

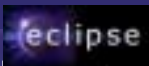


Scaling Large Eclipse Applications Progressively

Tod Creasey
IBM Eclipse Platform Team
tod_creasey@ca.ibm.com

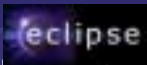
Agenda

- Problem Description
- What is already in Eclipse
- Progressive Disclosure using Activities
- Other Scalability Features in 2.1 and 3.0



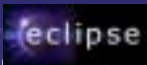
The Scalability Problem in Eclipse: Users

- Some large applications ship over 300 preference pages and 150 views
- Confusing for new user
- Needs to be powerful for standard user
- Most users only use portions of the functionality regularly
- Needs to be configurable at the product level



The Scalability Problem in Eclipse: Developers

- Needs to be configurable at the product rather than plug-in level
- Size of the workbench is very large on large apps
- Still need functionality for plug-in prerequisites but the user needs less clutter



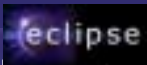
What Eclipse Provided in 2.1

- Action Sets
 - Refine searches
 - Filter visible items
- Categories
 - Allows for better grouping
 - Keeps the initial list small
- Still not enough for very large applications



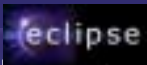
New in 3.0: Rich Client Platform

- Eclipse itself is broken into smaller pieces
- Dynamic loading of plug-ins allows for gradual additions of functionality
- Large Applications are still not reduced in size enough by this due to plug-in dependencies



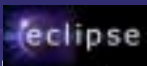
New in 3.0: Progressive Disclosure

- Products can define and group contributions into activities
- Dynamic plug-ins can load and unload plug-ins no longer in use
- Functionality is filtered until it is discovered
- Discovery is done via trigger points

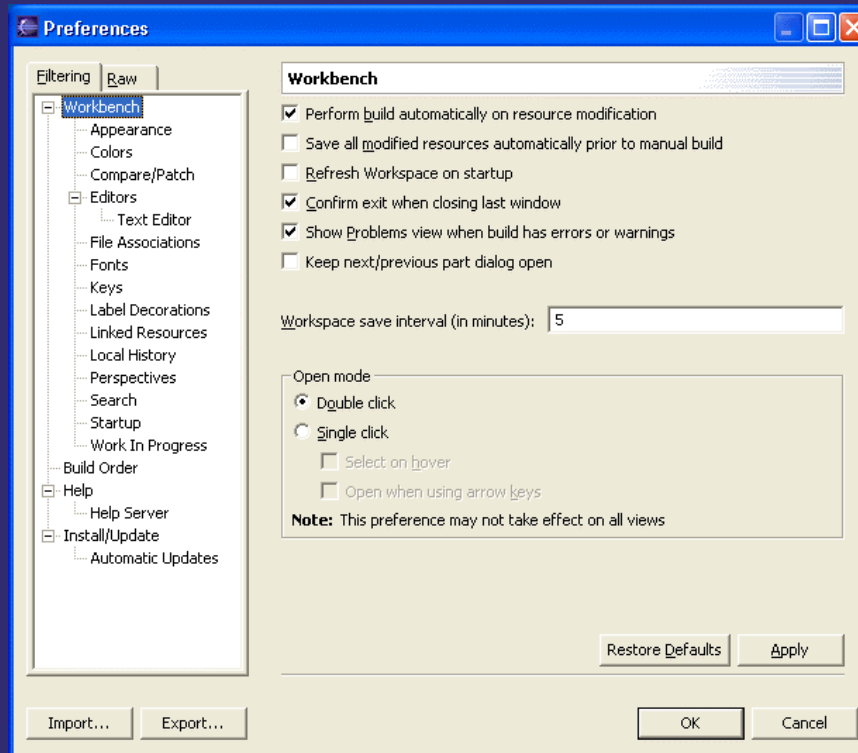


Progressive Disclosure: Filtering Functionality

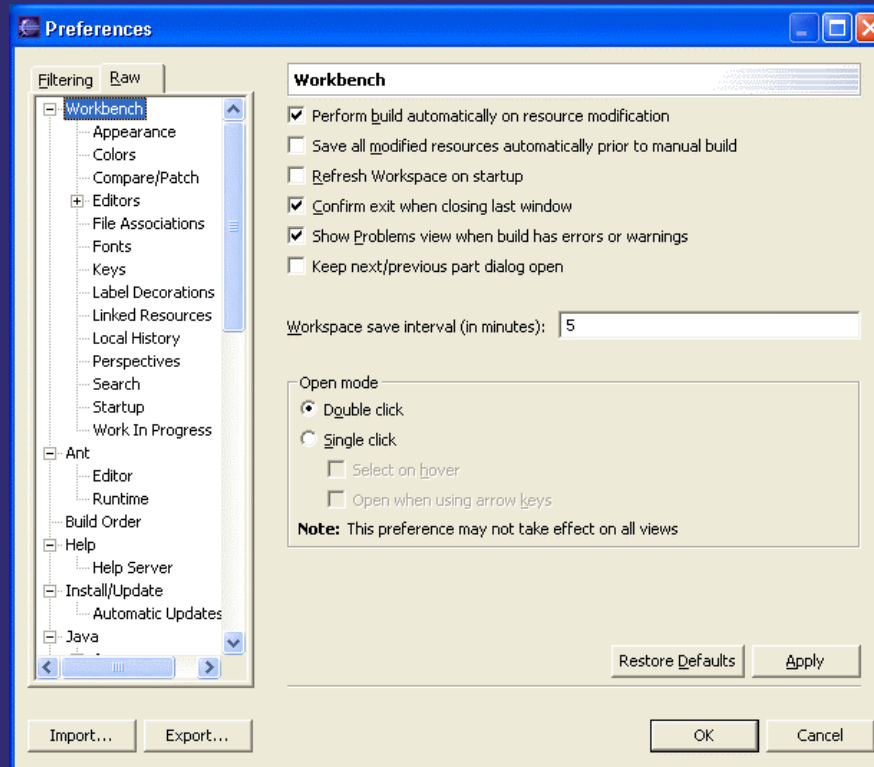
- Want the user to get the right choice quickly
- Large applications require too much digging
- Start with initial choices based on user input
- Expand choices as trigger points are hit



