IBM Rational Software Analyzer 的靜態分析：入門

程式碼靜態分析如何爲您與客戶節省時間和金錢

本文是四篇系列文章之一，旨在介紹 IBM® Rational® Software Analyzer，以及 Rational® Application Developer 與 Rational® Software Architect 當中的靜態分析功能。本文以概括、籠統的方式介紹靜態分析，其他系列文章則著重於如何使用與延伸 Analyzer 的部分功能。若要探索本軟體提供的所有功能，以上文章可協助您更快學會中階與進階功能。

科技業人士通常對自身的專業頗具信心，也確實爲頂尖的研發人才，然而，無論我們的教育程度多高、資歷多深，再怎麼盡心盡力，所撰寫的程式碼仍會有錯誤。大部分的軟體應用程式已變得相當複雜，要瞭解真正的需求，或撰寫原始碼以因應這些需求，顯得困難重重。要完成工作，卻不發生任何錯誤或明確缺陷（通常稱爲 bug），幾乎是不可能的。

請想想，您組織中的重要任務應用程式，幾乎都是由數百個類別與數千行程式碼所構成。爲協助研發人員瞭解與實作如此複雜的系統，我們開發了一些新技術，例如敏捷開發程序。然而，即使運用短開發循環與對等程式碼複查，銷售給客戶的軟體依然潛伏著大量缺陷。儘管目前方向正確，業界現有的做法仍有不足，而越來越需要自動分析工具的協助。

業界使用動態分析工具已長達數十年，工程師都會使用執行時期除錯器來爲程式碼除錯，或使用程式碼側寫程式，來發現效能焦點。不過在開發週期中，我們使用這些工具的時間點太晚，以致於不符合成本效益； 找出問題最適當的時間點，是在寫完程式碼後進行複查之時。有了靜態分析工具的協助，許多吃力工作將能自動處理。然而別忘了，靜態分析只是用來提升程式碼品質的一項工具，無法完全取代人工程式碼複查。

本文將爲您介紹 IBM® Rational® Software Analyzer 靜態分析工具，以利用自動程式碼分析減輕工作量，並說明此工具可爲軟體開發程序提供哪些優點。Rational Software Analyzer 可促進並簡化自動改善程式碼品質的程序，最初只用來建立靜態分析工具，並將其整合至其他的 Rational 產品，例如 Rational® Application Developer 與 Rational® Software Architect 的 API 與使用者介面，但現在已發展爲完整且獨立的產品。Analyzer 可讓開發人員在執行日常開發程序時，輕鬆存取某些功能，例如自動程式碼複查與結構分析。

什麼是「靜態分析」

靜態分析的意義不盡相同，放眼產品市場，許多公司都宣稱提供靜態分析工具。市場之所以能支持這麼多公司，是因爲靜態分析的概念相當廣泛，有些公司側重 C++ 程式碼複查，也有些公司只提供適用於 Java 程式設計語言的軟體衡量指標。有些人分析程式碼，是爲了 Web 應用程

式的安全問題，有些人掃描程式碼，則是爲了找出相依關係的問題。因此，靜態分析是一種多元而分歧的概念，需要進一步釐清。

究竟何謂靜態分析？*靜態分析*是指研究不會變動的事物，就軟體用語而言，可定義爲研究目前尚未執行的原始碼或二位元碼。您知道需要除錯器或側寫程式，以分析執行中的程式碼，然而從未執行程式的程式碼中，也有不少玄機。

舉例而言，如果您爲程式剖析所有的原始檔，即可確保原始碼符合預先定義的編碼標準。您也可以偵測一般效能問題，例如多次呼叫某方法，即使產生的結果不變。您還可以檢查每個類別的匯入項目，以瞭解該類別依附於哪些其他類別，或哪些其他類別依附於該類別。以上都不需要執行程式，甚至無需編譯程式。

靜態分析有許多種類，我們可以根據其提供值，將靜態分析分爲幾個常見種類。除了表 1 所列的種類外，當然還有許多其他類型的靜態分析，但表 1 針對 Rational Software Analyzer 相關系列文章的靜態分析主題，彙集了重要類型與形式。

表 1：常見的靜態分析種類

類型	值
程式碼複查	此類工具通常用來執行自動程式碼剖析，會透過剖析器載入與傳遞原始檔，以尋找違反已建立規則集的特定程式碼型樣。在 C++ 等語言中，這些規則許多會建置在編譯器之中，或存在於外部程式例如 Lint。在 Java 等語言中，編譯器的功能和自動程式碼複查大不相同。程式碼複查非常適合用來施行編碼標準、尋找基本效能問題，並尋找潛在 API 濫用。程式碼複查也可以納入較深入的分析形式，例如資料流、控制流及鍵入狀態等等，本系列其他文章，將介紹其中部分形式。
程式碼相依關係	相對於檢查個別原始檔的格式，程式碼相依關係工具檢查的是原始檔之間的關係（通常是類別關係），以便針對整體程式架構建立對映。相依關係工具通常用來找出程式碼中已知的設計型樣（良好型樣）或反型樣（不佳的型樣）。
程式碼複雜性	複雜性工具會分析程式碼，並與已建立的軟體衡量指標相比較，以決定程式碼是否需要如此複雜。假如特定的程式碼片段超出給定的臨界值，就會標幟爲重構候選者，以改善可維護性。
趨勢分析	趨勢分析不會直接使用程式碼構件，而是根據其他分析形式，研究程式碼品質究竟提升還是下降（基本上是在分析其他分析結果）。趨勢分析工具的分析結果，通常訴求的是經理、高階主管與客戶，而不是開發人員，因其主要指出品質改善方向，也就是回答：「程式碼的品質是變好還是變差？」

