This presentation explains the Interaction Services provided by IBM Lotus Expeditor Client for Desktop.

Goal and agenda

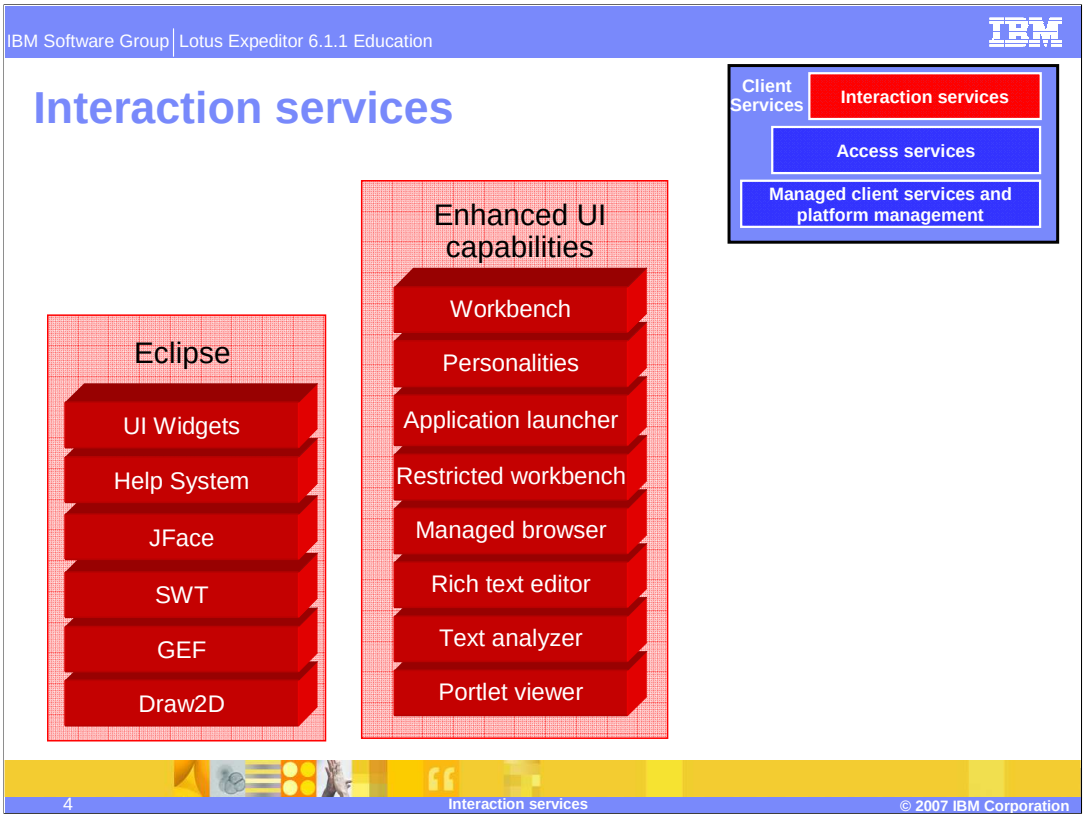
- Understand the Interaction Services provided by IBM Lotus Expeditor Client for Desktop
  - Overview
  - Eclipse user interface technology
  - Workbench
  - Preferences
  - Help system

Since the goal of this presentation is to help you understand the Interaction Services provided by IBM Lotus Expeditor Client for Desktop, we'll give you an overview of the interaction services, talk about Eclipse User Interface Technologies, discuss the work bench, explain how to set preferences, and describe how to contribute help to the help system.

Section

Interaction services

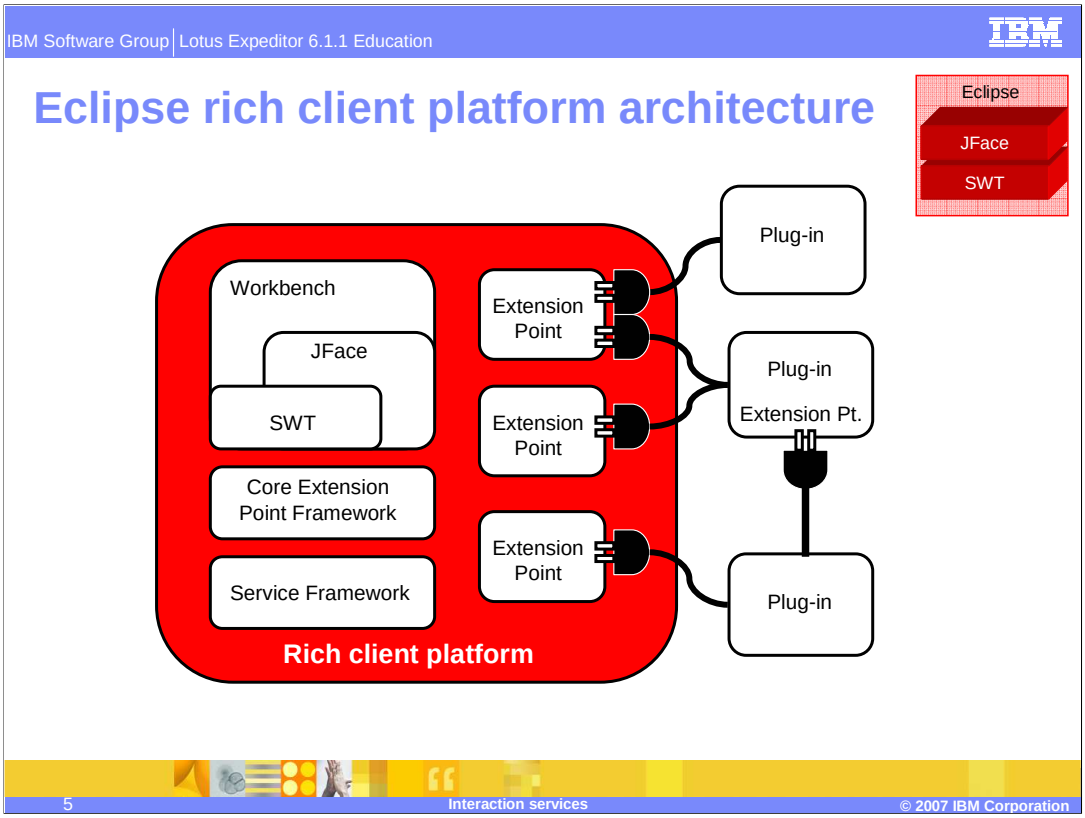
Let’s view the details of the Interaction Services.



The Expeditor Desktop client platform includes key interaction services provided by Eclipse and adds enhanced user-interface capabilities that result in an integrated Workbench window from which your end-users can install, launch and manage one or more applications.

You can deliver rich client applications that use Eclipse to present a graphical user interface and Web applications that enable a Web browser to render the user interface. You can also deliver Portlet applications and Composite applications. Composite applications are applications that enable independently developed components that provide specific business functions to run together in a single application. Additionally, there is support for User Interface widgets, including the Rich Text Editor, the Text Analyzer, the Portlet Viewer and the Managed browser.

