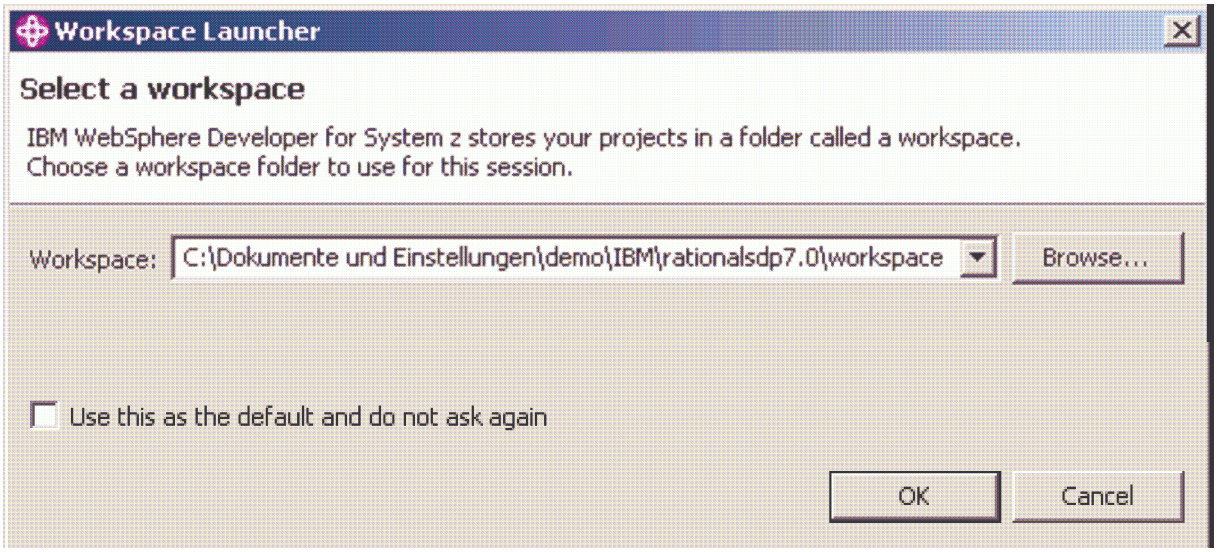
Tutorial 5

Entwicklung eines lokalen COBOL-Programms unter WDz V7.0

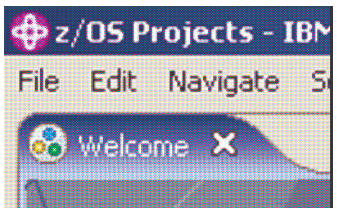
WDz-Nutzung und die z/OS-Perspektive

Start des WebSphere Developers for System z V7 (WDz) über Start → Programs → IBM Software Development Platform → IBM WebSphere Developer for System z → IBM WebSphere Developer for System z.

Ähnlich Eclipse benutzt WDz einen sogenannten Workspace für die Datenspeicherung. Das ist der Ort auf der Workstation, wo alle Programme und Projekte gespeichert werden. Akzeptieren Sie die Default Location, die WDz anbietet, und klicken Sie auf „OK“.



Klicken Sie auf „**x**“, um die Welcome View zu schließen.