靜態分析的優點

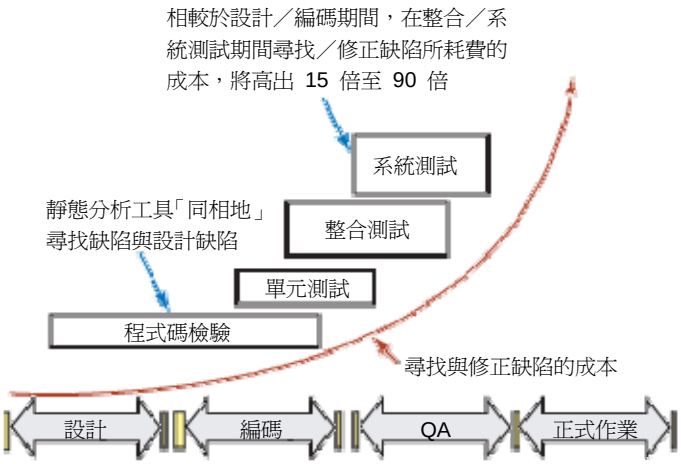
爲何要在您的開發程序中納入靜態分析，本文已提到部分原因。現在讓我們再次重申，進行靜態分析的兩個基本而有力的理由：爲了節約時間與節省金錢。使用靜態分析工具，可明顯省下大量時間：可讓您以更少時間，獲得品質更好的程式碼。許多研究，包括在 IBM 內部進行的研究顯示，即使是簡易的自動程式碼複查，都可以爲程式碼找出 5% 到 15% 的缺陷。

研究還顯示，若您的客戶發現任何缺陷，每個缺陷的代價是 $12,000 到 $18,000 美元。想想一般大型軟體在整個壽命期間，會有數千個缺陷，使用自動程式碼複查工具，便能省下 60 萬到 270 萬美元之多。無論您重視百分比還是金額數目，靜態分析工具的省錢功力確實驚人。

當然，避免讓客戶發現缺陷，是節省成本的最佳方法。您只要利用綜合性測試程序，通常就能達成這項目標，然而使用程式碼複查等靜態分析工具，將可讓您省下更多成本。

我們看過許多類似圖 1 所示的圖形，卻很少人說明其背後的緣由。在開發程序當中，發現缺陷的時機愈早，能省下的成本就愈多。而且只要使用簡易的自動程式碼複查，即可在專案的編碼階段開始尋找缺陷，甚至還可提前至開發人員鍵入程式碼之時。

圖 1：在開發生命週期的不同階段修正缺陷的成本比較



節省還有另一個面向，只是效果可能並不明顯。目前爲止，我們都把焦點放在開發人員如何利用靜態分析工具節省時間與金錢，那麼您的客戶又如何呢？事實證明，如果您的軟體讓客戶耗費過多時間或金錢，客戶可能會提出告訴。您希望客戶購買與使用您的軟體，因爲您的產品可靠實用，若將靜態分析新增至開發程序，客戶也能同時獲得時間與金錢的優勢。因爲高品質的程式碼，表示客戶不用浪費時間等待您修正他們報告的缺陷，獲利能力也不會因此受到影響。

Rational Software Analyzer 可用以滿足下列需求：

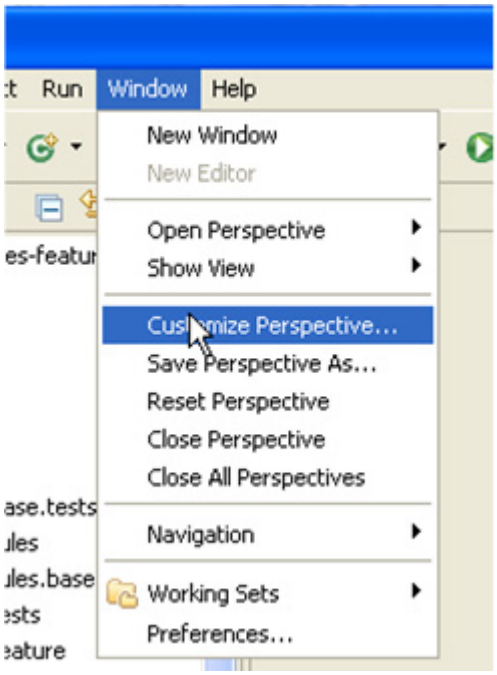
- 首先，它緊密整合 Eclipse、Rational Software Architect 或 Rational Application Developer 工作台，並提供完整存取權限給開發人員，撰寫程式碼時便能加以分析。
- 其次，Rational Software Analyzer 提供指令行或 ANT 作業格式，以支援與現有建置系統的整合。完整的 API 不但讓您使用內建分析技術，還可自行建立分析技術。
- 最後，可取出分析結果並產生報表，報表可以採用工作台格式與匯出 HTML 等格式，以便開發人員、經理與高階主管評估整體的程式碼品質。

指定用來執行分析的規則

Rational Software Analyzer 在工作台中可供使用時，您可以在 Java、Debug、C++ 與 Plug-in Development 視景中，看到新的功能表和工具列選項。其他視景中，您可能需要手動啓用這些功能：

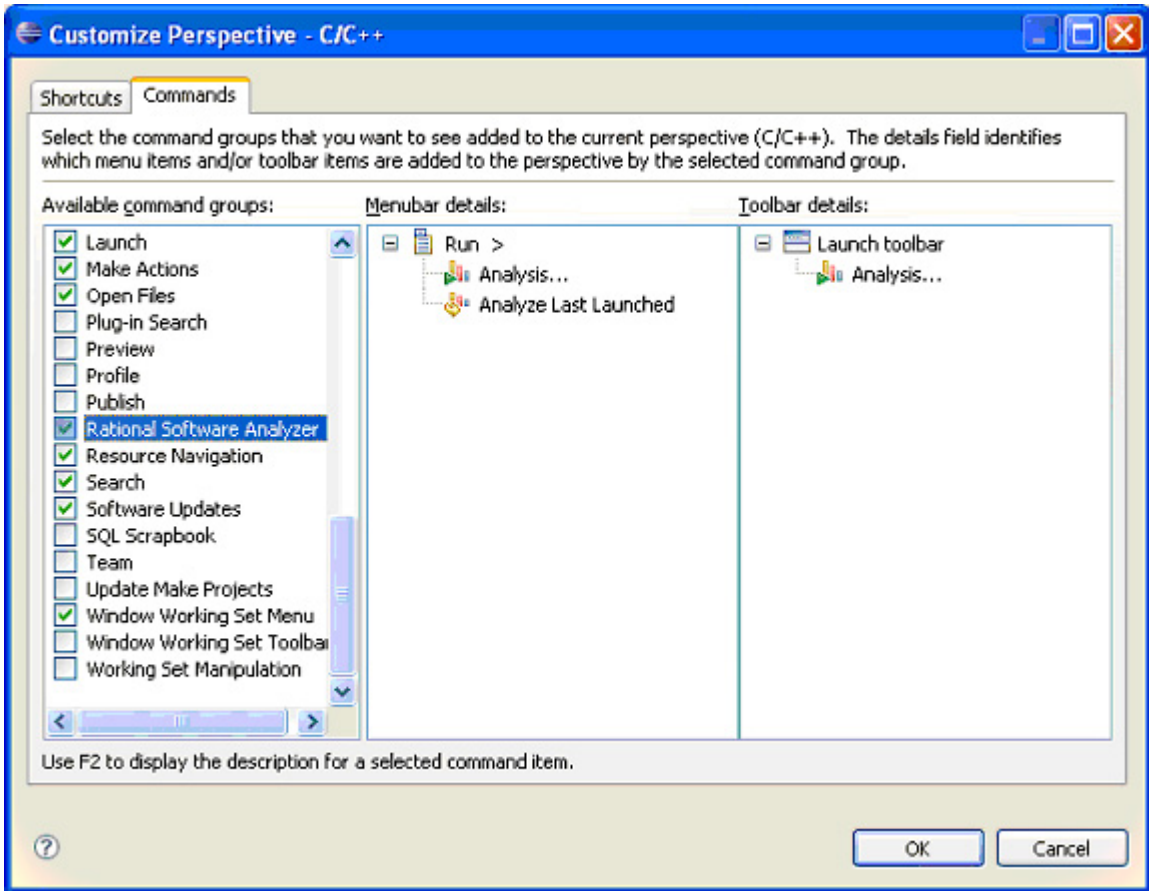
1. 從 Eclipse 功能表列選取 **Window > Customize Perspective**（請參閱圖 2）。