Some additional components from Eclipse have been packaged with the Expeditor platform. These are GEF (which stands for Graphical Editing Framework) and Draw2D. Draw2D is a lightweight toolkit consisting of graphical components. Draw2d focuses on efficient painting and layout of figures. The GEF plug-in adds editing on top of Draw2d. The purpose of the GEF framework is to facilitate the display of any model graphically using draw2d figures, support interactions from the mouse, keyboard, or the Workbench, and provide common components related to using the framework.

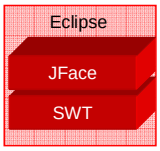


The Eclipse platform is structured around the concept of extension points. Extension points are well-defined places in the system where other components (called plug-ins) can contribute functionality.

Each major subsystem in the platform is itself structured as a set of plug-ins that implement some key function and define extension points. The Eclipse system is built by contributing to the same extension points that third-party plug-in providers can use. Plug-ins can define their own extension points or simply add extensions to the extension points of other plug-ins.

Note that at the core of Eclipse is the OSGi Service Framework.

Eclipse user interface technology



- Standard Widget Toolkit (SWT) provides a cross-platform API that is tightly integrated with the underlying OS GUI for a native look and feel
- JFace toolkit extends and interoperates with SWT to provide classes for handling common user interface programming tasks:
  - **Dialogs, wizards, and rich text editors** define a framework for building complex interactions with the user
  - **Viewers** handle the drudgery of populating, sorting, filtering, and updating widgets
  - **Actions** and **contributions** introduce semantics for defining user actions and specifying where to make them available

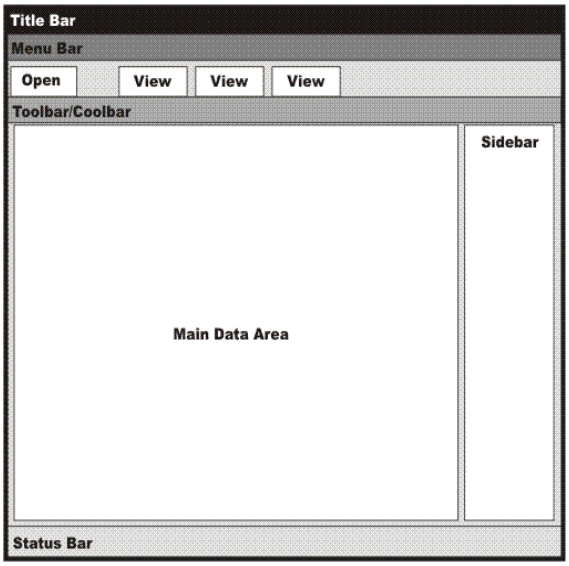
The Standard Widget Toolkit (or SWT) provides a completely platform independent API that is tightly integrated with the operating system's native windowing environment. Java™ widgets actually map to the platform's native widgets. This gives Java applications a look and feel that makes them virtually indistinguishable from native applications. In cases where native function is not provided, the SWT emulates it in a manner in keeping with the platform's normal look and feel. This toolkit overcomes many of the design and implementation trade-offs that developers face when using the Java Abstract Window Toolkit (or AWT) and the Java Foundation Classes (or JFC). AWT gives the least common denominator approach and is therefore functionally limited. JFC is more flexible, but because all widgets are painted by the toolkit, JFC may have trouble precisely emulating a native look and feel.

The JFace toolkit is a platform-independent user interface API that extends and interoperates with the SWT. This library provides a set of components and helper utilities that simplify many of the common tasks in developing SWT user interfaces. For example, it provides the dialogs, wizards, and rich text editors used by the Eclipse IDE. JFace also has tables and trees that utilize a model view controller (or MVC) architecture to separate data access login from data display logic. JFace also provides the mechanisms by which plug-ins programmatically contribute to the workbench.

User interface organization

Workbench

- Title bar
  - Displays program title and icon
- Menu bar
  - Contains set of actions provided by either default workbench or by other applications
- Banner bar
  - Optionally display a graphic and application name
- Launcher (open)
  - Lists running applications, each as an icon, from which users select applications
- Main data area
  - Primary data area, contains the perspectives and views for an application
- Sidebar
  - The sidebar is a view that displays vertically at the side of the workbench window
- Toolbar / Coolbar
  - Optionally displays icons for available actions
- Status bar
  - Used by an application to display its status



This figure in the slide illustrates the organization of the user interface provided by the client platform. The following parts of the user interface are displayed by default: the Title bar, the Menu bar, and the Status bar.

The main data area contains only a default image when the client platform starts. Once applications have been opened, the views associated with the application are displayed.

Application launcher

- The Launcher is represented in the UI as a “Open” button with a drop-down menu
  - Each entry in the drop-down menu is an application that can be launched
  - Populating the launcher
    - Dynamically through extension points
    - Direct calls into the launcher public APIs
  - Creating custom launcher contribution items
  - Contributing bookmarks to the launcher

The launcher is a button control that displays in the workbench area below the main menu. It is labeled with the text **Open**. When clicked, a hierarchical drop-down list of items displays. Users can click an item in the list to start an application, open an application document, or open a bookmark.

You can populate the launcher either dynamically through extensions or through direct calls into the Launcher public APIs located in the platform personality bundle.

The client enables you to open standard items, such as an application or a URL from the launcher by providing a collection of classes that extend the LauncherItemContribution class. You can implement your own custom contribution items.

Bookmarks are user-created references to applications, application documents, or Web sites and are stored within a folder in the launcher.

Application launcher

Application launcher

- The launcher displays available applications that are available to it using the launcherSet extension point
- Support is also provided for these extension points:
  - WctApplication
  - WctWebApplication
  - eRCP
- Displays Eclipse perspectives and content contributed by CAI application catalog
- Can contribute native applications to the launcher by using the launcherSet extension point

The application launcher is enhanced to support application hierarchies. Applications may be placed in folders, such as cascading submenus. For portal-defined perspectives, the application hierarchy maps to the page group hierarchy defined on the portal server. “Native” applications may also be added to the launcher and accessed from the workbench. Native applications launched from the workbench appear in their own windows as they do when launched from the operating system. Windows for applications launched from the workbench will appear in the application switcher. The launcher also supports multiple user workspaces using the command line –data option and launching multiple personalities.

Populating the launcher

Application launcher

- Populating the Launcher
  - Dynamically through extensions
  - Through direct calls into the Launcher public APIs located in the platform personality bundle
- Applications displayed in the launcher are provided using the LauncherSet extension point
  - Other extension points supported:
    - WctApplication extension point
    - WctWebApplication extension point
    - eRCP extension points

You can populate the launcher either dynamically through extensions or through direct calls into the Launcher public APIs located in the platform personality bundle.

