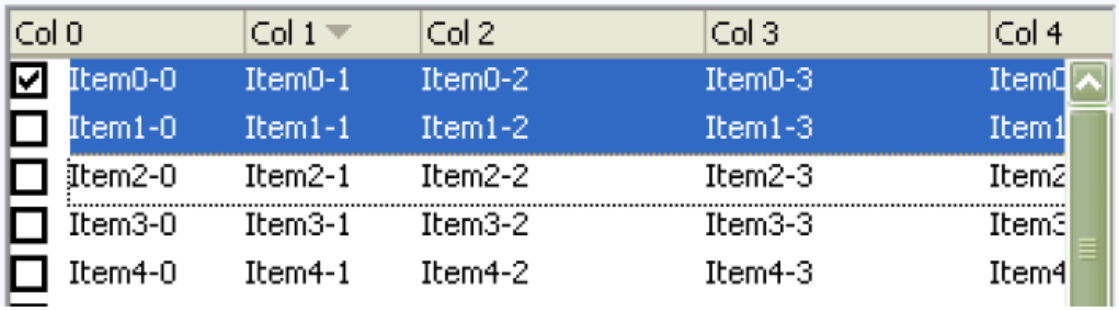


RWT

- Most basic widgets available
- Workbench widgets (CTabFolder, CBanner)
- Browser widget ☺
- Tree, Table

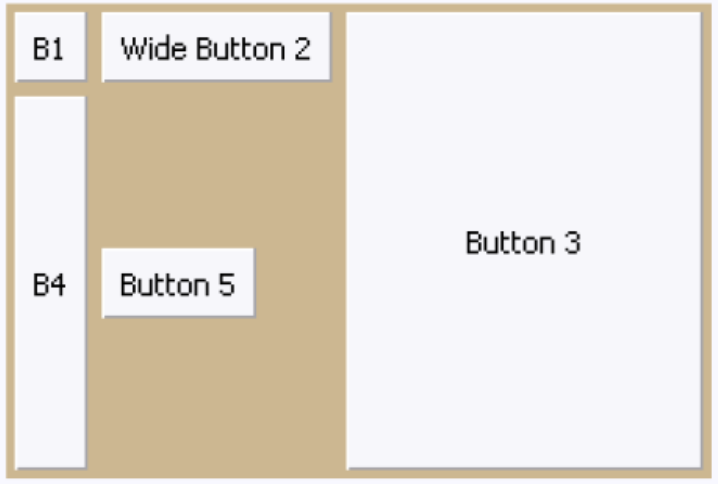






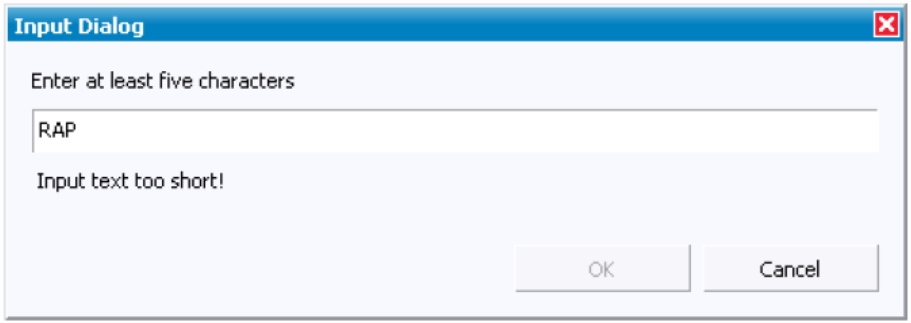
RWT Layouts

- All layouts available:
  - Grid, Row, Fill, Form, Stack
  - Basically the same as SWT Layouts
- Behaviour
  - Server calculates layout with pixel precision



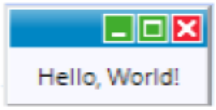
Event Handling

- All SWT Listeners supported
  - SelectionListener
  - ControlListener
  - ShellListener
  - MenuListener
  - …



- JFace data binding, validation also works

RAP Hello World

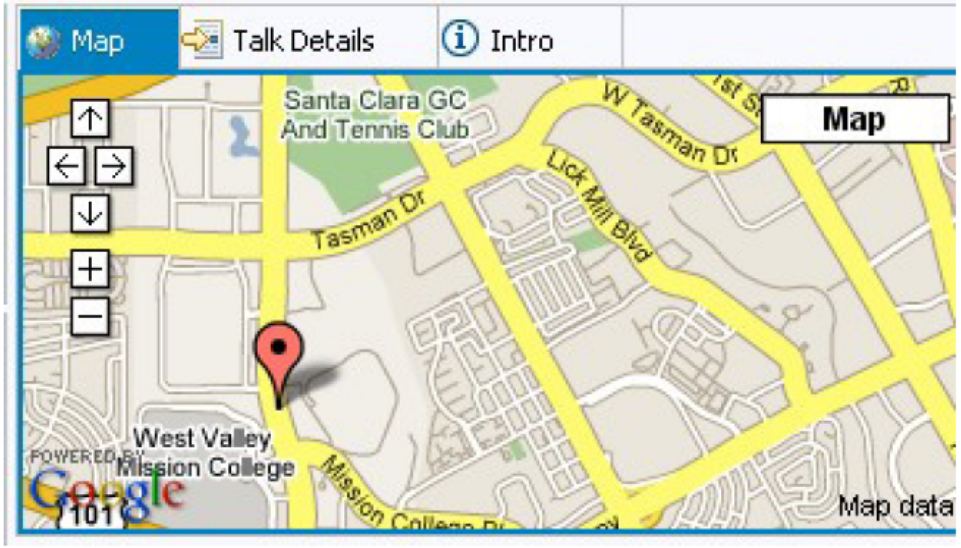



```
public class RWTHello implements IEntryPoint {

  public Display createUI() {
    Display result = new Display();
    final Shell shell = new Shell( result );
    RowLayout layout = new RowLayout();
    layout.justify = true;
    layout.pack = true;
    shell.setLayout( layout );
    Label label = new Label( shell, SWT.CENTER );
    label.setText( "Hello, World!" );
    shell.pack();
    shell.open();
    return result;
  }
}
```

```
<extension
    id="org.eclipse.rap.demo.demoentrypoint"
    point="org.eclipse.rap.ui.workbench.entrypoint">
  <entrypoint
      class="org.eclipse.rap.tutorial.HelloWorld"
      parameter="default"/>
</extension>
```