圖 2：自訂視景選項



2. 對話框出現時，請選取 **Commands** 標籤，並按一下 **Rational Software Analyzer** 勾選框，
3. 然後按一下 **OK** 以接受變更（圖 3）。

圖 3：Commands 標籤視圖



除了 Eclipse 工具列與功能表，您還可以看到新的靜態分析，可利用這些選項，建立、修改或執行分析配置。

4. 選取 **Run > Analysis** 功能表選項，以顯示主要的分析配置對話框。

您會看到一個對話框，類似從 Eclipse 工作台執行或除錯程式碼所用的對話框。爲了簡化，我們將其運作方式設計成類似現有的對話框畫面，您可以使用對話框左上角的按鈕，新增或移除分析配置。如名稱所示，配置是用來決定分析形式、要執行的規則，以及分析範圍（例如專案、工作集或整個工作區）。

5. 要開始使用，請選取配置左側的 **Analysis** 元素，然後按一下 **New** 按鈕。

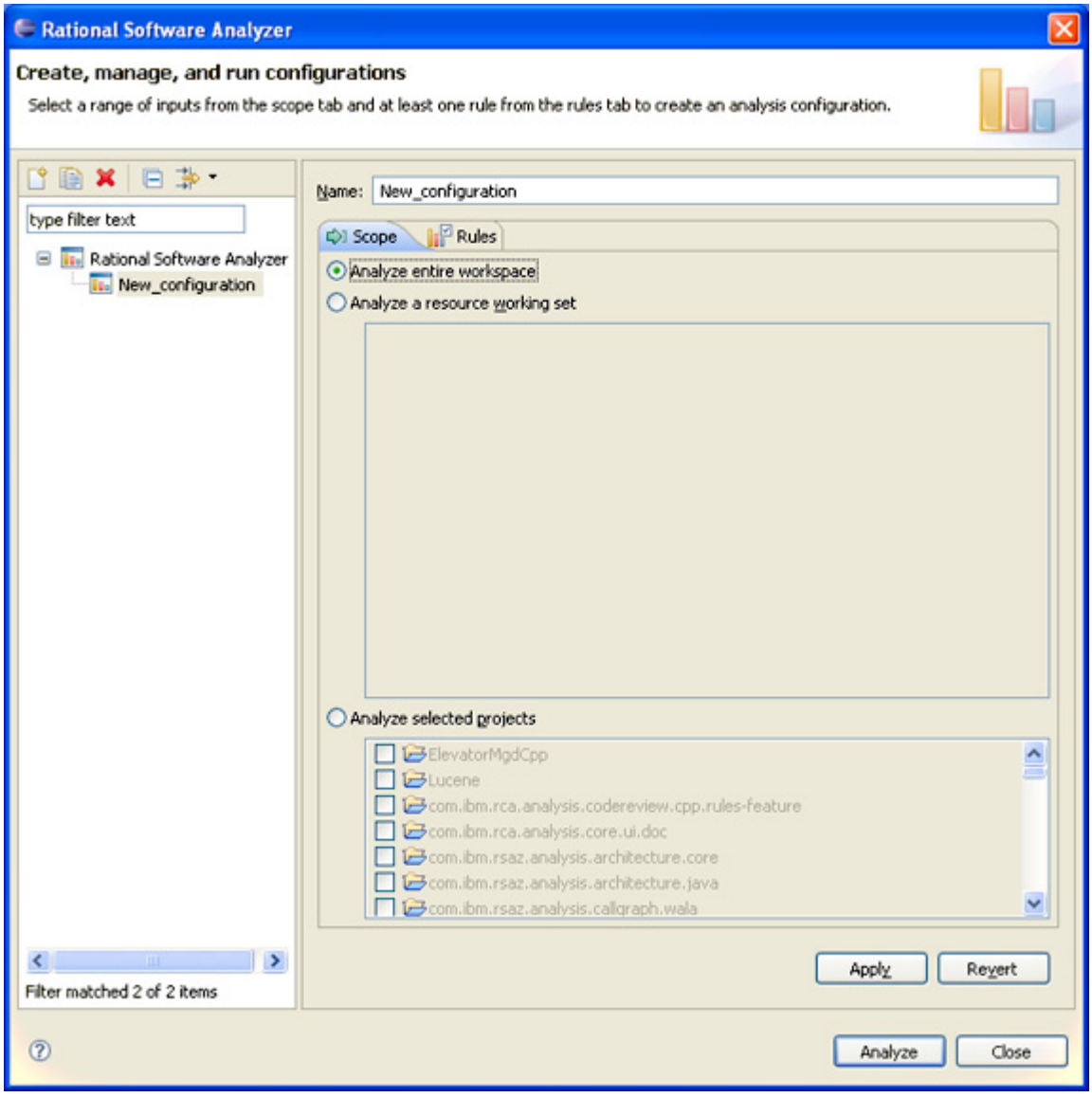
您會發現對話框右側，變成顯示基本配置的介面。

配置分析

建立分析配置的第一個步驟，是指定所要分析資源的預設範圍。您可以在 Scope 標籤中選取屬意的範圍，目前可用的選項包括整個工作區、工作集或專案集。本練習：

1. 建立新配置，並在 **Scope** 標籤中選取 **Analyze entire workspace** 選項，如圖 4 所示。

圖 4：指定分析範圍



在 **Rules** 標籤中，指定您希望執行的分析形式。您會發現，此標籤會顯示目錄（樹狀結構），您可於此選取或取消選取分析元素。此外，此標籤還包含其他按鈕，可用以匯入與匯出規則選項。網域樹狀結構最頂端的節點是分析提供者，代表分析架構所認可的分析工具類型。提供者包含一些種類，是鬆散的規則組織或其他種類。規則會定義要在分析期間產生的條件，以執行程序中所有吃重工作。