Platform integration – Extension points

Workbench

- **com.ibm.rcp.ui.LauncherSet**
  - This extension point defines an application to be launched.
  - Defined types of contributions:
    - urlLaunchItem
    - PerspectiveLaunchItem
    - nativeProgramLaunchItem
- **com.ibm.eswe.workbench.WctApplication**
  - This extension point defines an application to be launched.
  - The client uses this extension point to display the menu items that appear in the **Application > Open** menu, and to display the applications within the Application Launcher
- **com.ibm.eswe.workbench.WctWebApplication**
  - This extension point provides the definition of a Web application to be launched
  - The client uses this extension point to display the menu items that appear in the **Application > Open** menu, and to display the Web applications within the Application Launcher
  - BrowserOptions (optional) - Specifies type of browser to launch, bars and buttons to be displayed in browser, and user ID / password tags

The launcher displays applications that are available to it using the launcherSet extension point. It also provides support for WctApplication, WctWebApplication and eRCP extension points. In addition, it displays eclipse perspectives and content contributed by the programmatic CAI API.

This slide shows the three extension points used to register applications with the workbench.

BrowserOptions are part of the extension point for Web applications so each Web application can optionally specify the browser to launch, the bars and buttons to be displayed in the browser, and the user ID and password tags.

Extension point examples

Workbench

com.ibm.rcp.ui.LauncherSet

```
<extension point="com.ibm.rcp.ui.launcherSet“
id="com.xyz.test.extension.id">
  <LauncherSet
    id="com.xyz.launcher.tests.launcherItems"
    label="Test adding various launcher items">
    <urlLaunchItem
       iconUrl="icons/EmoticonHappy.gif"
       id="com.xyz.launcher.tests.googleLauncherItem"
       label="Test URL Launcher Item - Google"
       url= "http://www.google.com/"/>
  </LauncherSet>
</extension>
```


This example defines a urlLaunchItem that will be added to the launcher.

Extension point examples

Workbench

com.ibm.eswe.workbench.WctApplication

```
<extension point="com.ibm.eswe.workbench.WctApplication">
   <DisplayName>%orderentry.application.name</DisplayName>
   <PerspectiveId>
      com.ibm.pvc.samples.orderentry.richapp.OrderEntryPerspective
   </PerspectiveId>
   <Icon>icons/OEwctapp_32x32.gif</Icon>
</extension>
```

com.ibm.eswe.workbench.WctWebApplication

```
<extension point="com.ibm.eswe.workbench.WctWebApplication">
   <DisplayName>%webapp.name</DisplayName>
   <Url>/OrderEntry</Url>
   <BrowserOptions showAddressbar = “false"
                   showToolbar="false"/>
   <Icon>icons/OEwctwebapp_32x32.gif</Icon>
</extension>
```

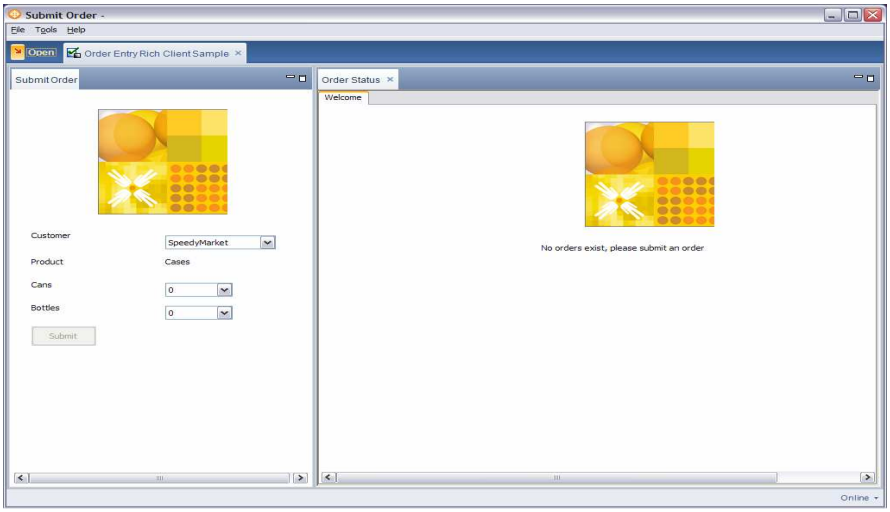

The first example shows an example of how to define a rich client application that can be launched on the Workbench.

The second example shows how to define a Web application that can be launched on the Workbench. This example uses the URL element to specify the Web application URL and removes the browser address bar and tool bar.

Rich client application

Workbench

com.ibm.rcp.ui.LauncherSet (PerspectiveLaunchItem)
com.ibm.eswe.workbench.WctApplication

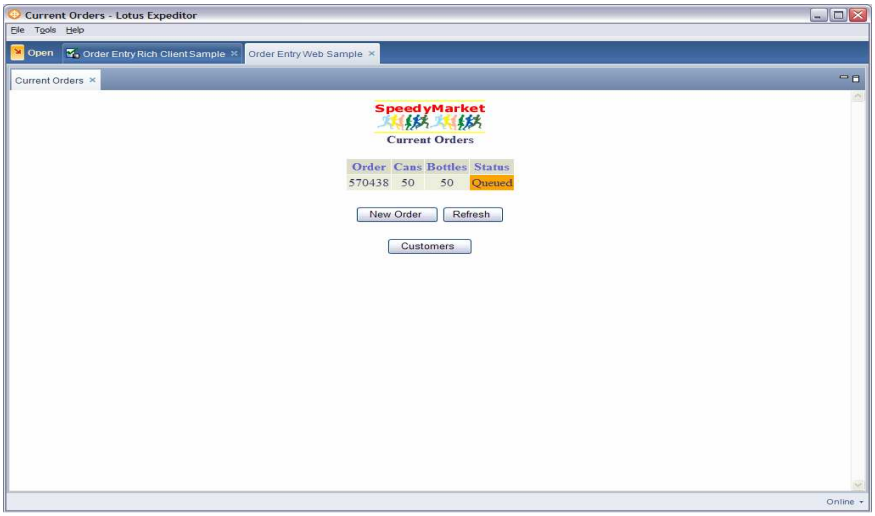


This screen capture shows a rich client application running in the workbench. The application has two views so you can submit orders while simultaneously viewing the status of orders you have already submitted. Notice the application switcher, which allows allow you to switch between two applications.