Extensibility (Custom Widgets)

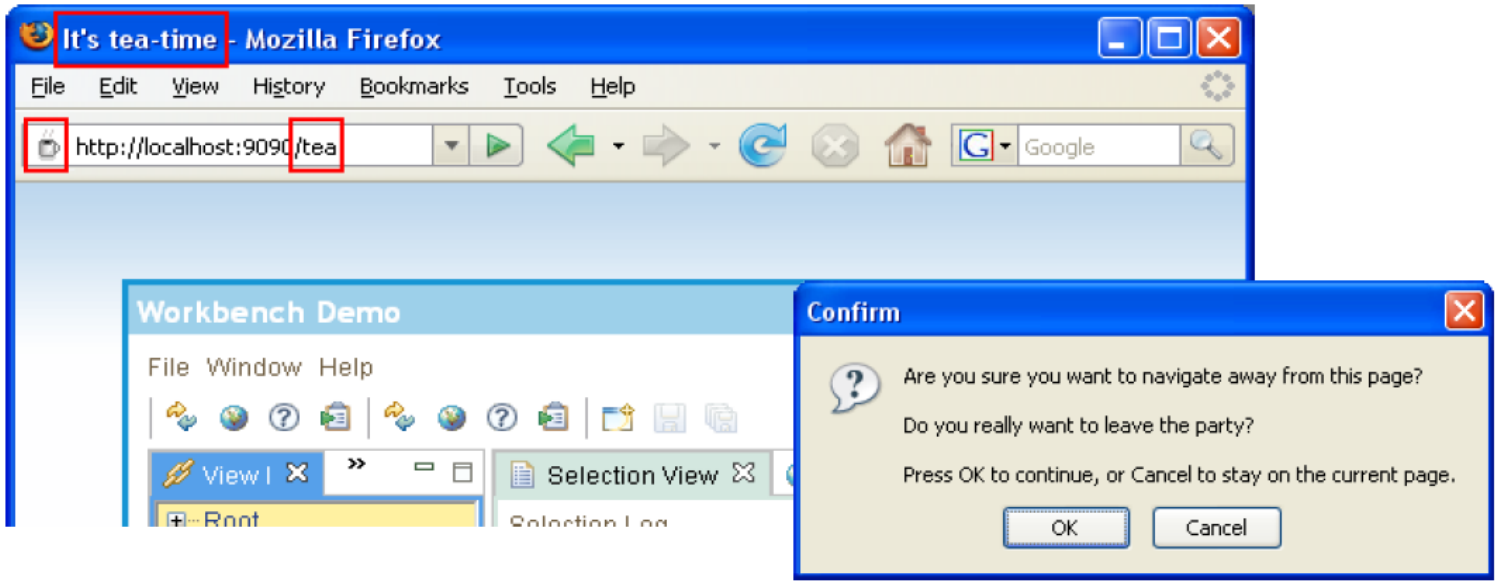
- Possible, but requires deep understanding
  - Component developer: Javascript, qooxdoo, RAP
  - Application developer: only Java API
- Example
  - Google Maps integration: RAP Help



RWT Customization (Themes)

- Unique description
- CSS-based theming
  - More powerful than Eclipse 4 CSS
  - Widget settings
    - Size
    - Colors
    - Fonts
    - …
  - Works for custom widgets (if implemented)