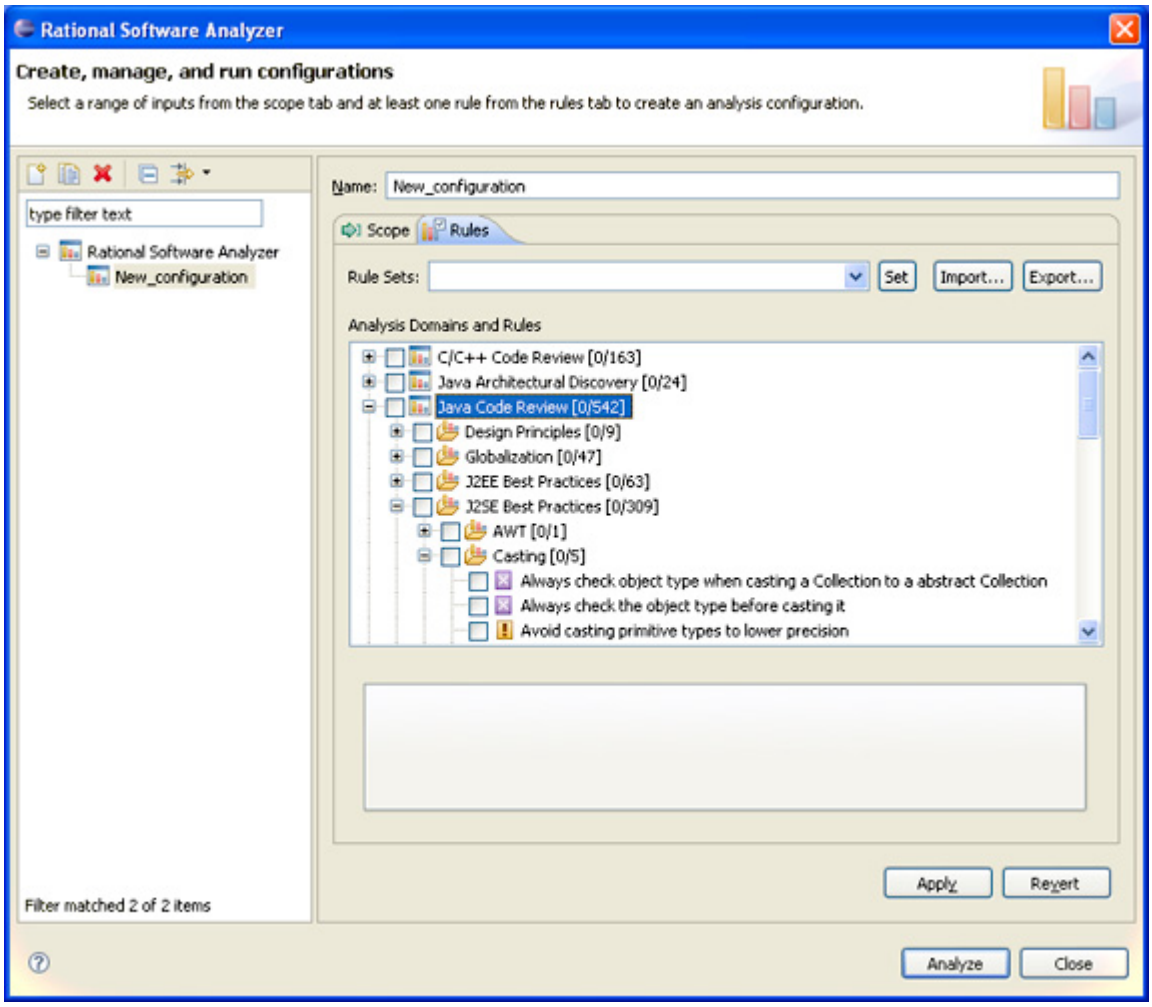
在樹狀結構中，每個節點前方的勾選框是用來控制元素的狀態啓用，元素選取或取消選取時，所有子節點將設定爲相同狀態，如此便可快速選取整個種類，甚至整個樹狀結構。本練習：

2. 選取整個 **Code Review for Java** 分支。

附註：

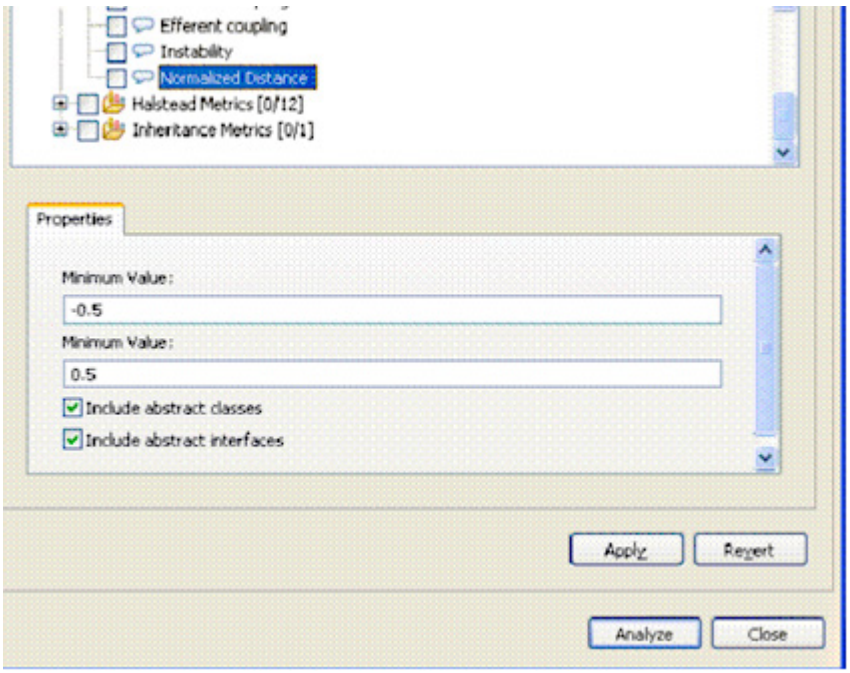
若您在分析配置對話框中看到的規則數目與圖 5 所示不同，不用擔心，許多 Rational 產品都提供 RCS 功能，其中所納入的規則集各有不同。