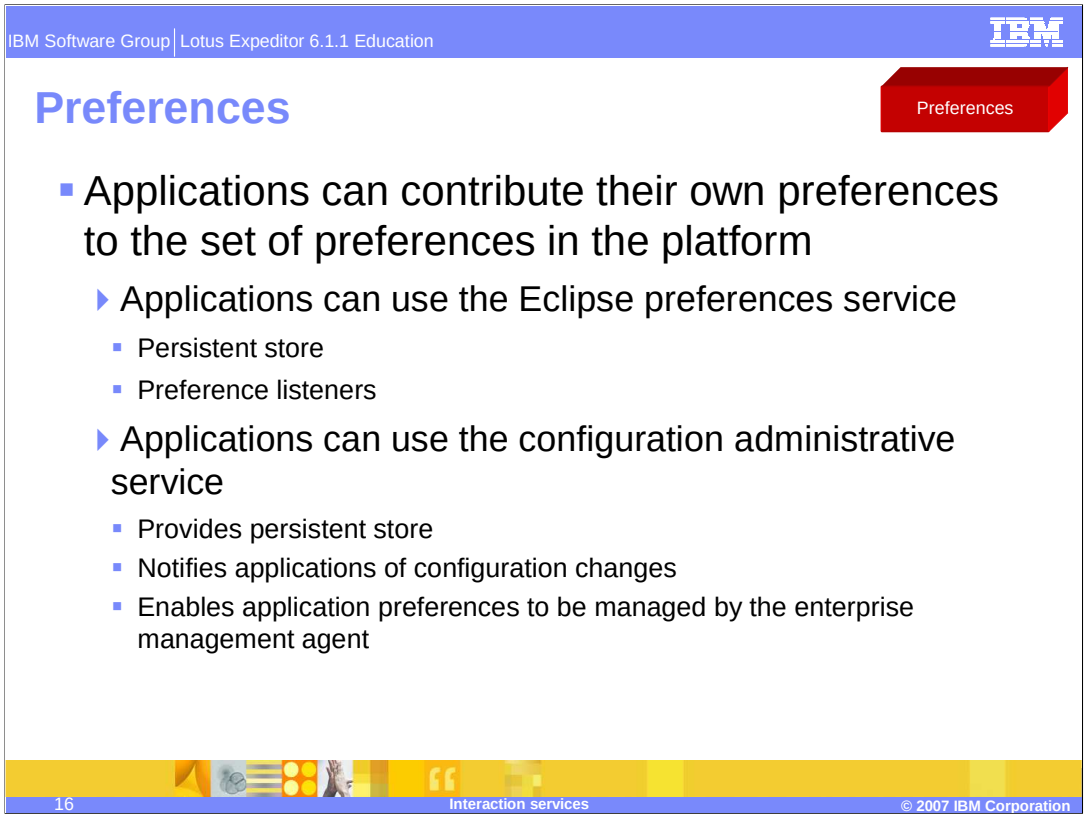
Web application

com.ibm.rcp.ui.LauncherSet (urlLaunchItem)
com.ibm.eswe.workbench.WctWebApplication

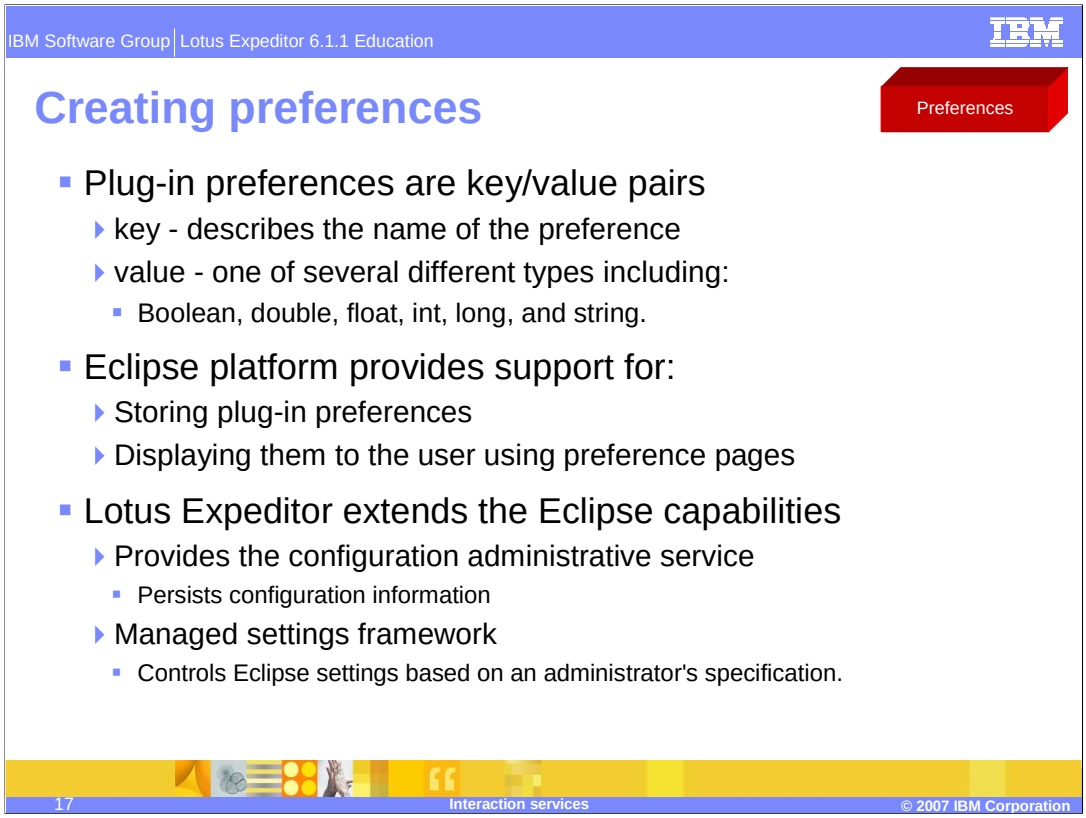
Workbench
Browser customization



This screen capture shows a Web application running in the workbench. Notice that the user experience consists of a single view. In this example, you must go to another view to submit new orders for the current customer or to select another customer.



The Eclipse platform provides support for storing plug-in preferences and displaying them on pages in the workbench Preferences dialog box. The Lotus Expeditor Client for Desktop extends the Eclipse capabilities by including the Configuration Admin service that can persist configuration information. Applications that use Configuration Admin will be notified when configuration information changes. If Configuration Admin is used to store configuration information, system administrators can query and update configuration values using the enterprise management agent.



Plug-in preferences are key/value pairs, where the key describes the name of the preference, and the value is one of several different types, including Boolean, double, float, int, long, and string. The Eclipse platform provides support for storing plug-in preferences and displaying them on pages in the workbench Preferences dialog box.

Lotus Expeditor extends the Eclipse capabilities by including the Configuration Admin service, which can persist configuration information and the Managed Settings framework, which can control Eclipse settings based on an administrator's specification.