Progressive Disclosure: Filtered Preferences

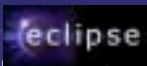


Progressive Disclosure: Unfiltered Preferences



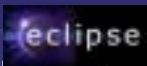
Progressive Disclosure: Filtering Functionality

- Currently filter
 - New Wizards
 - Import Wizards
 - Export Wizards
 - Show Views list
 - Open Perspective list
 - Toolbar contributions
 - Menu contributions
 - Perspective pages
 - Property pages
 - Editors

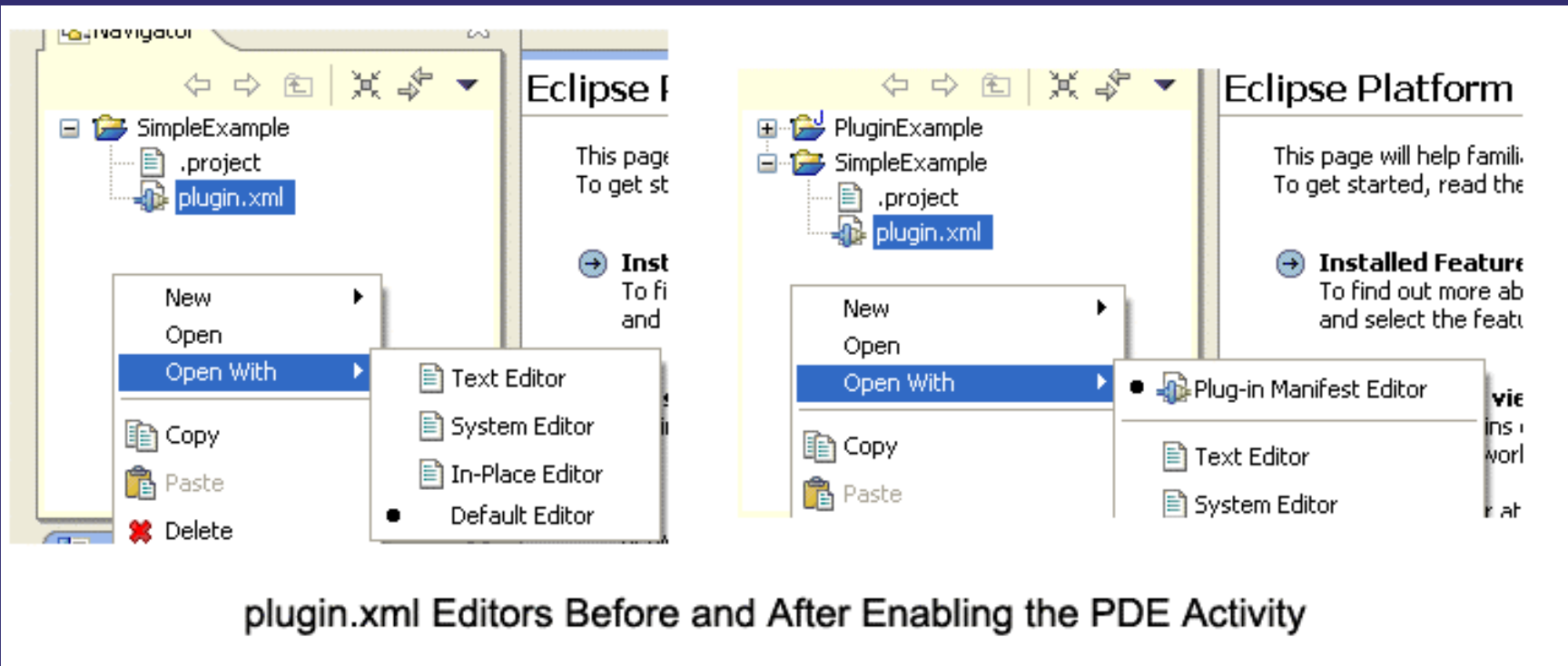


Progressive Disclosure: Trigger Points

- User will choose initial functionality (potentially based on welcome page)
- Creating or loading projects that are tied to an activity enables the activity
- Currently trigger points are:
 - New wizards
 - Import wizards
 - Loading a project with a nature