圖 5：分析配置對話框



部分規則會顯示額外的配置選項，在此情況下，Rules 標籤的下半部會顯示規則的現行設定，否則即爲空白。圖 6 顯示某個 Java 軟體衡量指標的範例規則參數清單，供您參考。規則以此方式顯示參數時，您可以視需要加以調整，新的值將會隨規則選項自動儲存。

圖 6：範例規則參數清單



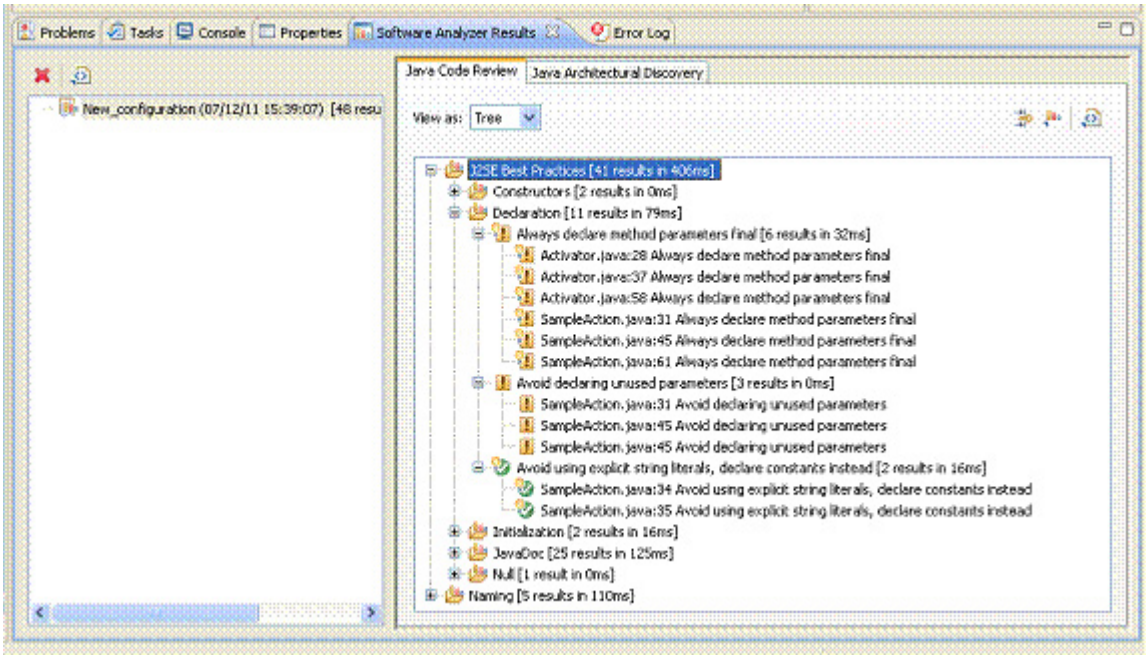
執行分析並顯示結果

1. 要開始分析，請按一下 **Analyze** 按鈕。

便會在 Eclipse 工作台中看到 Analysis Results 視圖。取決於您執行的分析種類，結果視圖將有所不同。有些結果視圖支援以多種格式檢視結果（例如表格或樹狀結構），像是 Java 程式碼複查所提供的視圖。

如圖 7 所示，若您的分析配置包含多種分析類型選項（在此例中爲程式碼複查與架構探索），結果視圖將會爲每個分析提供者的結果，提供專用標籤。

圖 7：Java Code Review 結果視圖

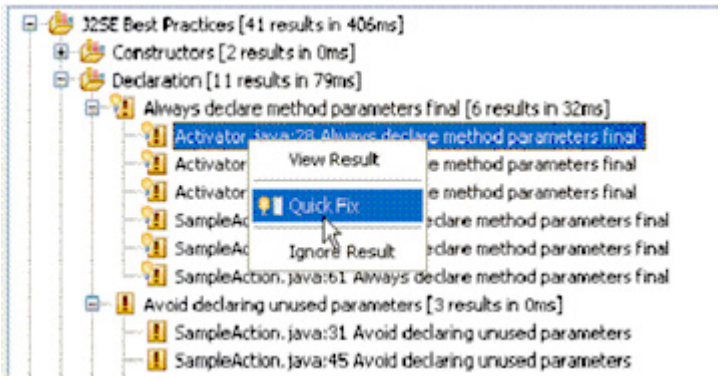


提示：

在結果上按一下滑鼠右鍵，即可執行特殊作業，例如檢視其中發生問題的原始碼，或使用自動結果「快速修正」問題；若規則作者爲規則提供快速修正常式，快速修正便輕而易舉。

2. 若已啓用 **Quick Fix** 功能表選項，選取後便可帶領您進行問題修正程序。

圖 8：Quick Fix 選項



請注意，用來呈現結果的檢視器，是所包含資料類型的功能。檢視結果時，編輯器可能會開啓原始檔，其中包含強調文字、UML 圖或統計資料表。檢視結果的方式不一，完全由規則作者決定。對 Java 程式碼複查分析提供者而言，所有結果都以可編輯的 Java 原始檔進行檢視。