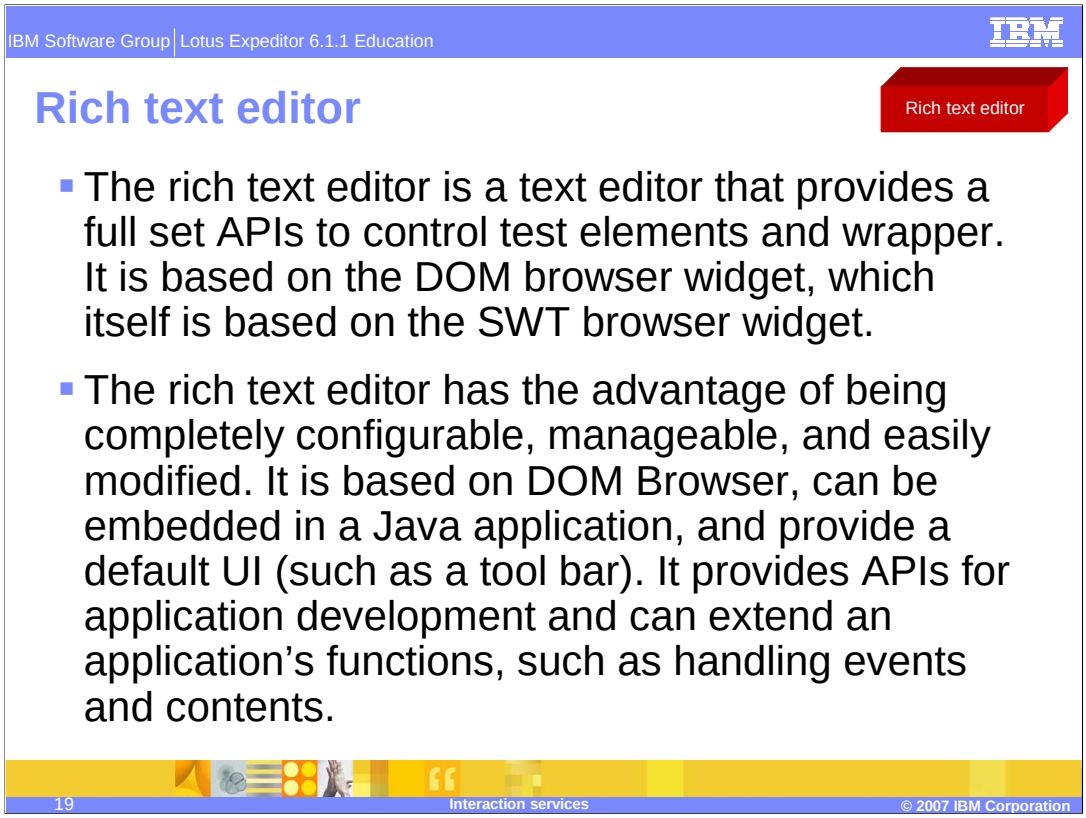
User interface widgets

UI widgets

- Provide a common look and feel for all user interfaces that use the managed client platform
- These widgets provide the following features:
  - Render the toolbar skin, which means it formats the toolbar to make its style consistent with the style defined for the application it displays in
  - Manage drag and drop of items within the toolbar

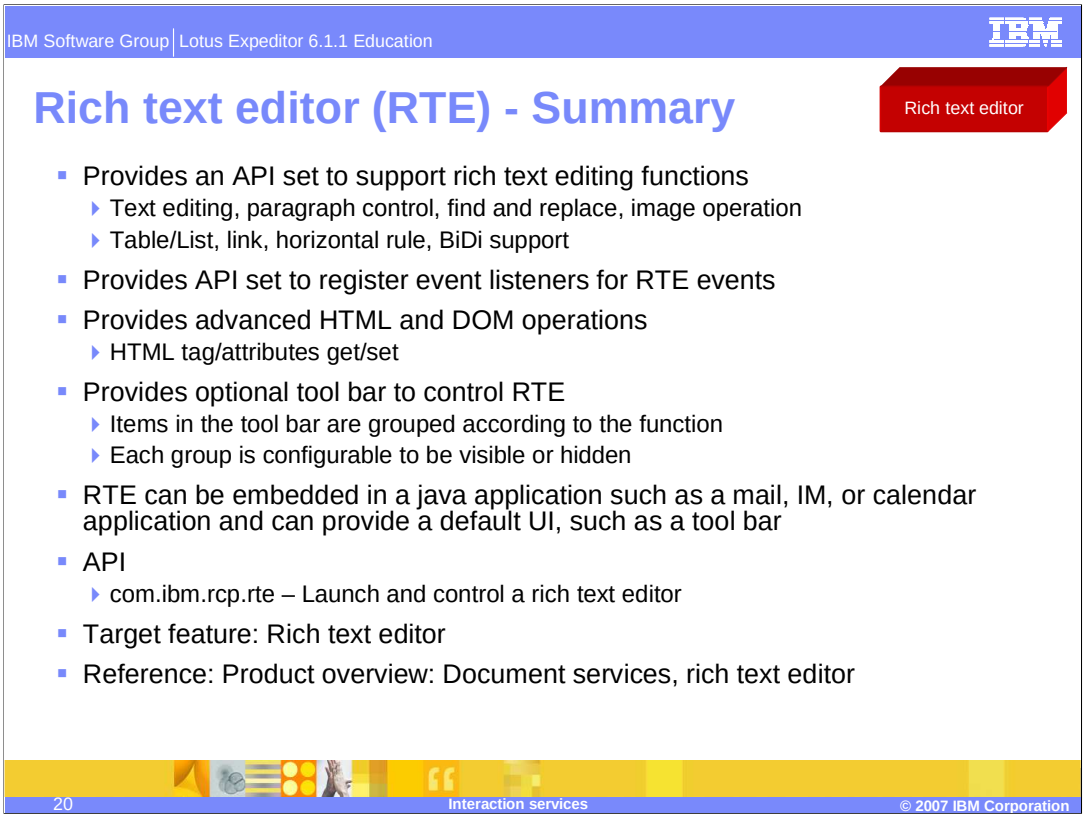
The user interface widgets provide a common look and feel for all user interfaces that use the managed client platform. These widgets provide the following features:

- Render the toolbar skin, which means they format the toolbar to make the style consistent with the style defined for the application it displays in
- Manage drag and drop of items within the toolbar



The rich text editor is a text editor that provides a full set APIs to control the test elements and wrapper. It is based on the DOM browser widget, which itself is based on the SWT browser widget.

The rich text editor has the advantage of being completely configurable, manageable, and easily modified. It is based on the DOM browser, can be embedded in a Java application, and can provide a default user interface (such as a tool bar). The rich text editor provides APIs for application development and can extend an application's functions, such as handling events and contents.



The rich text editor is a DOM-based editor that provides HTML and plain text editing. It is a very thin layer on top of the DOM browser and the editing operation is achieved primarily by using DOM calls.