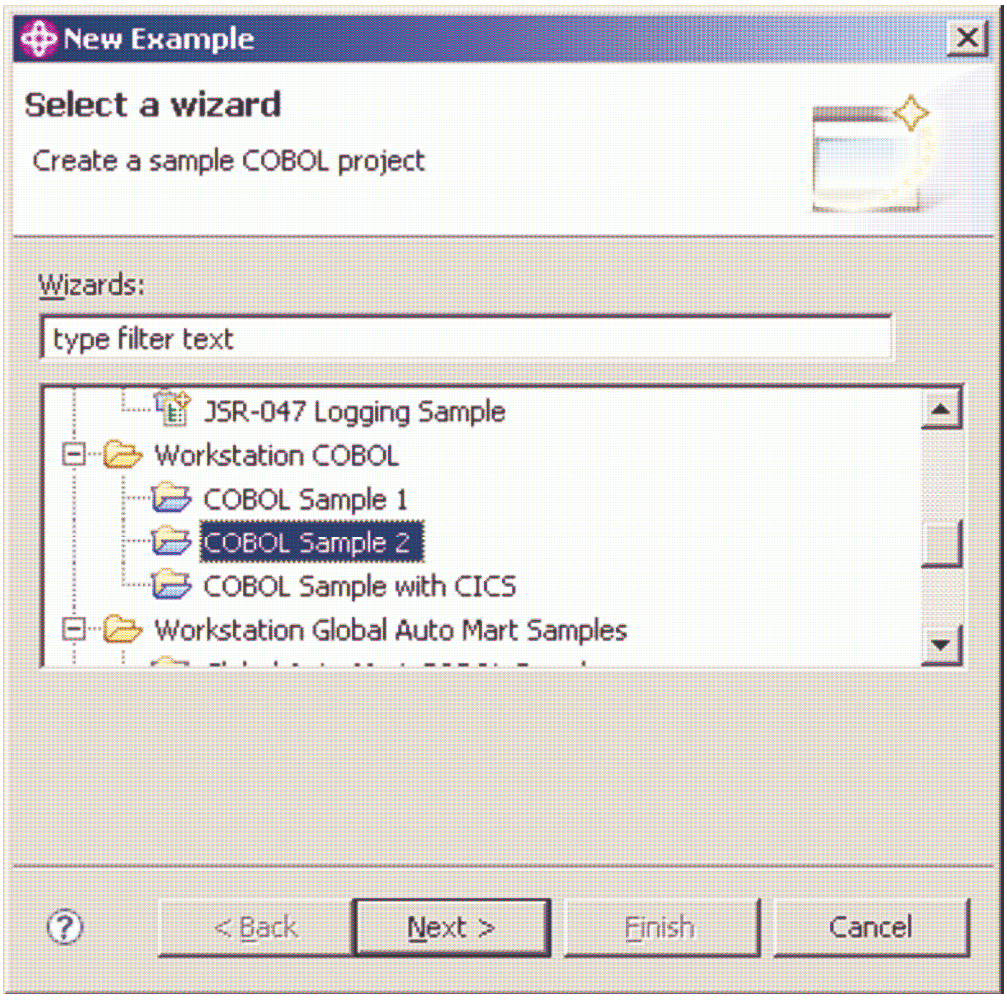


Die Default-Perspektive nach der WDz-Installation ist die Perspektive „z/OS-Projekte“. Letztere werden Sie benutzen, um Projekte und Projekt-Files auf Ihrer Workstation, auf den z/OS-Systemen oder in einer verteilten Umgebung zu erzeugen, zu editieren und zu bearbeiten. Sie erinnern Sich, dass eine Perspektive viele Views hat. Um eine View zu schließen, klicken Sie auf den Close Button, der mit einem X in der rechten Ecke der View markiert ist. Um es zu öffnen, Klicken Sie auf Window → Show View → Other ... und selektieren die View, die Sie zu öffnen wünschen.

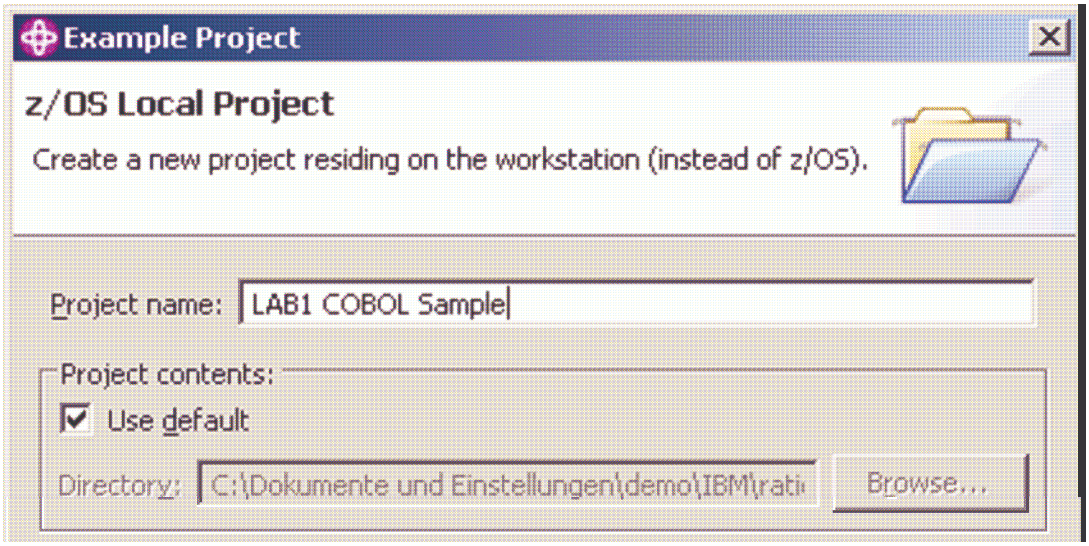
Laden eines lokalen COBOL-Programmbeispiels

Sie werden ein COBOL-Programmbeispiel laden, um damit lokal zu arbeiten.

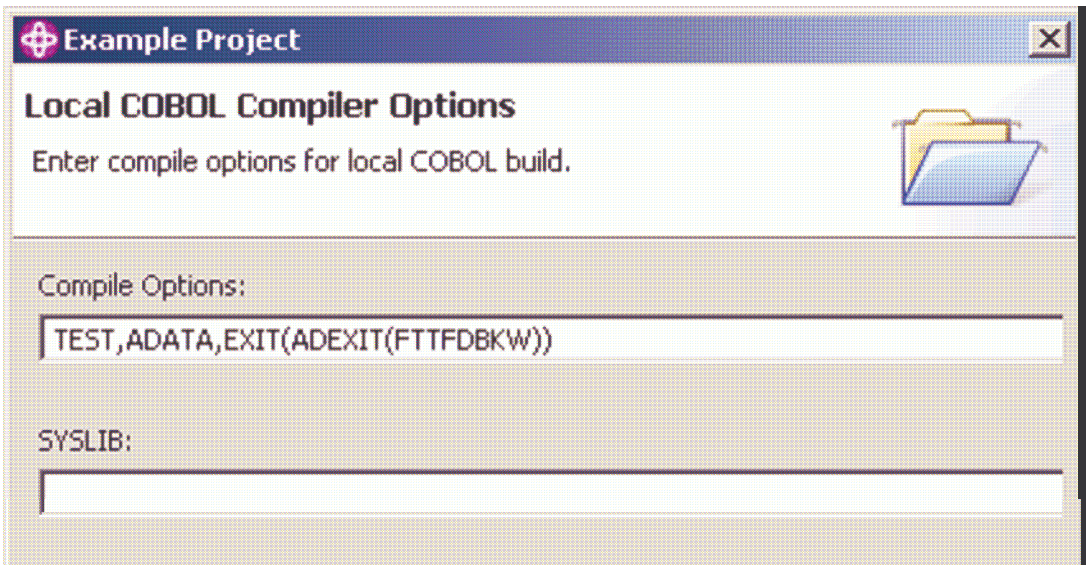
Von der Perspektive „z/OS-Projects“ selektieren Sie File → New → Example...
Expandieren Sie „Workstation COBOL“ (Klick auf +) und selektieren Sie COBOL Sample 2. Anschließend auf „Next“ klicken.