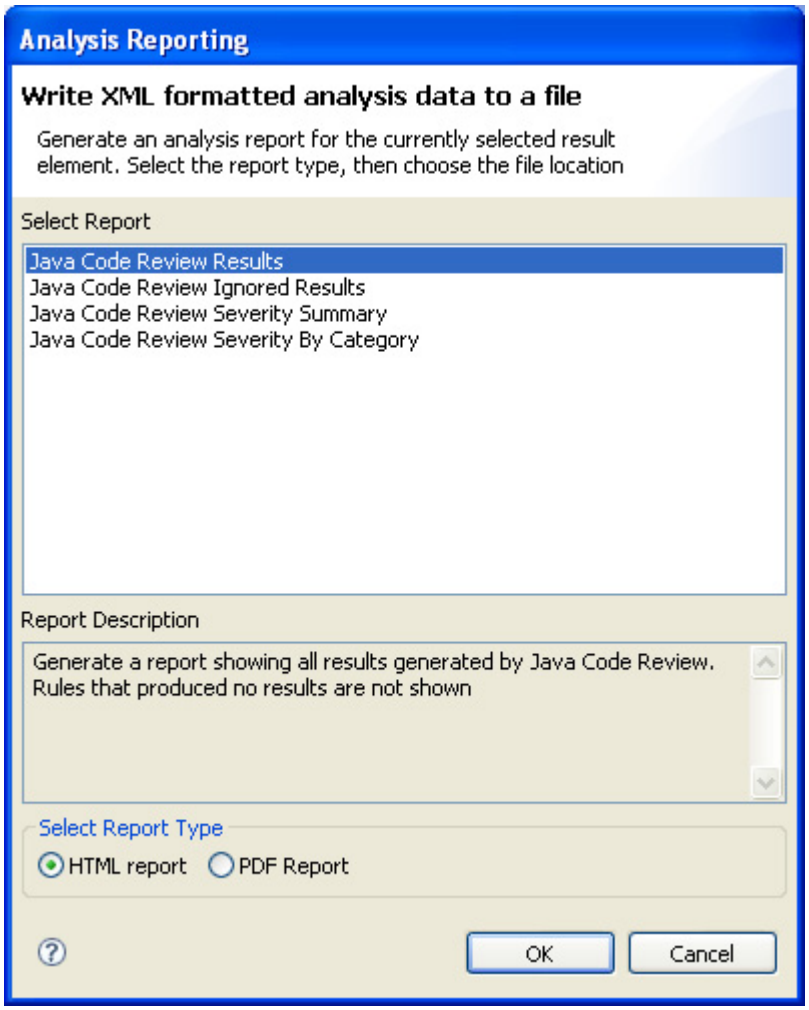
匯出與匯入

在結果視圖中，可能有其他一般功能可供使用（取決於執行的分析類型），這些功能採用資料匯

出與產生報告的形式。

如名稱所示，資料匯出可讓您將原始分析結果匯出成檔案，一般是 XML 檔案格式。資料的匯出類型由分析提供者決定，分析提供者會供應已知資料匯出項目清單，並可選擇執行項目（請參閱圖 9）。

圖 9：Analysis Reporting 視圖



產生報告與匯出資料相當類似，事實上，這兩項功能共用匯出程式。然而，產生報告會產生良好格式化的頁面，可儲存在本端，或直接寫入遠端網站。您可以取用任何現成報告檔，並視需要加以修改（例如新增公司標誌）。所產生的報告類似圖 10 與圖 11 所示，但由於產生報告引擎相當彈性，也可提供其他變化。

圖 10：Java 程式碼複查

Java Code Review

May 14, 2007 2:04 PM

Design Principles

Avoid methods with more than 3 parameters

/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/AnalysisLaunchConfigurationDelegate.java:62
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui//AnalysisLaunchConfigurationDelegate.java:206
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/IAnalysisResultView.java:26
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/ResultTab.java:50
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/ResultViewDefault.java:108

Avoid using the negation operator "!" more than 3 times

/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/ResultsFrameView.java:314

Globalization

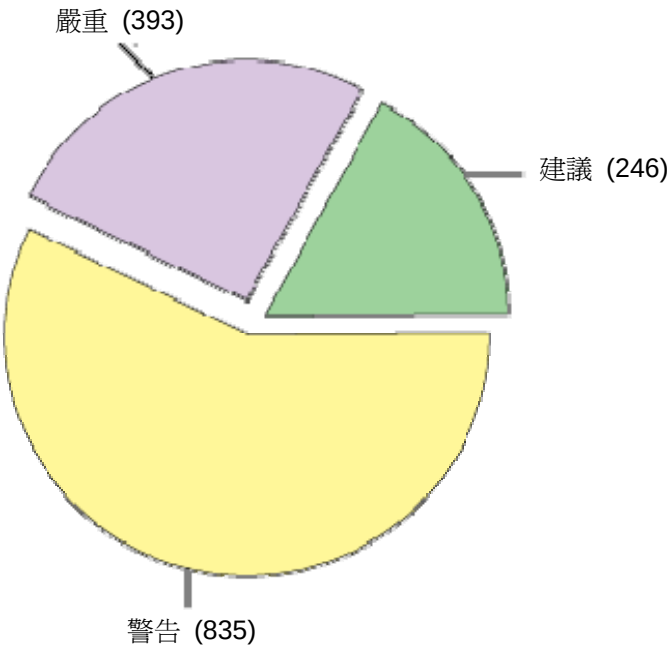
Avoid using java.lang.String + operator

/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/UIMessages.java:20
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/actions/AbstractResultAction.java:100
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/actions/AbstractResultAction.java:125
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/dialogs/ExportDialog.java:204

Avoid using java.lang.String.equals() for multilingual strings

/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/actions/AbstractResultAction.java:87
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/actions/AbstractResultAction.java:112
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/ResultViewDefault.java:220
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/ResultViewDefault.java:346
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/ResultViewDefault.java:355
/org.eclipse.tptp.platform.analysis.core.ui/src/org/eclipse/tptp/platform/analysis/core/ui/views/ResultViewDefault.java:361

