

Editor Focused Workbench

Tom Eicher, Christof Marti, Daniel Megert IBM OTI Labs Zurich

- When writing code, you want to:
 - see the context of the edited source code,
 - have proposals of possible completions,
 - avoid typing mistakes and syntax errors,
 - avoid entering the same text repeatedly,
 - minimize scrolling, dragging, clicking.
- When browsing code, you want to:
 - directly navigate type and call hierarchies,
 - quickly jump to a specific type or method,
 - get an overview of a source file,
 - view relevant code section at a glance,
 - navigate back and forth in the browsing history.
- When drinking coffee, you want to:
 - have a big mug,
 - increase the caffeine level.



Java - TestCase.java - Eclipse Platform	uning Dug IIIndau IInda						2
Edit Source Refactor Navigate Search P		A _ * /_ /					
		1.664					_
Package Explorer 🗸 🗙	🕐 *TestCase, java X 🔄 TestResult, java			E Outline			
수 수 🖻 🛛 🗶 🔽	64 * }		^	📕 👌 📭 🛛) 🗙 o 🖌 🦆		
🖃 🙀 JUnit	65 *			ju	nit.framework		
🗄 🖶 junit.awtui	66 * @see TestResult			📃 🗄 📲 im	port declarations		
🗄 🖶 junit.extensions	67 * @see TestSuite			📄 🖻 🗝 🖸 A Te	estCase		
🚊 🌐 junit.framework	68 */				fName : String		
🕀 🕗 🚺 Assert.java	69				C TestCase()		
吏 🚺 AssertionFailedError.java	70 public abstract class TestCase extends Assert implemen				C TestCase(String)		
吏 近 ComparisonFailure.java	71 /**)		
🛨 🗻 Protectable.java	72 * the name of the test case			· · · · •	createResult()		
🛨 🚺 Test.java	73 */			•) getName()		
🛨 近 TestCase.java	74 private String fName;				run()		
吏 🗊 TestFailure.java	75		-		🗛 run(TestResult) 🗌		
吏 🚺 TestListener.java	76 /**			•) runBare()		
吏 🚺 TestResult.java	77 * No-arg constructor to enable serialization. Thi			· · · · •	vunTest()		
🛨 🔄 TestSuite.java	78 * is not intended to be used by mere mortals with				setName(String)		
🗄 🐅 junit.runner	79 */			· · · · •	setUp()		
🗉 🚮 junit.samples	80 public TestCase() {			· · · · •	tearDown()		
🗉 🌐 junit.samples.money	81 fName= null;				🔺 toString()		
🕀 🚮 junit.swingui	82 }						
吏 🔂 junit.swingui.icons	83 /**						
吏 🌐 junit.tests	84 * Constructs a test case with the given name.						
🖅 🖶 junit.tests.extensions	85 */						
吏 🖶 junit.tests.framework	86 public TestCase (String	g name) {					
🗉 🚮 junit.tests.runner	& 87 fName= na;						
吏 🖶 junit.textui	88 }						
🕀 🚻 JRE System Library [jdk1.4.2]	89 /** 90 * Counts the number (
📖 🖹 cpl-v10.html	90 ^ Counts the number (DI test cas	es executed by run				
	91 7		¥				
			>				
	Noblems (9 items)				8	≱ ⊳ ▼	,
	Description	Resource	In Folder	Location			
	O The import junit.runner is never used	-	JUnit/junit/runner	line 4			
	O The import junit.runner is never used	Sorter.java	JUnit/junit/runner	line 5			
	O The local variable result is never read	SimpleTest.j JUnit/junit/samples TestSuitePa JUnit/junit/swingui	line 49				
	🔇 Unnecessary semicolon		line 150				
	Unnecessary cast to type Test for expres		JUnit/junit/swingui	line 102			
	The local variable h is never read Problems Tasks Console	TextRunner	1Unit/iunit/tests/runner	line 34			
Package Explorer Hierarchy							

Classic Workbench Setup

The classic workbench setup allocates a **small area** of the application window to the editor. Parts of the relevant code **remain hidden**. Some of the displayed views are **not always needed**. Unproductive activities like scrolling, managing views, and manipulating the UI are **too frequent**.

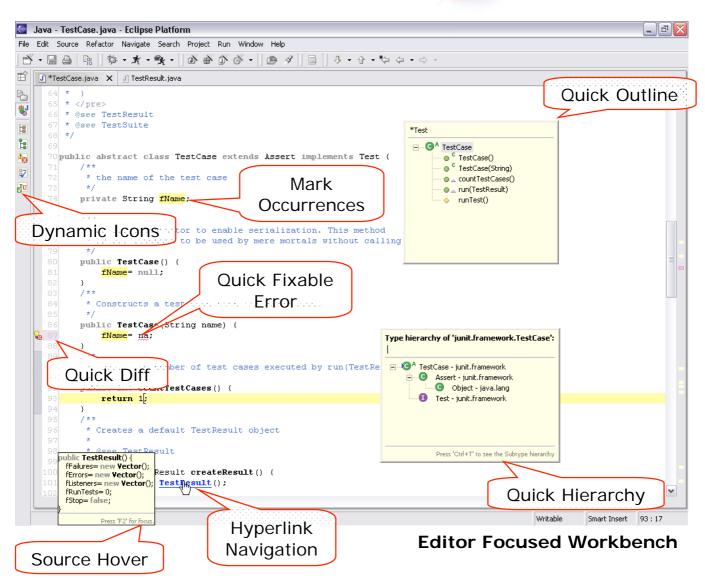


During development you can maximize the editor area by removing persistent views and reducing dialogs. To compensate for the resulting loss of context information, Eclipse offers a growing set of specialized UI elements in the following areas:

In-place context information

- Quick navigation views
 - Keyboard accessible Fast Views
 - Quick views: Outline, Type / Method Hierarchy ^{3.0}
- Condense information
 - Dynamic icons
 - Quick Diff 3.0
 - Dynamic error annotation
- Lightweight access to reference information
 - Dynamically mark occurrences 3.0
 - Javadoc and source hovers
- In-place interactions
 - Code input and transformation
 - Content Assist
 - Local rename
 - Code templates
 - Quick Assist
 - Navigation
 - Filterable pop-up dialogs
 - Hyperlink navigation
 - Go to next / previous annotation





An editor focused workbench configured to use **Fast Views** and **in-place controls** offers a lean user interface that lets you concentrate on writing code with **efficient browsing tools** and **precise context information** just one keystroke away.

Outlook:

- Folding of code blocks (e.g., Javadoc comments, method bodies)
- Flat Outline showing all inherited members