The rich text editor enables customers to develop their own rich text editor applications or applications that require rich text editing features, such as the ability to integrate a spell checker (called the text analyzer). It provides APIs to support rich text editing function, such as

- Text editing
- Paragraph control
- Find/Replace
- Image operation
- Table/List
- Link
- Horizontal rule
- BiDi support

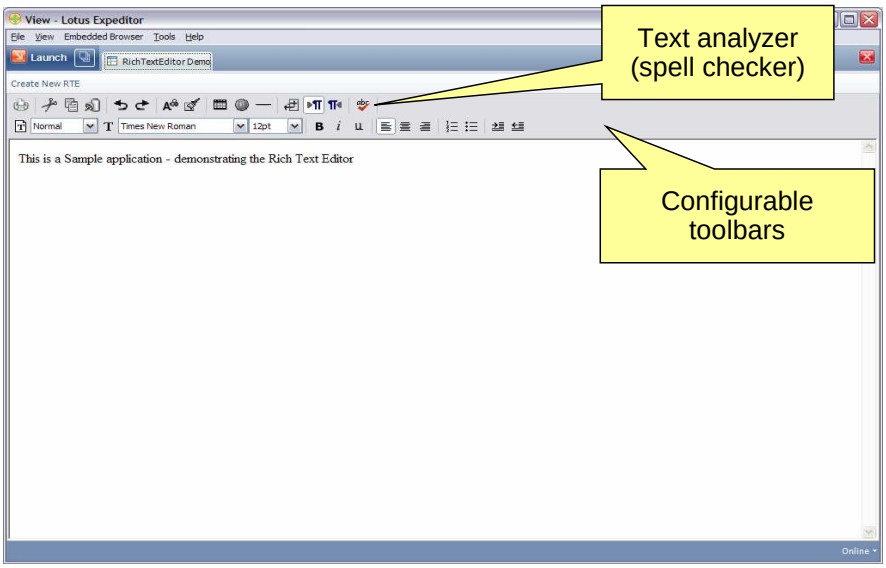
The Rich Text Editor provides advanced HTML and DOM operations, such as:

- HTML format content operations
- HTML tag/attributes
- And it can operate on the DOM Element presentation of a document directly

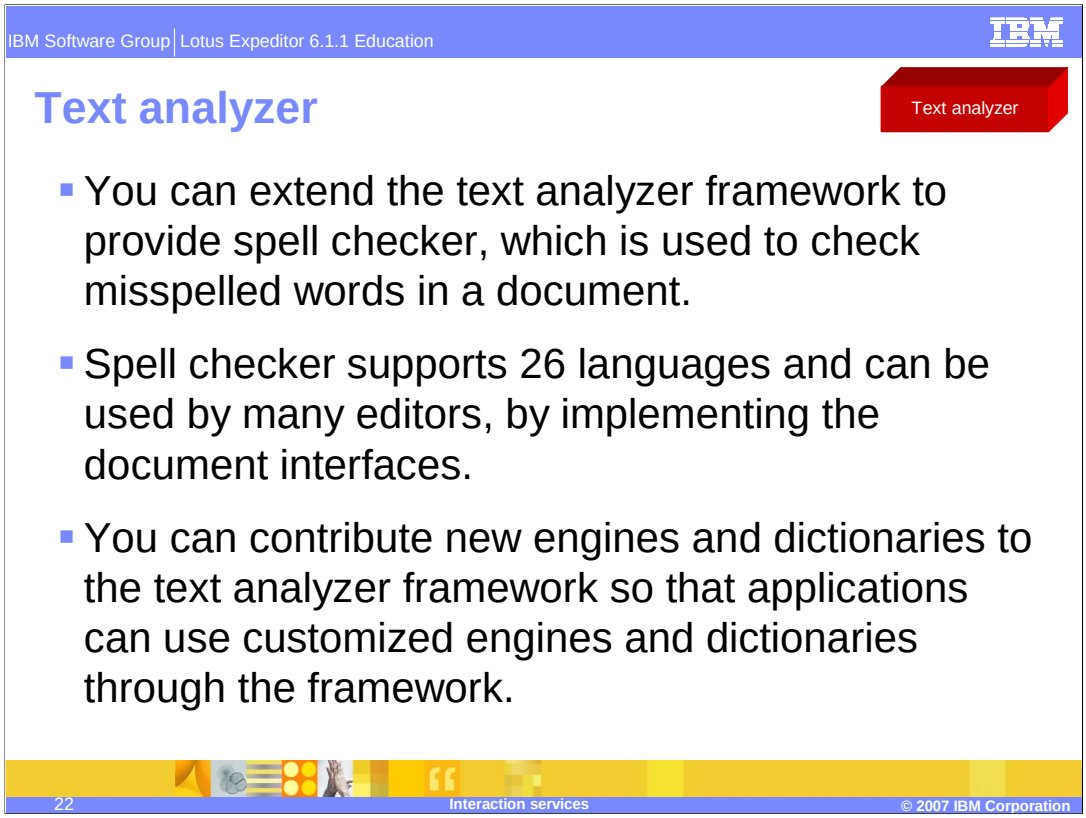
RTE provides a wrapper as a user interface. Included is an optional tool bar to control RTE. Items in the tool bar are classified in several groups according to the function. Each group is configurable to be visible or hidden.

Rich text editor sample

Rich text editor



This slide shows a screen capture of the rich text editor screen sample.



You can extend the Text Analyzer framework to provide Spell Checker, which is used to check misspelled words in a document.

Spell checker supports 26 languages and can be used by many editors by implementing the document interfaces. You can contribute new engines and dictionaries to the Text Analyzer framework so that applications can use customized engines and dictionaries through the framework.

Use the com.ibm.rcp.textanalyzer.Dictionaries extension point for applications to contribute new dictionaries.

Use the com.ibm.rcp.textanalyzer.Engines extension point for applications to contribute new engines.