圖 11：以圓餅圖說明 Java 程式碼複查的「嚴重性摘要」

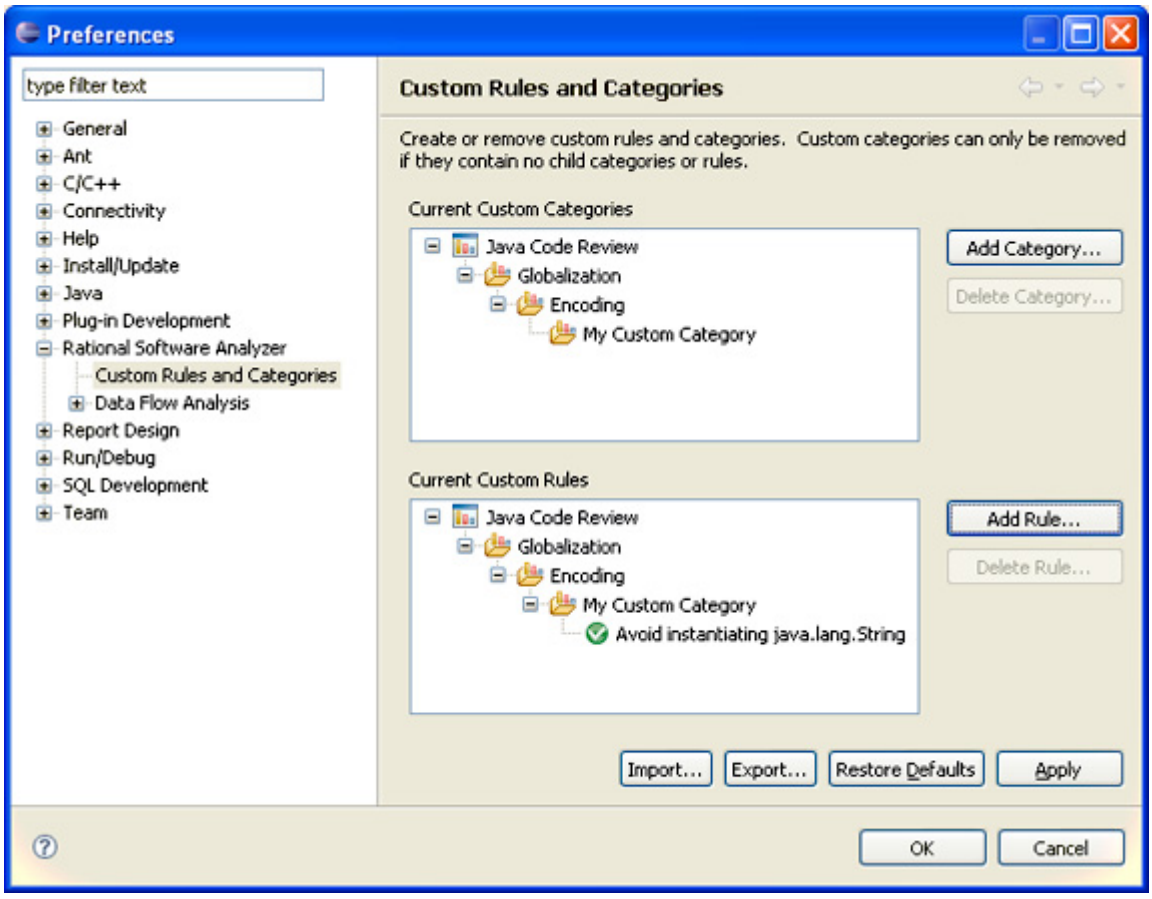


（選用）建立自訂規則與種類

除了 Rational Software Analyzer 所提供的規則，與協力廠商開發人員所提供的規則之外，您還可以從範本建立自訂種類與自訂規則，無需撰寫任何程式碼。若要建立新的自訂規則與種類：

1. 透過選取 **Window > Preferences** 跳至 Preference 頁面，
2. 然後，在 Preferences 樹狀結構中選取 **Analysis > Custom Rules and Categories**（請參閱圖 12）。

圖 12：Preferences 中的 Custom Rules and Categories 視圖



3. 按一下 **Add Category** 以新增自訂種類，

即可進入簡易精靈，您可在此選擇母項種類以及新種類的名稱。自訂種類的樹狀結構控制項，會針對任何自訂種類，顯示完整的種類路徑。只有先前定義的自訂種類，才可以刪除。

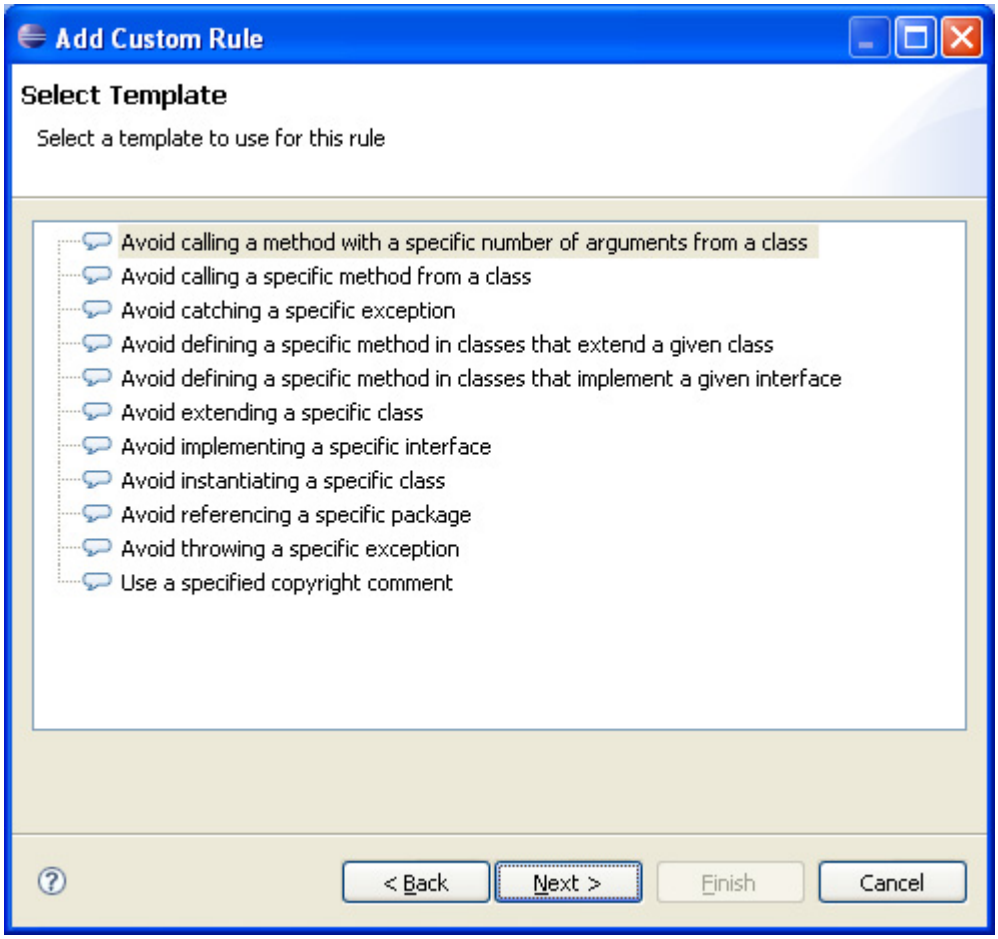
4. 按一下 **Add Rule** 以啓動規則建立精靈，

在第一個精靈畫面中，您可以選擇要在分析種類樹狀結構中的何處放置規則。

5. 在第一個精靈頁面中選取種類，然後按一下 **Next**。
6. 第二個精靈頁面中，您將會看到列示所有規則範本的清單。選取您希望使用的規則範本，作爲新規則的基礎。