In dem sich öffnenden Fenster „Example Project“ als Project name „LAB1 COBOL Sample“ eingeben und Klick auf „Next“.

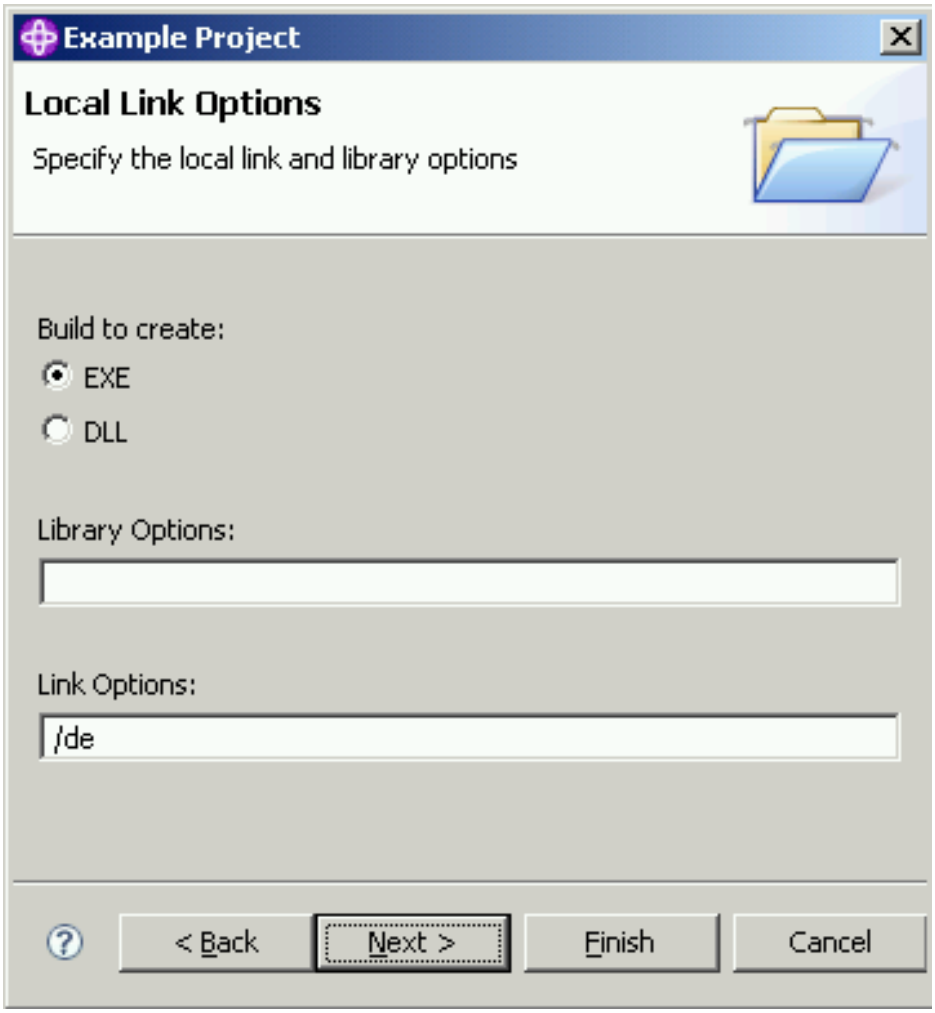


Auf dem anschließenden Panel muss unter „Compile Options“ TEST spezifiziert sein. Der folgende Screen erscheint:

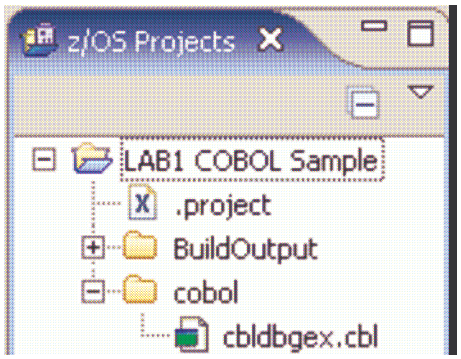


Diese Standard-Compiler-Optionen werden bei COBOL-Programmen benutzt. Die Compiler-Option „TEST“ liefert Sub-Optionen, um Debugging-Funktionen zu definieren. Einige Sub-Optionen werden nur mit spezifischen COBOL-Versionen verwendet.
Klick auf „Next“.

Auf dem folgenden Link-Panel muss die Option für Local Link „EXE“ markiert werden. Das bedeutet, wenn ein COBOL-Programm von diesem Projekt erstellt wird, eine *.exe erzeugt wird. Die Option „/de“ im Feld „Link Options“ muss auch spezifiziert werden. Letztere legt fest, dass das Programm Debugging-Informationen enthält.
Klicken Sie auf „Finish“.



Das Ergebnis Ihrer Arbeit wird aussehen wie:

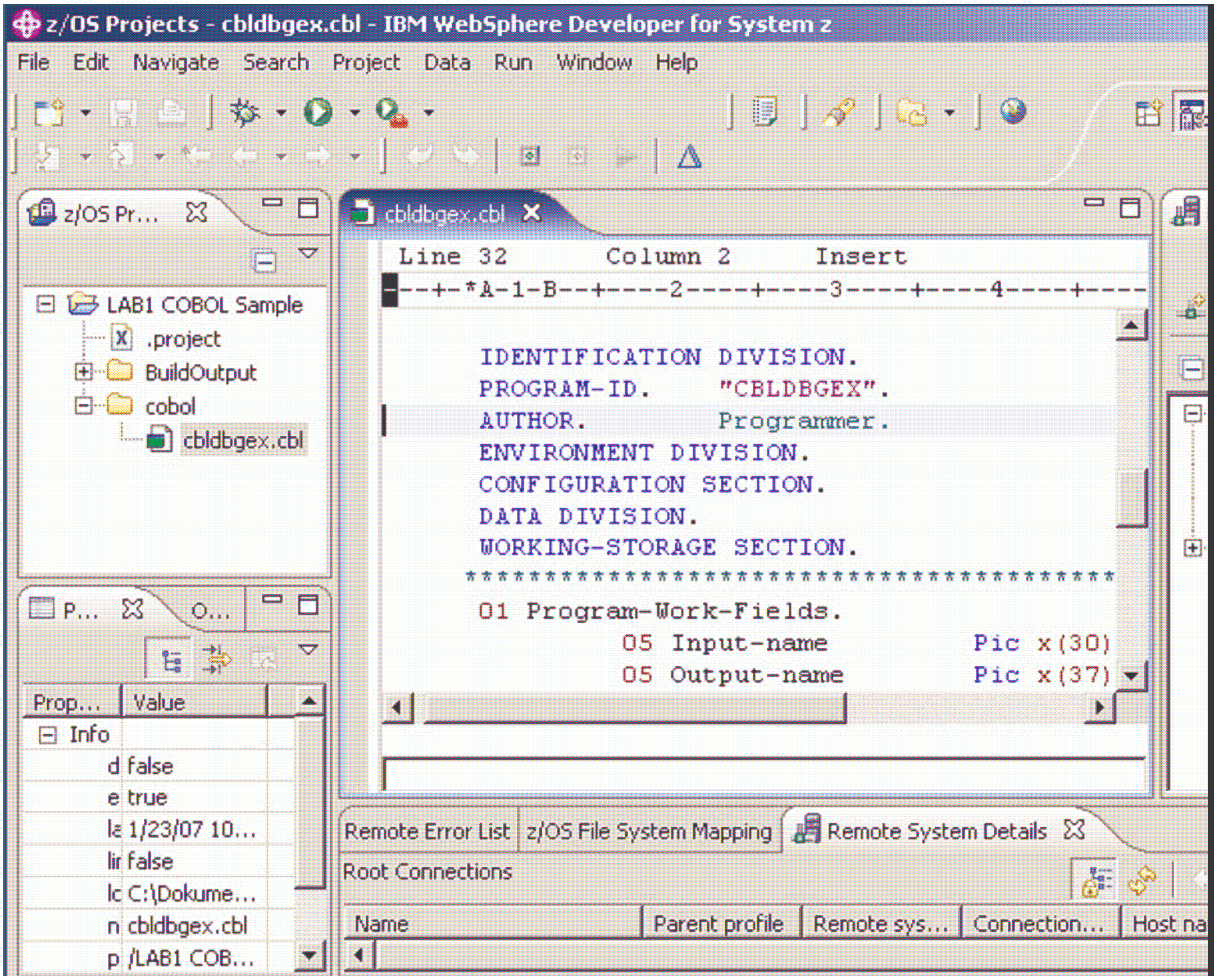


Damit haben Sie das COBOL-Programmbeispiel geladen.

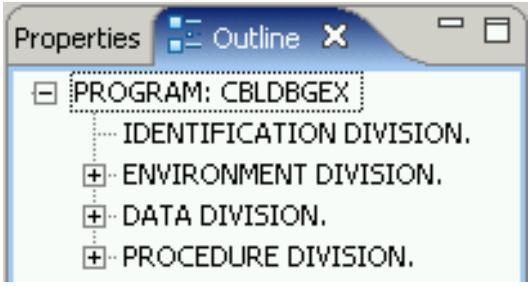
Arbeiten mit einem lokalen COBOL-Programm

Editieren eines COBOL Source Files

Indem Sie die z/OS Projects-Perspektive benutzen, expandieren Sie das Projekt „LAB1 COBOL Sample“ und den cobol-Ordner. Doppel-Klick auf „cbldbgex.cbl“, danach sollten Sie folgenden Screen sehen:



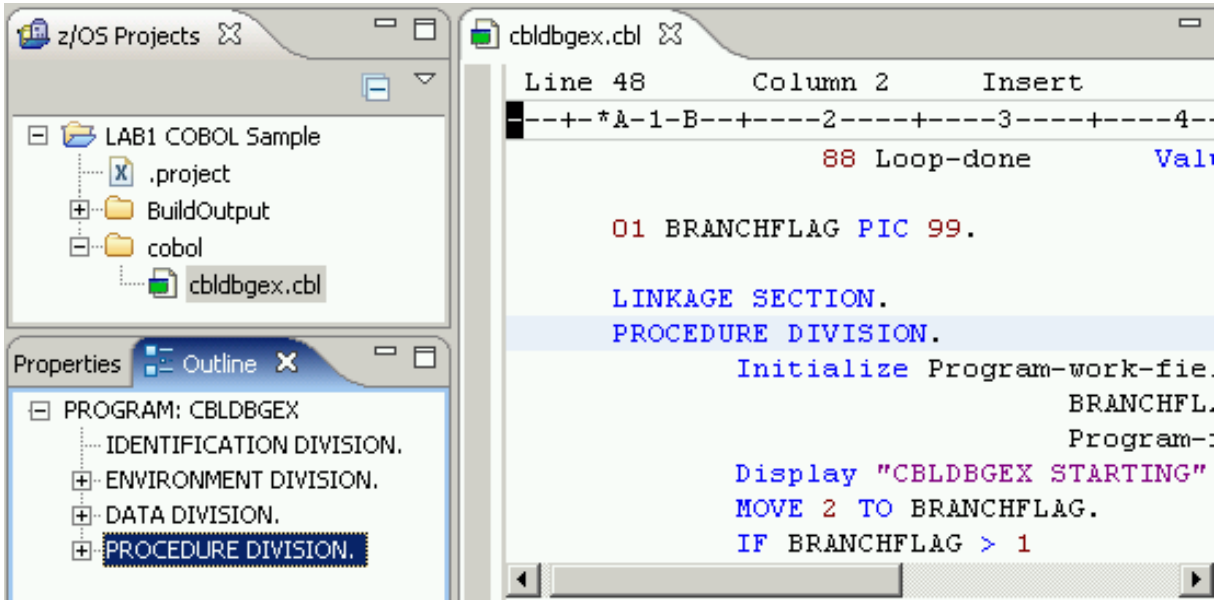
Unten links klicken Sie auf „Properties Outline“, um zur Outline View zu gelangen:



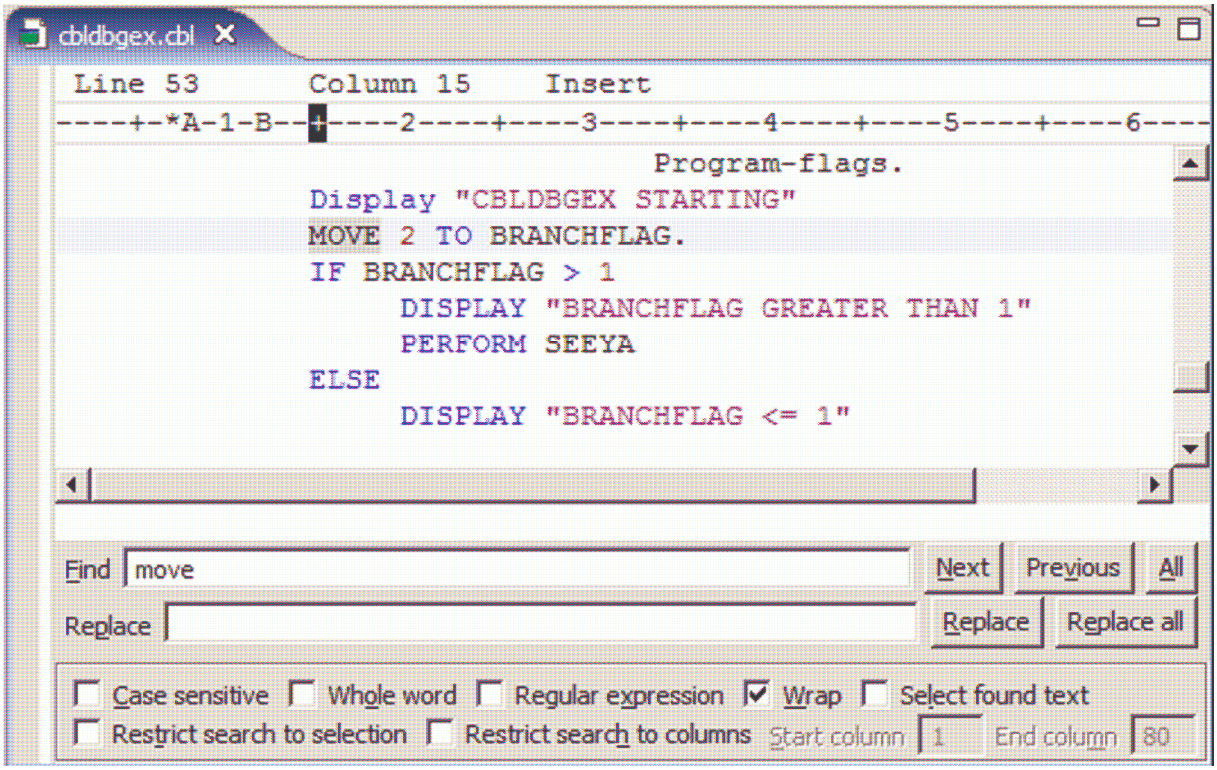
Die Outline View zeigt alle strukturierten Files an, die momentan in dem Editor-Bereich geöffnet sind. Die Inhalte der Outline View sind Editor-spezifisch.

Expandieren Sie die +-Zeichen und betrachten Sie die COBOL-Struktur.

Klicken Sie auf PROCEDURE DIVISION, als Ergebnis zeigt der Editor folgenden Bereich:

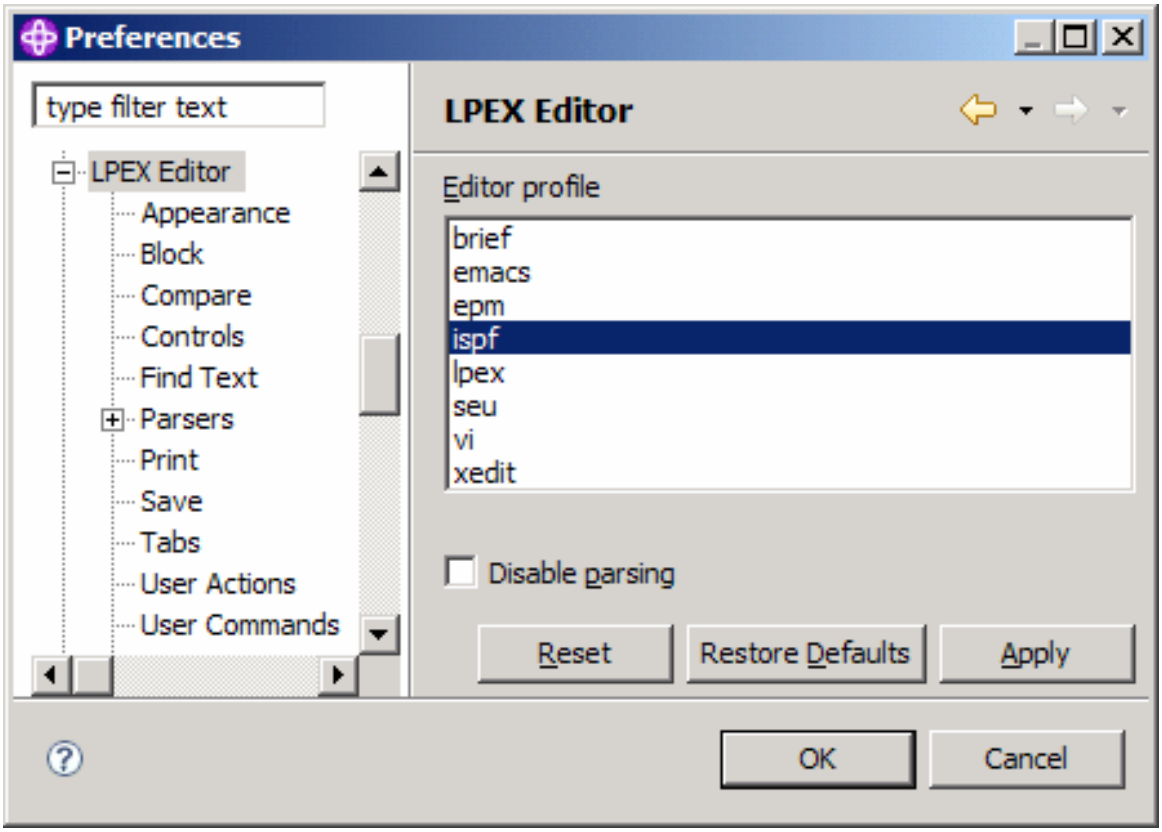


Das Default Profile, das vom LPEX-Editor benutzt wird, ist lpex, der das Key-Verhalten definiert. Der LPEX-Editor kann das Verhalten anderer Editoren wie ISPF und XEDIT emulieren.
Benutzen Sie die Tasten Ctrl + F und geben Sie „move“ ein, um das Wort zu finden. Es ergibt sich folgender Screen:

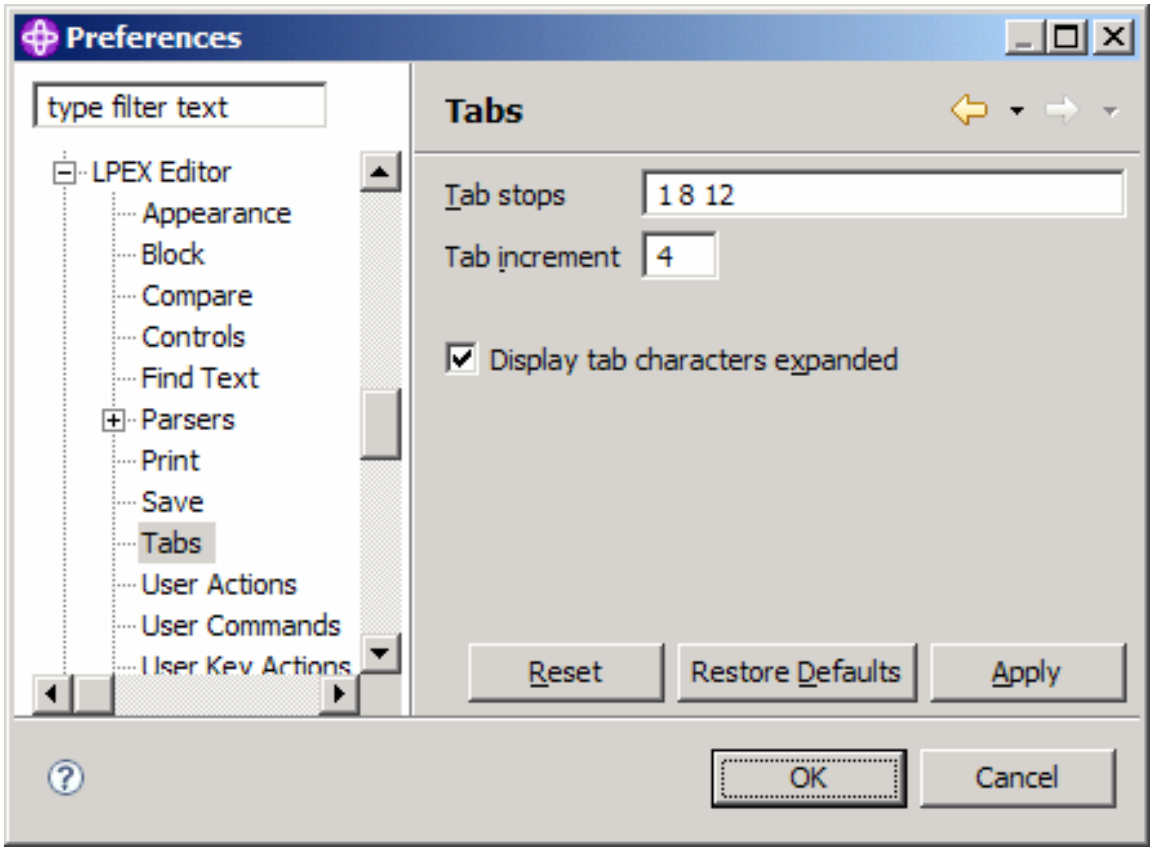


Wenn Sie ISPF benutzen wollen, dann setzen Sie das Editor Profile auf ISPF. Öffnen Sie den Dialog „Preferences“ durch selektieren von Windows → Preferences.