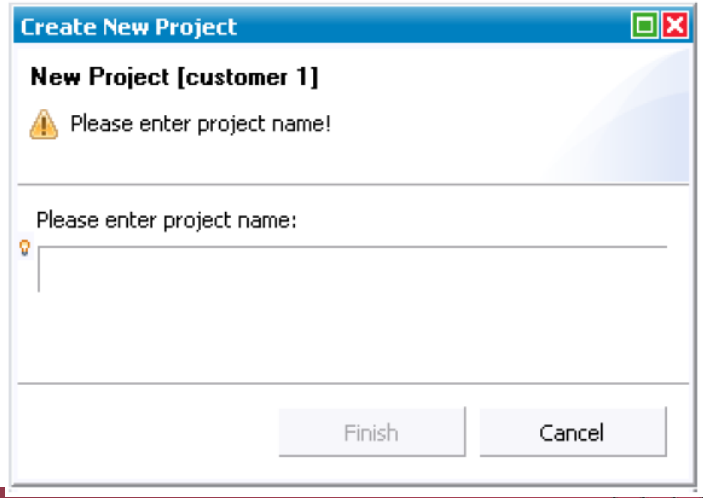
Branding



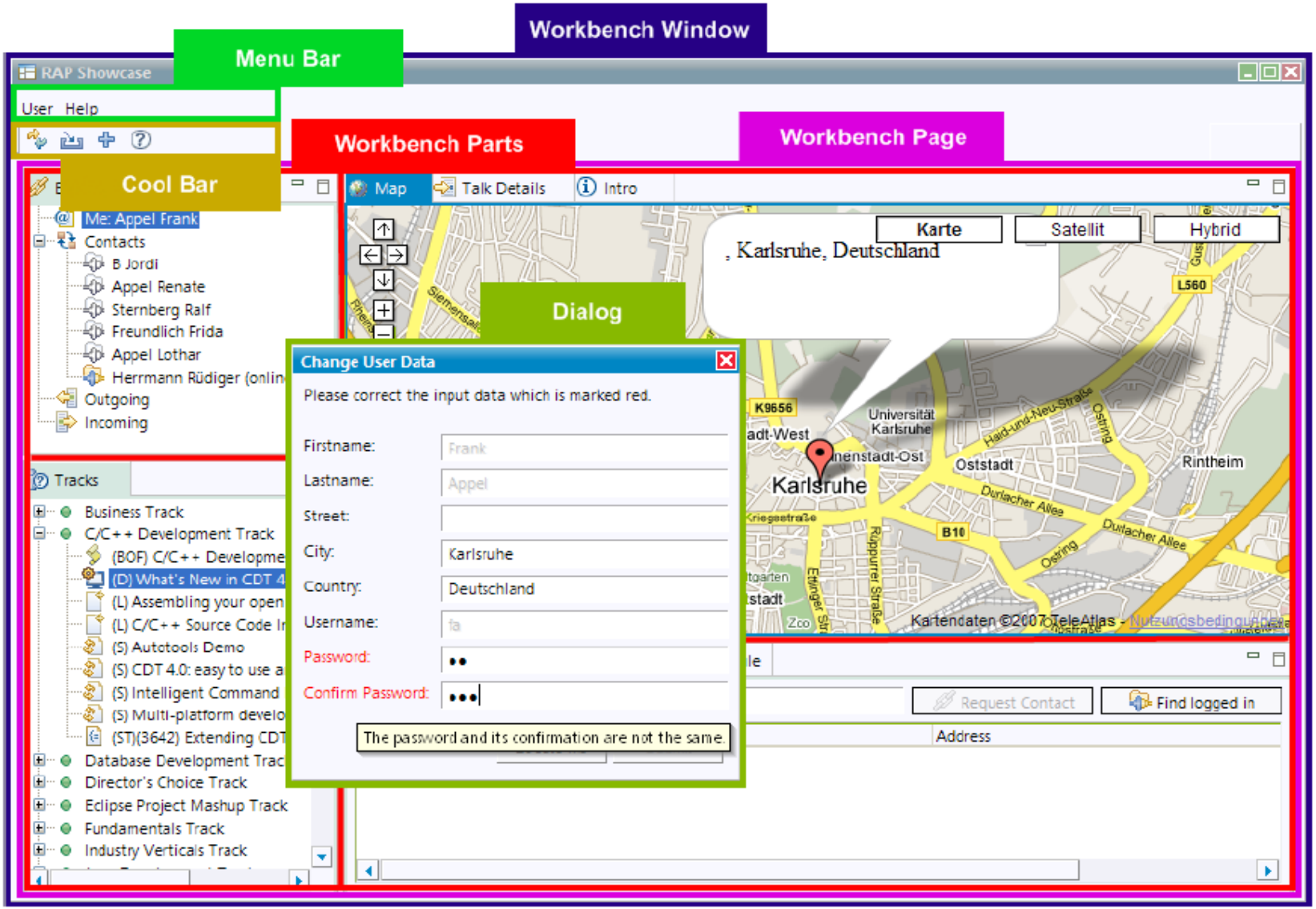

```
<extension
      point="org.eclipse.rap.ui.branding">
   <branding
         id="org.eclipse.rap.demo.branding1"
         servletName="tea"
         defaultEntrypointId="org.eclipse.rap.demo.entrypoint1"
         themeId="org.eclipse.rap.demo.alttheme"
         title="It&apos;s tea-time"
         favicon="icons/favicon2.ico"
         body="body.html"
         exitConfirmation="Do you really want to leave the party?">
   </branding>
</extension>
```

RAP and JFace

- JFace viewer framework works as expected
  - TableViewer, TreeViewer
  - Provider, Sorter, Filter, Decorator
  - Cell Editor
- Dialogs
  - Standard dialogs (MessageDialog, ErrorDialog)
  - Modal dialogs
  - Custom dialogs
- Wizards

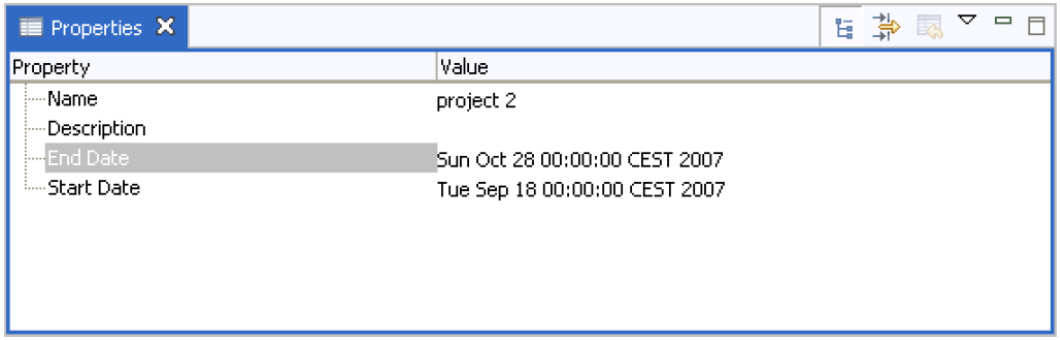


RAP and Workbench



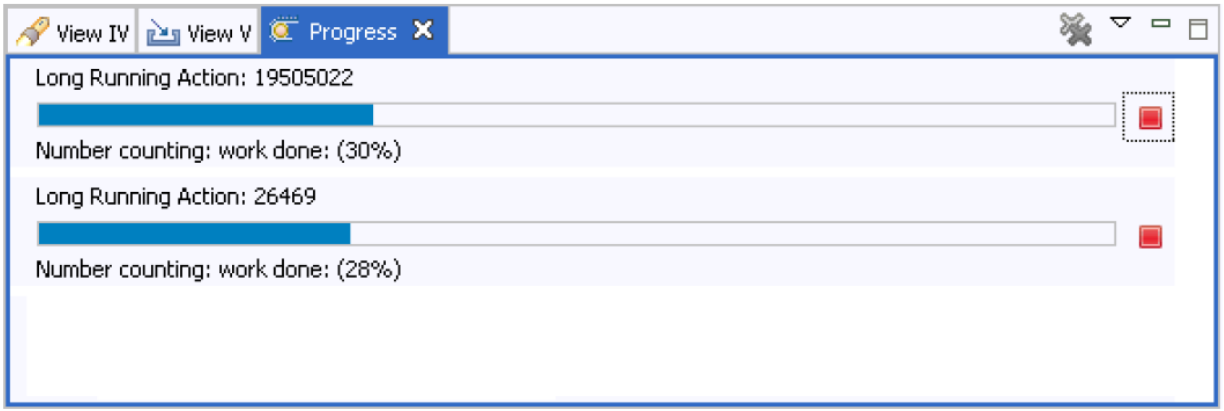
RAP and Workbench

- View
  - Full Support
  - Outline, Properties
- Editor
  - Multi-page editors as well
- Perspective
  - Built-in perspective switcher
- Actions, Commands