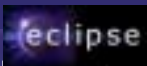


Progressive Disclosure: Trigger Points

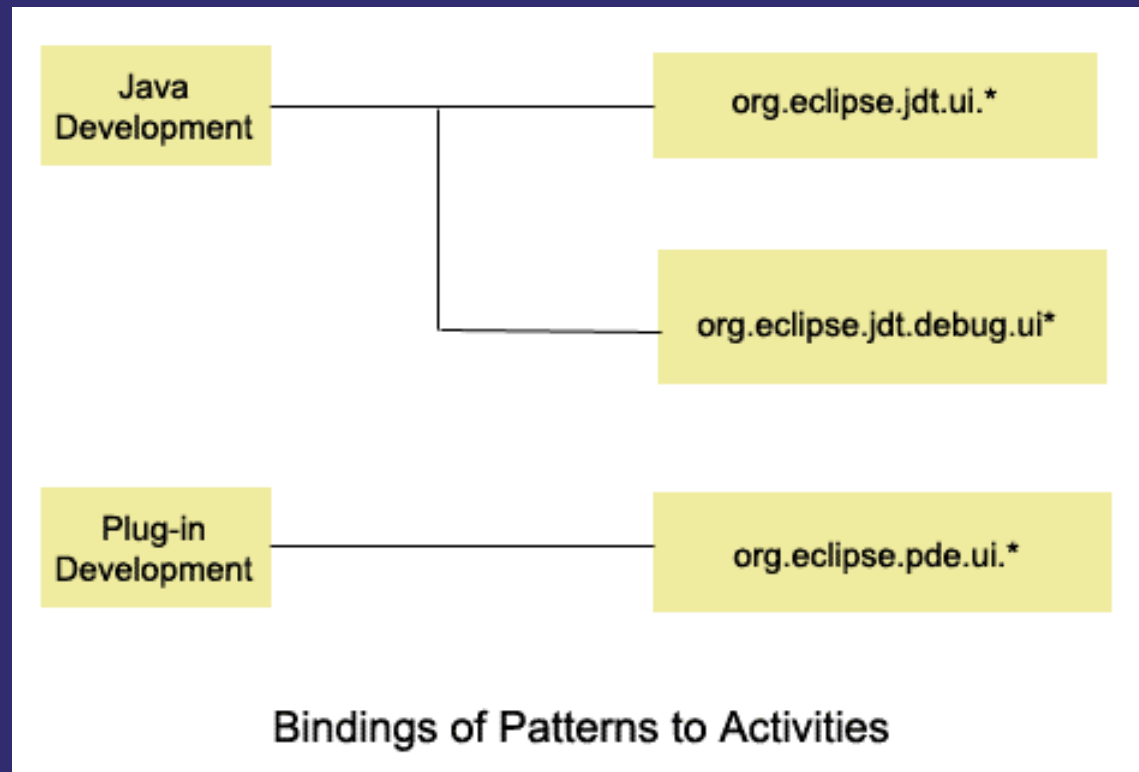


Defining Activities using Plug-ins

- Use the `org.eclipse.ui.activities` extension point
- Pattern bindings - the association of regular expressions to activities. Plug-in id + local id are both used in most supplied functionality.
- Can define activities and categories of activities at the product/system level. Plug-ins do not force the decision.
- Public API available for other systems to participate in filtering

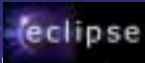


Example Pattern Bindings



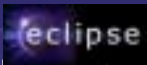
What we are not doing

- Not going to try and define all activities at the plug-in level
- Not disabling functionality
- Not preventing use of the code by other plug-ins
- This is just a filtering feature!



Other Features That Address Scalability

- Contexts
- New Look
- Background Jobs
- Definition of fonts and colors through extension points



Reducing the Clutter: Contexts

- Contexts solve a different problem
- Contexts are a mechanism for adding more dynamic functionality
- Debug is a good example – certain commands only interesting while debugging
- Not necessarily related to any of the activities filtering



Reducing the Clutter: New Look

- New Look enables toolbars on current focus view
- Tries to reduce the busy look of the workbench
- Want to draw users attention to the current task



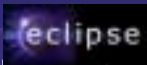
Moving to the Background

- Job framework added for 3.0
- Does the scheduling and thread management for the whole workbench
- Locking mechanism allows for finer grained blocking – do not need to lock entire workbench for just one resource
- Support mostly at the Core level
- See Jean-Michel Lemieux's talk for more detail



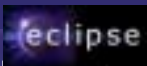
Coming in 3.0: What's there now

- Filtering of functionality
- Workbench trigger points
- API for adding activity support to new systems
- Filtering of commands using contexts
- Extension points for colors and fonts



Coming in 3.0: What is coming

- Activity based initial user experience
- More fine grained support for job status in views
- New look
- Most of this should be ready for M7



Demo

- IBM and WebSphere are trademarks of International Business Machines 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.
- Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Other company, product, service names, and logos may be trademarks or service marks of others.