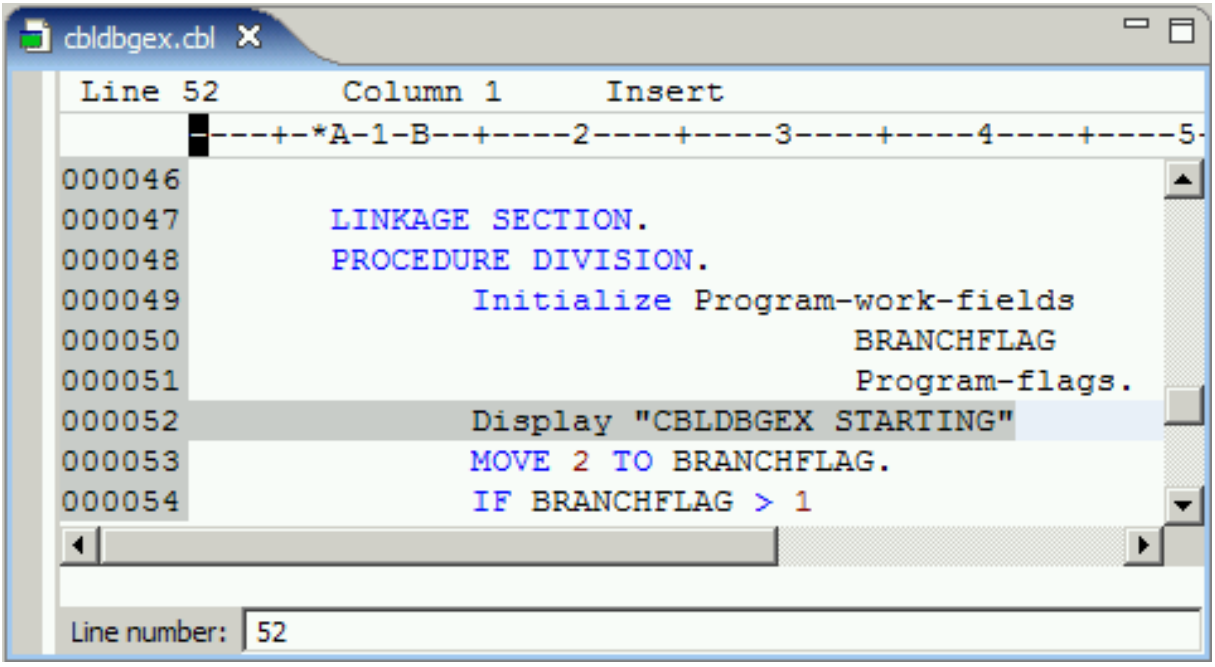
Auf dem linken Panel klicken Sie auf „LPEX Editor“, und von der Editor Profile-Liste klicken Sie auf „ispf“. Schließen Sie das Fenster noch nicht.



Erzeugen von Tabs für den Editor:
Da Sie COBOL benutzen, ist es vorteihaft, einige TABS für die Navigation im Editor zu generieren.
Expandieren Sie den LPEX-Editor, klicken Sie auf „Tabs“ und geben Sie 1 8 12 als Tab Stops ein. Ändern Sie auch das Tabs-Increment auf 4 (anstatt von 8). Jedesmal, wenn die Tab-Taste gedrückt wird, wird der Cursor zu den Positionen 1, 8, 12 bewegt. Das Increment 4 wird Stops bei 14, 16, 20, usw. erzeugen.
Klicken Sie auf „OK“.