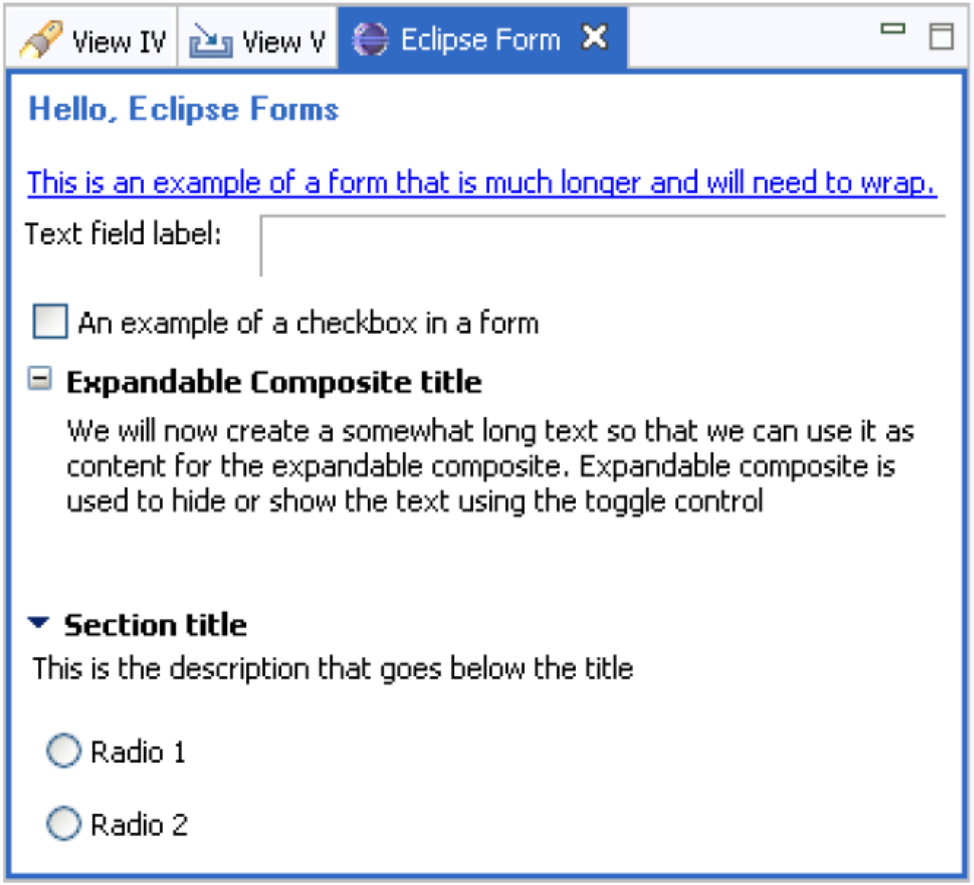
Workbench jobs, tasks

- Support for Job API
  - Run in background
  - Progress Monitors



Forms, Data binding

- Data binding works
- Eclipse UI Forms works



RAP Deployment

- An RAP application can be deployed to
  - JEE Server konténerbe (WAR)
    - Uses OSGi Servlet Bridge
  - OSGi standalone (default)
    - Uses Jetty as web server

RAP – More information

- http://www.eclipse.org/rap/
- http://rapblog.innoopract.com/
- http://www.ibm.com/developerworks/library/os-eclipse-richajax1/
- http://www.infoq.com/news/2008/06/eclipse-ganymede-rap;jsessionid=A0CC55BBF7B9C7CB44D51CD26D6FB6EE

RAP and RCP Together: Single Sourcing in Practice

Main Differences

RCP	RAP
File System	File Upload
Graphical Canvas	Limited Canvas
One User	Multiple Users

Session Management

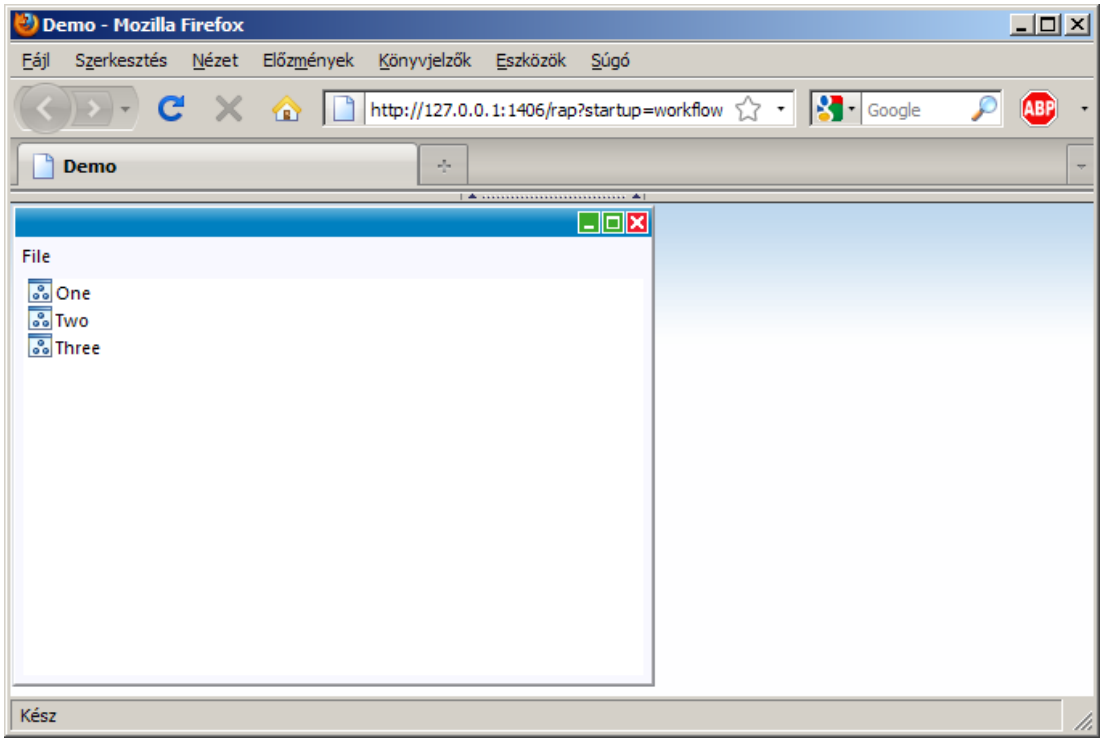
- Multiple users => session management
  - No static/singleton states!
  - RWT session store
    - RWT.getSessionStore()