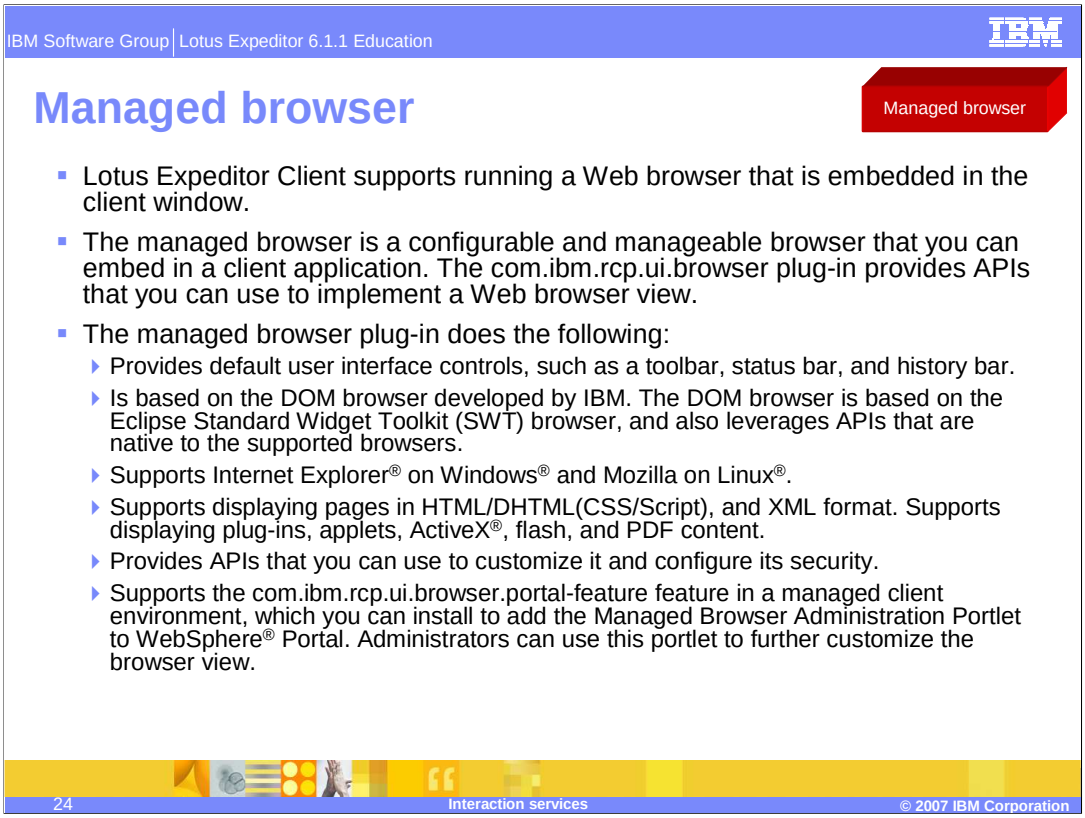
Text analyzer - Summary

Text analyzer

- Performs spell checking for misspelled words
- Provides JFrost and POE engines, as well as, a default set of dictionaries
- Provides the framework to add your own engine and dictionaries
- API
  - com.ibm.rcp.textanalyzer – Provides common text analyzer constants and methods
  - com.ibm.rcp.textanalyzer.dictionarymanager – Manages dictionaries provided by different spell checker engines
  - com.ibm.rcp.textanalyzer.spellchecker – Checks for misspelled words in editors
  - com.ibm.rcp.textanalyzer.udm – Manages user dictionaries
- Extension points:
  - com.ibm.rcp.textanalyzer.DictionaryTarget – Add more dictionaries
  - com.ibm.rcp.textanalyzer.Engine – Contribute spell check engines
- Target feature: Common text analyzer framework
- Reference: Adding spell checking to applications

In summary, Spell Checker is used to check misspelled words in a document. It supports 26 languages.

Dictionaries can be provided which are specific to a professional field, medical or legal, for example.



The Lotus Expeditor Client supports running a Web browser that is embedded in the client window.

The Managed Browser is a configurable and manageable browser that you can embed in a client application. The com.ibm.rcp.ui.browser plug-in provides APIs that you can use to implement a Web browser view.

The Managed Browser plug-in provides default user interface controls, such as a toolbar, status bar, and history bar. It is based on the DOM Browser developed by IBM. The DOM Browser is based on the Eclipse Standard Widget Toolkit Browser, and leverages APIs that are native to the supported browsers. The Managed Browser plug-in supports Internet Explorer on Windows and Mozilla on Linux and supports displaying pages in HTML/DHTML(CSS/Script), and XML format. It supports displaying plug-ins, applets, ActiveX, flash, and PDF content and provides APIs that you can use to customize it and configure its security. The Managed browser plug-in supports the com.ibm.rcp.ui.browser.portal-feature in a managed client environment, which you can install to add the Managed Browser Administration Portlet to WebSphere Portal. Administrators can use this portlet to further customize the browser view.

Managed browser - Summary

Managed Browser

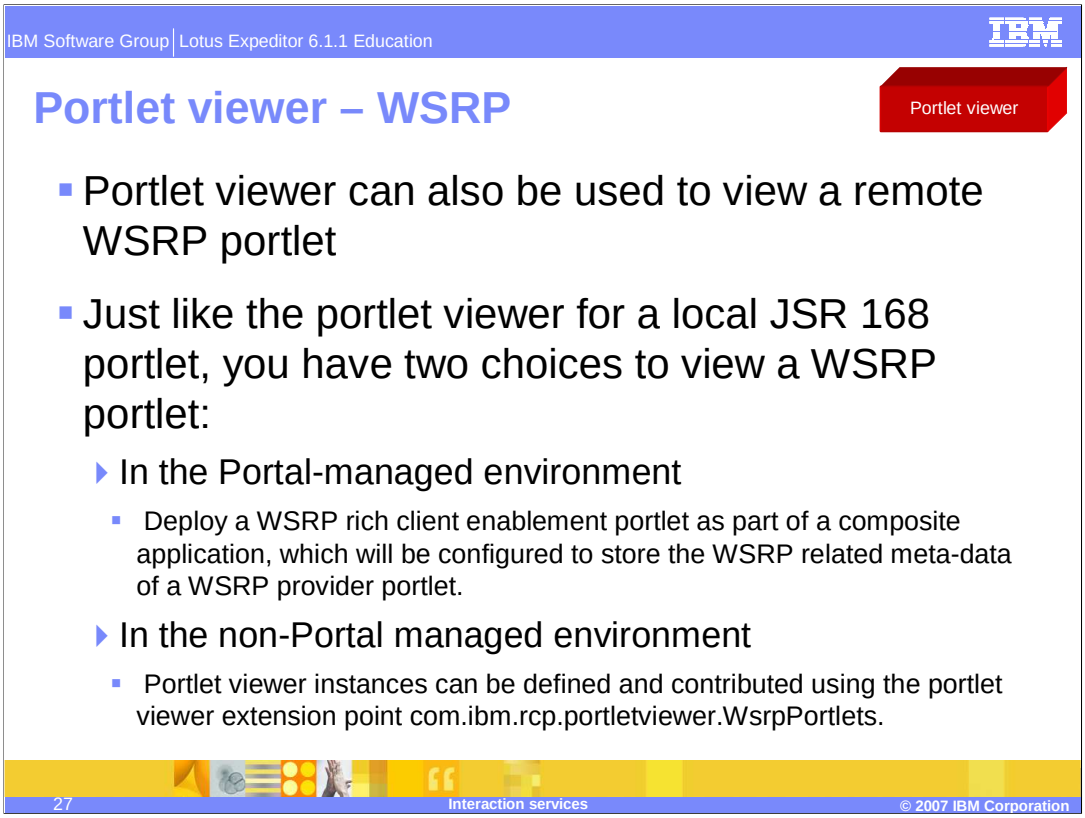
- Based on the Eclipse SWT browser widget
- Browsers supported
  - Windows - MS IE
  - RHEL 4 - Mozilla 1.7
  - NLD 9 – Mozilla 1.7
- Expose APIs to allow applications to control the browser view, configurable controls Application customization using configuration file (plugin.xml)
- Supports being embedded in a Java application, provides default UI such as tool bar and controls
- APIs:
  - com.ibm.rcp.ui.browser – provides configurable managed Web browser
  - com.ibm.rcp.ui.browser.bookmarks – provides bookmark management
- Extensions:
  - com.ibm.rcp.ui.browser.BrowserEditor
  - com.ibm.rcp.ui.browser.BrowserView
- Target feature: Browser component
- Reference: Product overview

The Managed Browser extends an application's functions, such as launching the browser, handling events, and managing preferences for the user interface. The Managed Browser is based on the SWT browser widget in Eclipse. It can support Internet Explorer on Windows and Mozilla on Linux.