請注意，並非所有分析提供者都支援自訂規則，不過 Java Code Review 提供多個自訂規則，您可以自行刪除。

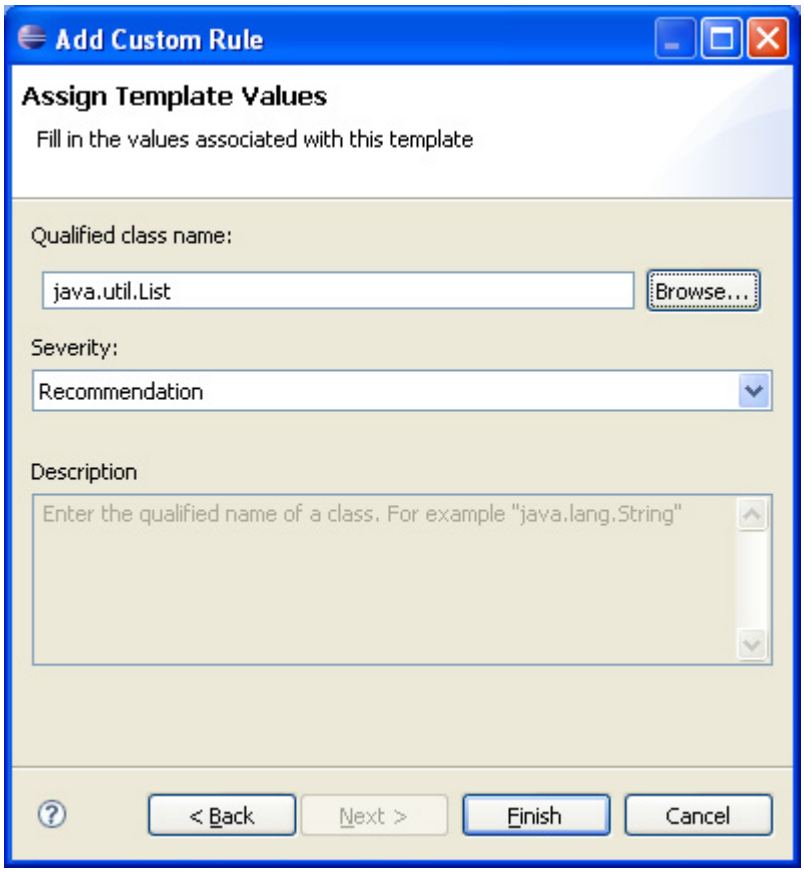
圖 13：可用來建立自訂規則的範本



在最後的精靈頁面上，您將看到規則範本所定義之每個參數的登錄項目。在圖 14 所示的範例中，所選規則範本只定義了一個參數，因此您在所提供的欄位中，只能輸入一個完整類別名稱。

7. 您可以使用 **Browse** 按鈕以瀏覽現有類別，或手動在文字框輸入有效的類別名稱。

圖 14：Assign Template Values（參數）視圖



8. 按一下 **Finish** 按鈕建立以範本爲基礎的規則，並將其新增至規則樹狀結構。

此後您即可選取此規則，作爲任何分析配置之部分。

後續內容

這是四篇系列文章中的第一篇，以籠統的方式介紹靜態分析，說明 Rational Software Analyzer 的重要功能；Rational Software Analyzer 可協助您提早找出開發週期中的程式碼品質問題。第二部分中，您可以深入瞭解 Rational Software Analyzer 中的 Java 程式碼複查功能，包括深入研究基本規則編寫、使用所提供的 API，並利用更多進階功能，建立規則範本或內含變數資料的規則。

資源

學習

- 造訪 Rational Software Analyzer area on developerWorks 以獲得深度資訊介紹。
- 本系列文章的第二部分，將介紹如何建立規則與規則過濾

器，以延伸 Java 程式碼複查。

- 本系列文章的第三部分，針對如何加強 Java 程式碼複查的規則，提供建議。
- 本系列文章的第四部分，則探討整合自身分析工具的相關技術。
- 造訪 Rational software area on developerWorks，以取得 Rational Software Delivery Platform 產品的技術資源與最佳實務。
- 瀏覽 Rational 電腦化、Web 型與講師教授型線上課程。透過初級至進階的課程強化您的技能，並進一步瞭解 Rational 工具。您可以透過電腦化訓練或 Web 型訓練，購買本型錄上的各項課程，另外也免費提供部分「入門」課程。
- 訂閱 Rational Edge 新聞信件，以取得有效軟體開發概念的相關文章。
- 訂閱 IBM developerWorks 新聞信件，每週更新最佳的 developerWorks 教學課程、文章、下載項目、社群活動、網路廣播與事件。
- 瀏覽技術書店，以取得以上主題和其他技術主題相關的書籍。

取得產品與技術

- 下載 IBM Rational 軟體的試用版。
- 下載 IBM 產品評估版，並取得 DB2®、Lotus®、Tivoli® 與 WebSphere® 的應用程式開發工具與中介軟體產品。

討論

- 參加 developerWorks Static Analysis 討論區，這是 Rational Software Analyzer 使用者的專屬園地。
- 查看 developerWorks 部落格，並參與 developerWorks 社群。

關於作者

Steve Gutz 是 IBM Rational 程式碼分析工具的開發經理，除了架構與管理 IBM 的商用產品外，Steve 也對 Eclipse 測試與效能工具專案 (TPTP) 貢獻良多，他目前致力於改善基本程式碼複查工具的實作與整合。Steve 在 2002 年加入 IBM Ottawa Lab，他曾於多家政府機關與私人企業擔任資深管理與高階主管職位，其中包括兩家他本人擁有且成功經營的新創公司。他曾撰寫兩本書籍與多篇文章，並經常於會議活動中演講。