RWT Session Management

- Querying Session id:
  - `RWT.getSessionStore().getId();`
- User-specific session info
  - `RWT.getSessionStore().getHttpSession().put(key, value);`
- Session listener
  - `RWT.getSessionStore().addSessionStoreListener( myListener);`
- HttpServletRequest object also available
  - `RWT.getRequest();`

Appearance

- Different UI paradigm
- Window in window
  - Problematic
  - Closable



Appearance

- Solution - advisor:

```
@Override
public void preWindowOpen() {
    IWorkbenchWindowConfigurer configurer =
        getWindowConfigurer();
    configurer.setShellStyle(SWT.NO_TRIM);
}

@Override
public void postWindowOpen() {
    final IWorkbenchWindow window =
        getWindowConfigurer().getWindow();
    Shell shell = window.getShell();
    shell.setMaximized(true);
}
```

No border

Maximalized windows

Single Sourcing

- Common functionality between RAP and RCP
  - Minimal costs
  - Common codebase
    - Few code duplication
- Identify UI-specific code
  - Use target platforms
  - Use plug-in fragments

Plug-in fragments

- Not a standalone plug-in
  - Defines a host
  - No Activator (no separate lifecycle)
  - No unique classloader for fragments
    - Reuses host classloader
- Tipical uses
  - Multilanguage support
  - Multiple platform support

Single Sourcing with RAP fragment

- Host plugin: RCP
  - Optional dependencies:
    - org.eclipse.ui
    - org.eclipse.rap.ui
- Fragment: RAP
  - Every RAP-specific code goes here
  - Entry point
  - Branding

Hiding API differences

- Use common base classes

```
public abstract class AboutActionHelper {

  private static final AboutActionHelper IMPL;

  static {
    Class clazz = AboutActionHelper.class;
    Object newInstance = ImplementationLoader.newInstance( clazz );
    IMPL = ( AboutActionHelper )newInstance;
  }

  public static IWorkbenchAction create( IWorkbenchWindow window ) {
    return IMPL.createInternal( window );
  }

  protected abstract IWorkbenchAction createInternal( IWorkbenchWindow
    window );

}
```

Hiding API differences

- Load implementation using reflective API

```
public final class ImplementationLoader {

  public static Object newInstance( Class type ) {
    String name = type.getName();
    Object result = null;
    ClassLoader loader = type.getClassLoader();
    try {
      Class clazz = loader.loadClass( name + "Impl" );
      result = clazz.newInstance();
    } catch( Throwable t ) {
       // ...
    }
    return result;
  }

  private ImplementationLoader() {}

}
```

Hiding API differences

- RAP-specific implementation

```
public class AboutActionHelperImpl extends AboutActionHelper {
  private static final class AboutAction extends Action
    implements IWorkbenchAction  {
      private AboutAction() {
        setText( "About" );
        setId( "aboutAction" );
      }
      public void run() {
        MessageDialog.openInformation(null, "About", "About");
      }
      public void dispose() {}
  }
  protected IWorkbenchAction createInternal( IWorkbenchWindow
    win ) {
    return new AboutAction();
  }
}
```

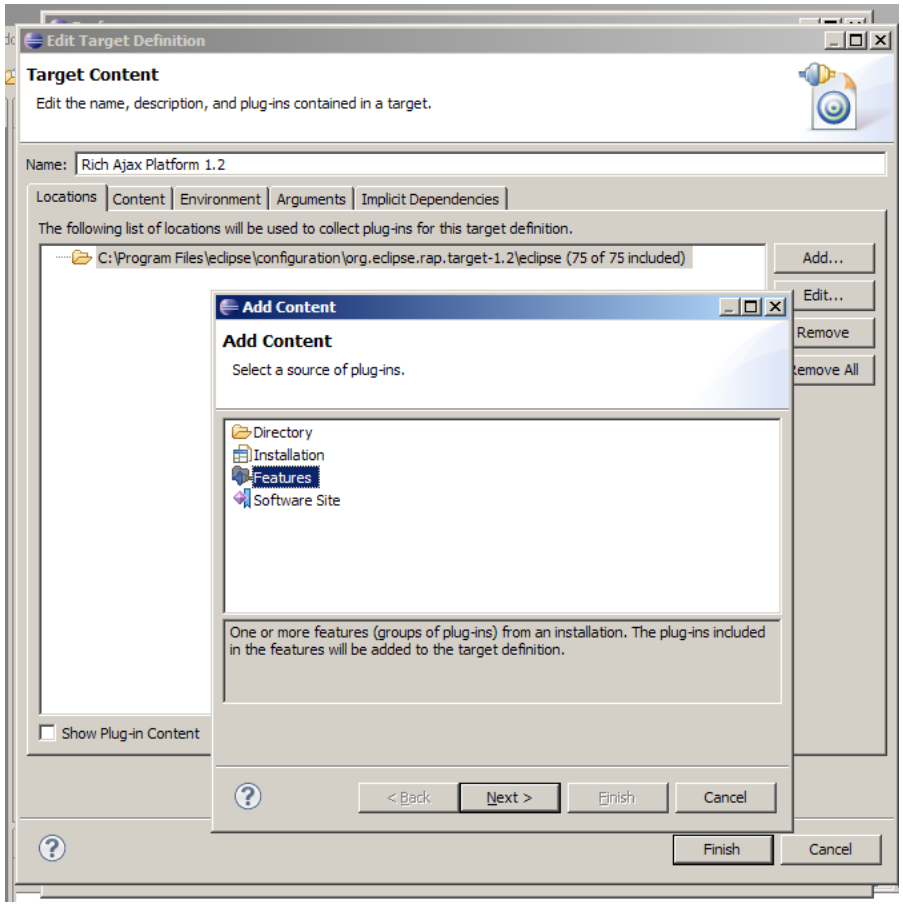
Hiding API differences

- RCP-specific implementation

```
public class AboutActionHelperImpl extends AboutActionHelper {
    protected IWorkbenchAction createInternal( IWorkbenchWindow
     win ) {
     return ActionFactory.ABOUT.create(win);
   }
}
```