The Portlet Viewer is an Eclipse view wrapper for a JSR 168 portlet. It consists of a SWT browser instance whose URL points to a JSR 168 portlet deployed onto the Portlet Container. The main benefit of the Portlet Viewer is that it provides a unified rich client component for the composite application framework for portlet presentation and interaction.

A Portlet Viewer instance can be instantiated and contributed to the Lotus® Expeditor platform in two ways:

- In the Portal-managed environment, portlet information will be stored in the Composite Application XML file and passed down to the Lotus Expeditor platform through the Composite Application Infrastructure. The portlet information from the CA XML file will be translated to a Portlet Viewer instance by the Topology Handler. It is important to note that the portlets are deployed as part of a composite application and that the Portlet Viewer instances are merely views within the application perspective.
- In the non-Portal managed environment, the Portlet Viewer instances can be defined and contributed using the Portlet Viewer extension point com.ibm.rcp.portletviewer.portlets.



The Portlet Viewer can also be used to view a remote WSRP portlet through the WSRP feature in Lotus® Expeditor. Just like the Portlet viewer for a local JSR 168 portlet, you have two choices to view a WSRP portlet:

In the Portal-managed environment, you can deploy a WSRP rich client enablement portlet as part of a composite application, which will be configured to store the WSRP related metadata of a WSRP provider portlet. The WSRP rich client enablement portlet will export the WSRP metadata into the Composite Application XML file and pass it down to the Lotus Expeditor platform through the Composite Application Infrastructure. The metadata will be used by the portlet viewer to start the WSRP facility in Lotus Expeditor and present the WSRP provider portlet.

In the non Portal managed environment, the Portlet Viewer instances can be defined and contributed using the Portlet Viewer extension point com.ibm.rcp.portletviewer.WsrpPortlets.

Portlet viewer - Summary

Portlet Viewer

- Display portlet output fragment in an Eclipse view using SWT browser
  - Display portlet modes (EDIT, VIEW, HELP) as view actions and portlet dynamic title as view title
  - Supports JSR 168 Portlet and WSRP (Web Services Remote Portlets)
- Extension Points
  - com.ibm.rcp.portletviewer.wsrp.wsrpPortlets - contains data for WSRP Portlet.
  - com.ibm.rcp.portletviewer.controllers – contribute a new controller
  - com.ibm.rcp.portletviewer.portletdata – provides the data for the portlet viewer to display the portlet
- Target feature: Portlet viewer
- Reference: Developing portlet applications

In Lotus Expeditor, the portlet container provides the runtime for the JSR 168 portlets.

The portlet container serves the portlet requests from the clients and usually returns the portlet fragment.

In Eclipse or non-headless environment, the Portlet Viewer is used to display a portlet fragment.

The Portlet Viewer is an Eclipse SWT view that contains a SWT browser instance that will make the portlet request to the portlet container and display the portlet fragment.

Since the portlet supports different modes, the viewer also displays the modes of the portlet on its window.

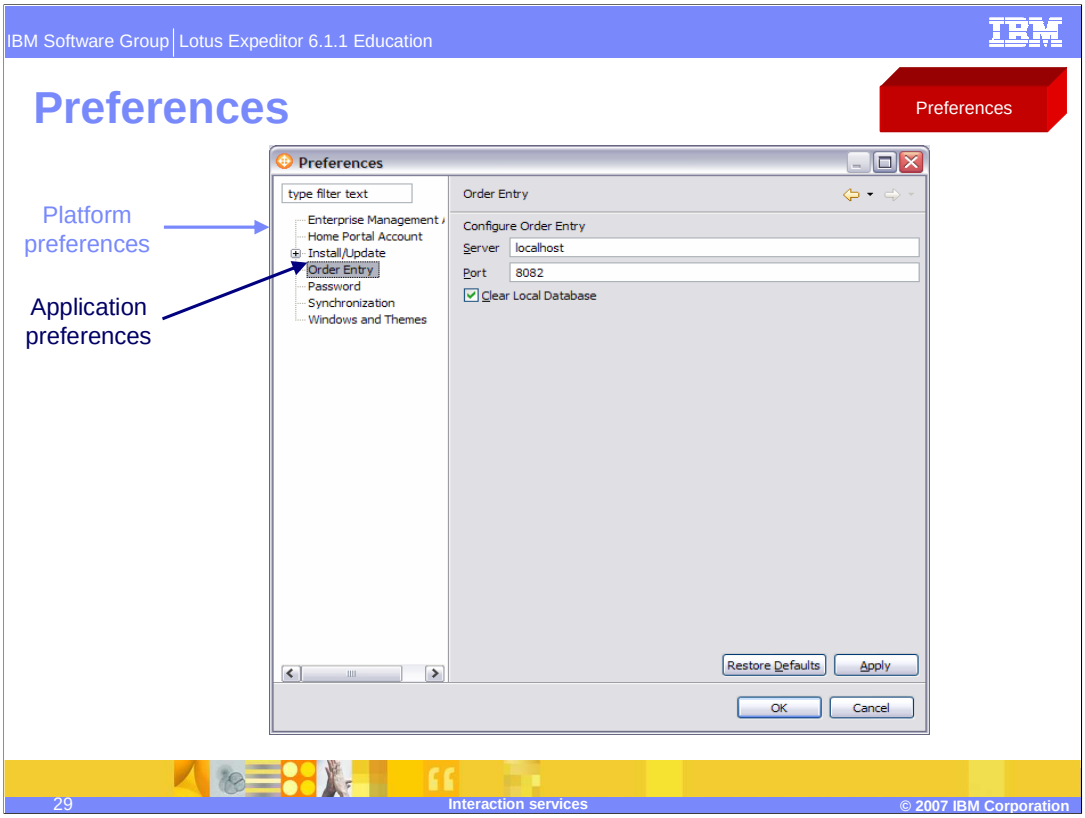
Furthermore, it is also an EventHandler of the OSGi Event Admin.

The portlet viewer is also suitable for displaying WSRP portlet.

The Expeditor platform provides WSRP consumer functionality.

WSRP is a presentation-orientated Web service interface of portal server.

During runtime, the WSRP consumer contains placeholder portlets, which will interact with remote portlets, and retrieve the markup fragments from them. The portlet viewer’s mechanism can also be used to display WSRP portlet, too.



This slide shows an example of an application, called the Order Entry application. The application displays preferences in the workbench.

Help framework

Help System

- Applications can contribute plug-ins to provide help content
- Help system aggregates help contents

If you are creating a set of Help information for your application, and you intend on using the built-in Lotus Expeditor help plug-ins, you should use the Eclipse PDE to create a help plug-in. The Help plug-in provides for XML configuration of the Table of Contents and content specified as HTML.

For more information on creating a help plug-in, refer to the **Plug-in Help** section in the *Platform Plug-in Developer's Guide* located in the Eclipse Help system.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject= Feedback about xpdv6.1.1_interaction_services.ppt

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM Lotus WebSphere

ActiveX, Internet Explorer, Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.