Target Platform Management

- No custom workspace required
- Add required plug-ins to the target platform



RAP Branding

- Different options than in RCP (extension point)
  - servletName
    - For the URL (default: rap)
  - defaultEntrypointId
    - Application entry point
  - themeId
    - (CSS) Theme identifier
  - title
    - For the HTML title
  - favicon
  - body
    - Additional HTML code for the page body
  - additionalHeaders gyermekelemek
    - Additional HTML code for page header

RWT Theming

- Widgetek appearance
- CSS-like definition

```
* {
  font: 12px Tahoma;
}
```

Changes default font for all elements

```
TableItem:even {
  background-color: #eef7ff;
}
```

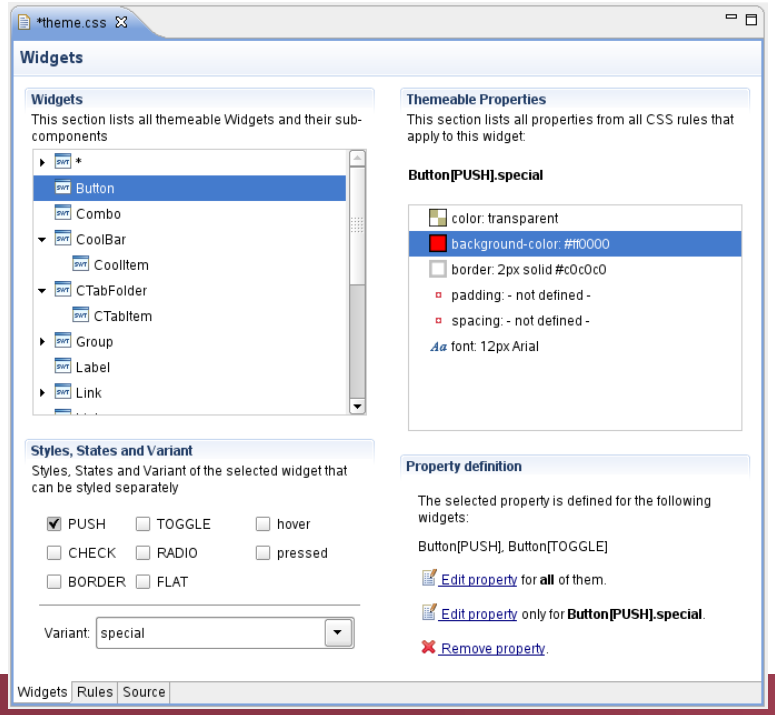
Background for table even rows

RWT Theming

- Selectors:
  - Widget (Button, List…)
  - Style (PUSH, TOGGLE, CHECK, RADIO, BORDER, FLAT)
  - State (hover, pressed, disabled, selected)
- Properties:
  - color
  - padding
  - border
  - font
- Reference:
  - http://eclipse.org/rap/developers-guide/devguide.php?topic=theming.html&version=2.2

RWT Theme Editor

- RAP Theme Editor
  - Form-based editor
  - Preview functionality
  - http://wiki.eclipse.org/RAP_Theme_Editor



Custom RWT Widget

- Required steps
  - Java: Widget-interface creation
  - JavaScript: Widget creation with qooxdoo framework
  - Combining client and server-side
  - Javascript registration in plugin.xml

- More details: http://eclipse.org/rap/developers-guide/devguide.php?topic=custom-widget.html&version=2.2

Theming Custom Widgets

- org.eclipse.rap.ui.themeableWidgets extension point
- Required:
  - Theme adapter: org.eclipse.rwt.theme.IControlThemeAdapter
  - Theme definition: XML fájl
  - Appearance fragment: CSS template

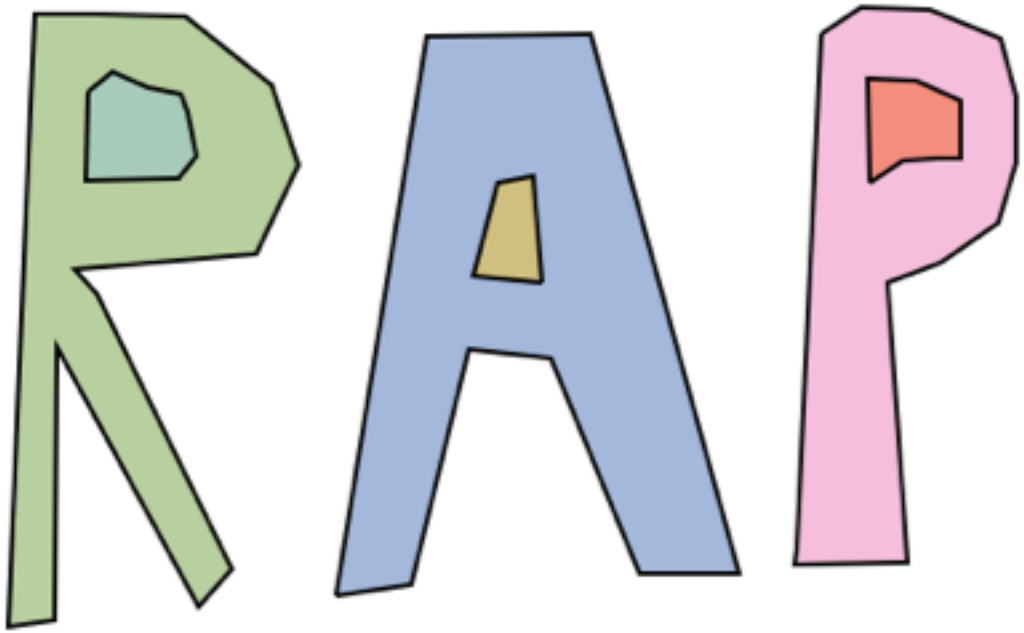
- In more details: http://dev.eclipse.org/viewcvs/index.cgi/org.eclipse.rap/org.eclipse.rap.help/help/html/advanced/theming-custom.html?root=Technology_Project&view=co

Additional features – 1.

- Theme contribution:
  - Extending existing themes
  - Uses org.eclipse.rap.ui.themes extension point

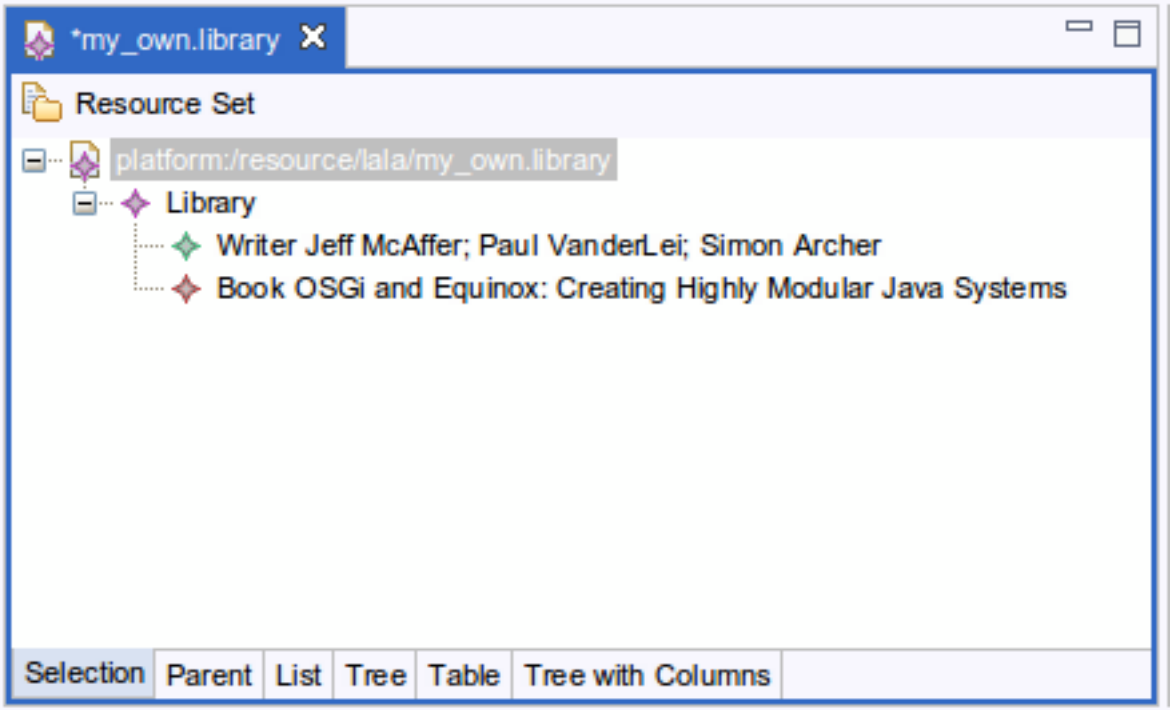
Additional Features – 2.

- Graphics Context:
  - SWT Canvas
  - Subset of SWT options



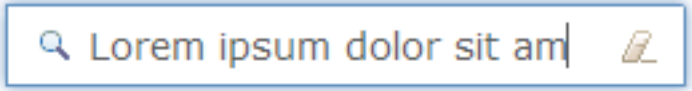
Additional Features – 3.

- EMF on RAP
  - EMF tree editor in RAP
  - Can be generated from EMF genmodel

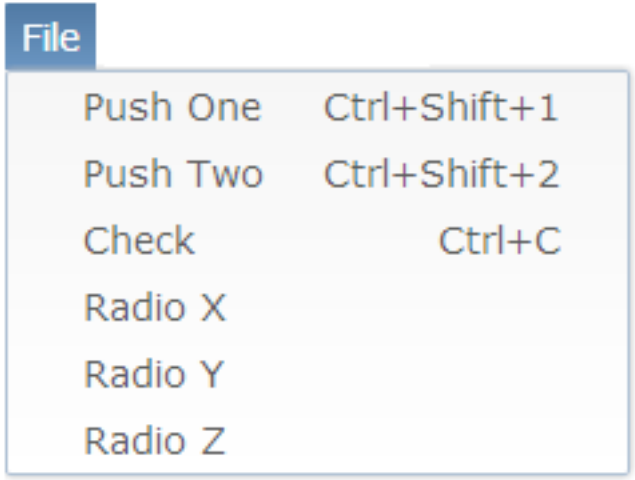


Additional Features – 4.

- SWT.ICON_SEARCH and SWT.ICON_CANCEL support for Text widget

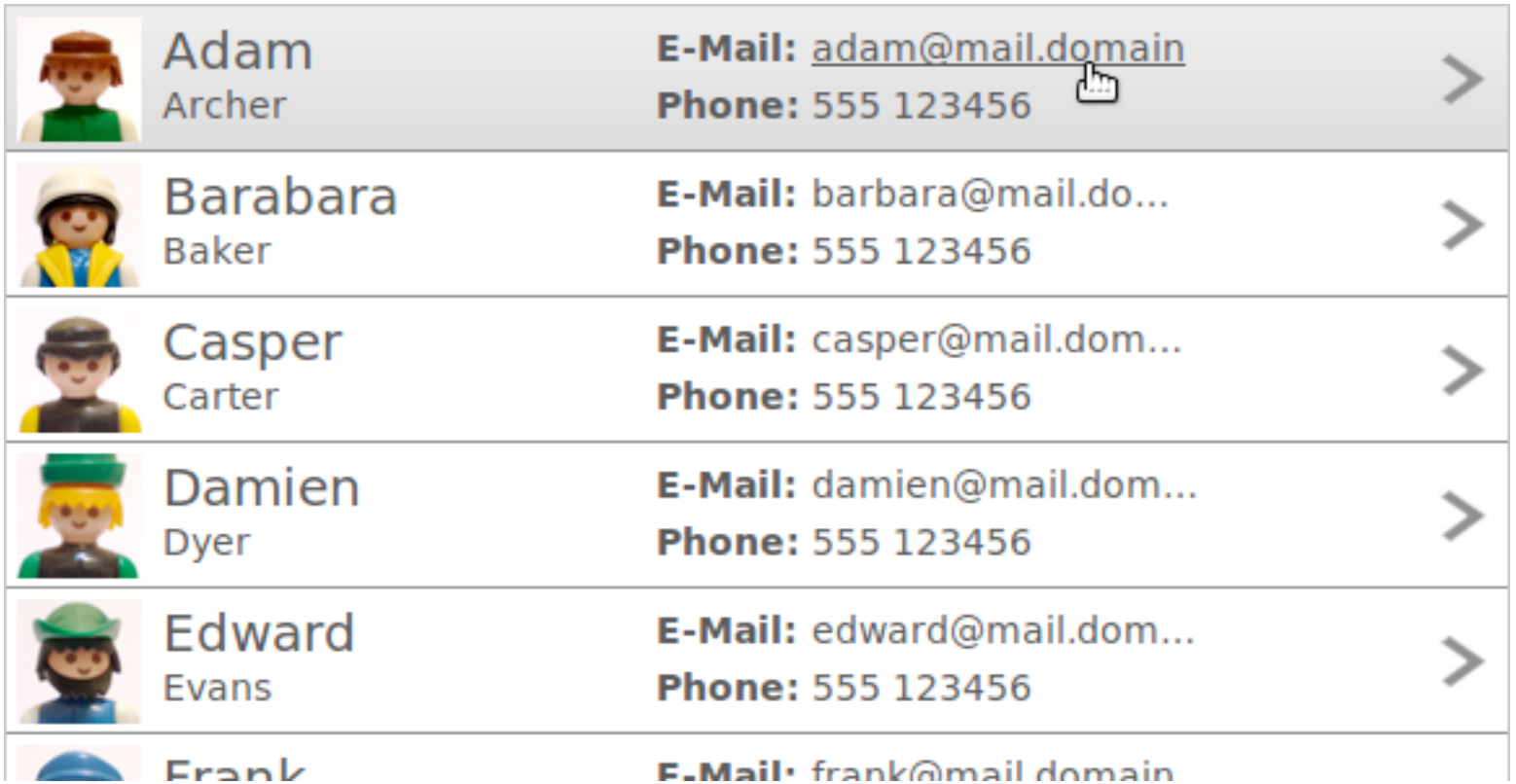


- Keyboard accelerators and mnemonics



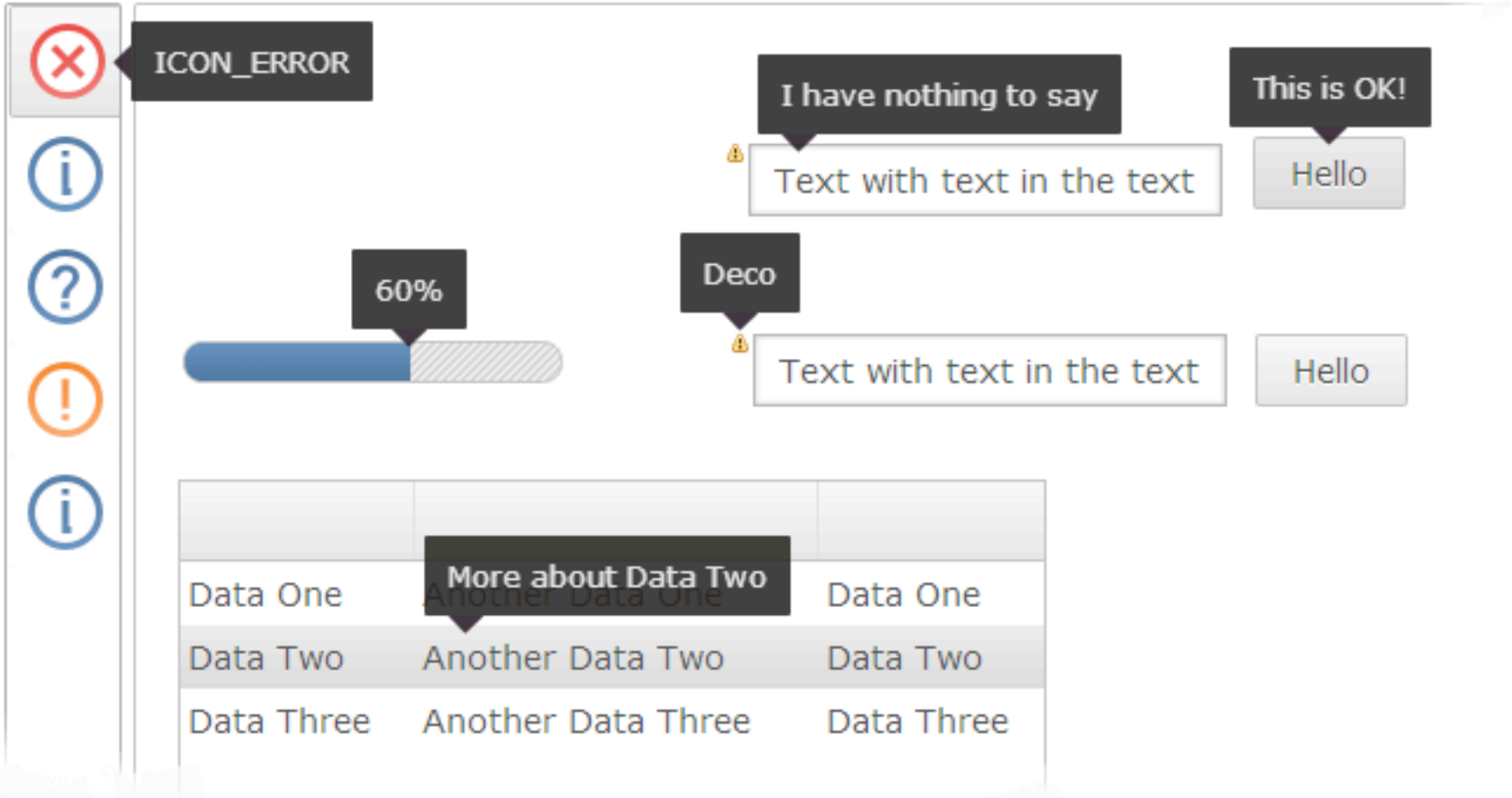
Additional Features – 5.

- Row Templates



Additional Features – 6.

- Tooltips



Additional Features – 7.

- Other important parts
  - Browser History
  - DateTime widget
  - Drag and drop
  - Eclipse Help, Cheet Sheet