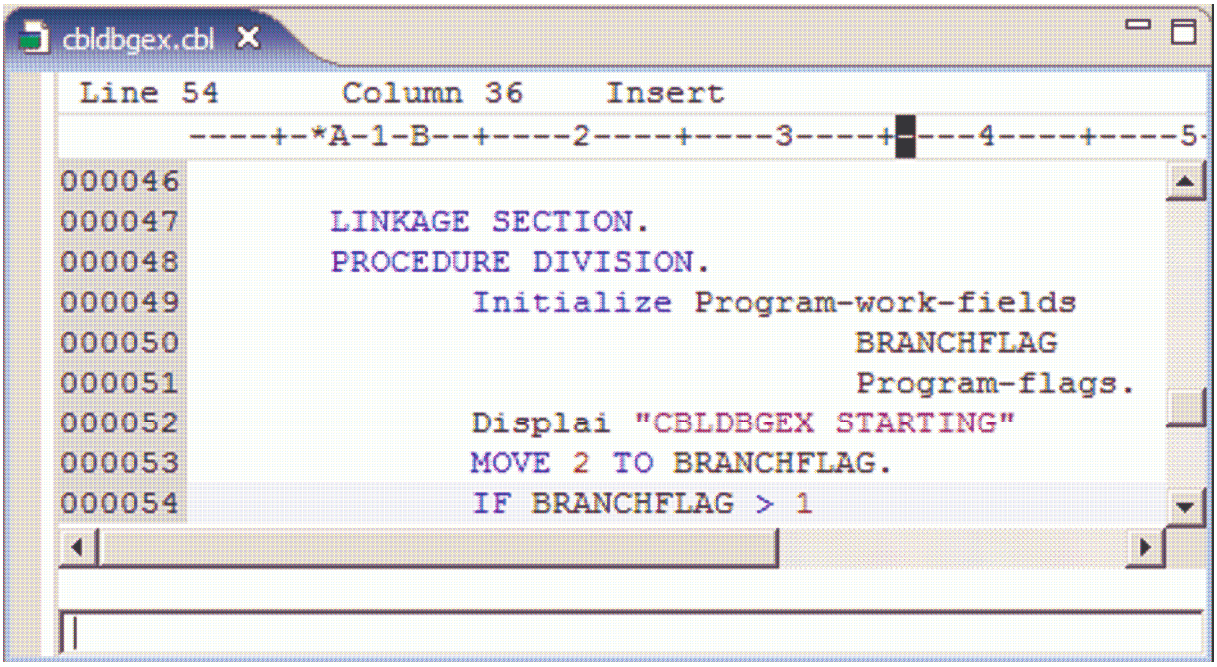


Sie sind nun in der Lage, Prefix-Kommandos im Prefix-Bereich einzugeben, so wie z.B. „d“ für „delete“, „m“ für „move“ usw.
Sie können auch Tastenkombinationen für Kommandos nutzen, d. h. Ctrl + L für „Locate“, Ctrl + F für „Find“ usw.
Um zur Zeile 52 zu gelangen, betätigen Sie Ctrl + L, geben 52 im Feld „Line number“ (unten) ein und betätigen „Enter“. Als Ergebnis erscheint die Zeile 52 als markiert.

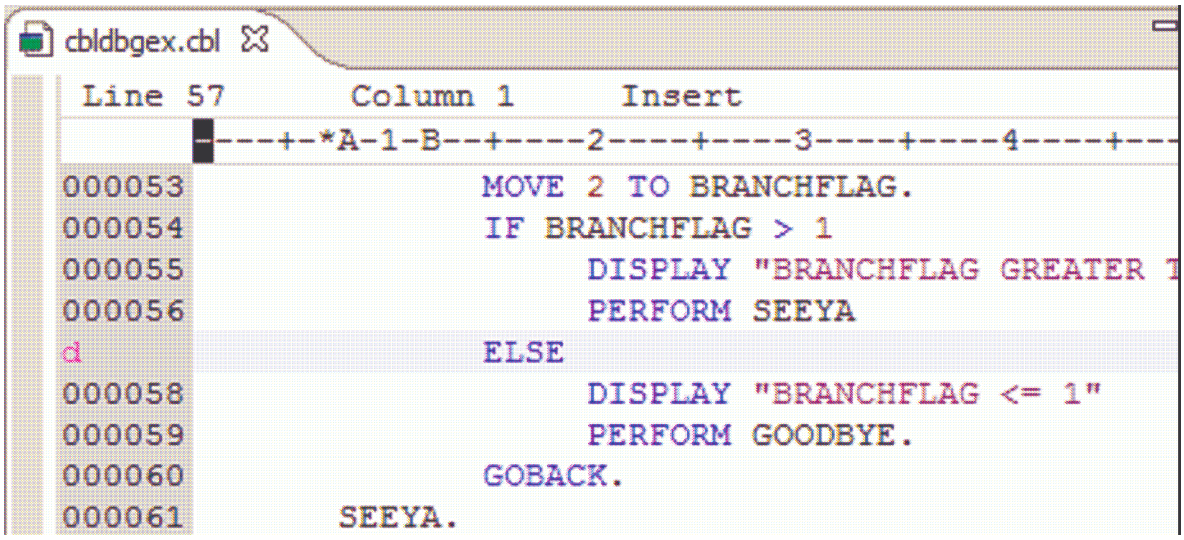


Ändern Sie in der Zeile 52 das Wort „Display“ zu „Displai“ und drücken „Enter“. Da der benutzte Editor intelligent ist, merkt er, dass „Displai“ kein gültiges COBOL-Statement ist. Das ursprünglich blaue Statement ändert seine Farbe in schwarz. Der * auf der linken Seite des File-Namens zeigt an, dass das File geändert wurde. Der * verschwindet wieder, wenn das File gespeichert wird.

Betätigen Sie Ctrl + S, um die Änderung zu speichern. Jedoch wird der Fehler nicht angezeigt, da das Programm übersetzt werden muss, um den Fehler zu finden. Es ergibt sich nachfolgender Screen:

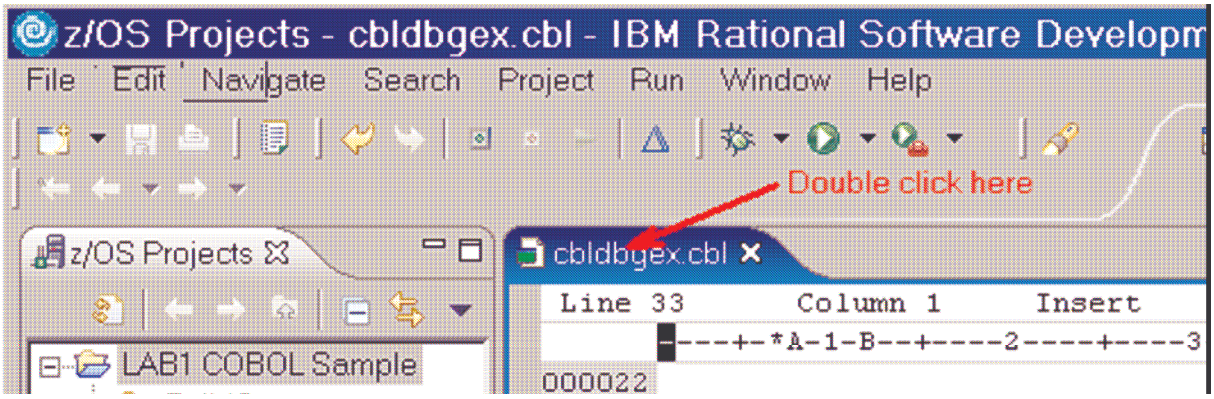


Nehmen Sie eine andere Änderung vor: Gehen Sie auf die Zeile 57, geben Sie ein „d“ im linken Bereich (grau) ein und betätigen „Enter“, um die Zeile zu löschen. Als Ergebnis erscheint der folgende Screen:



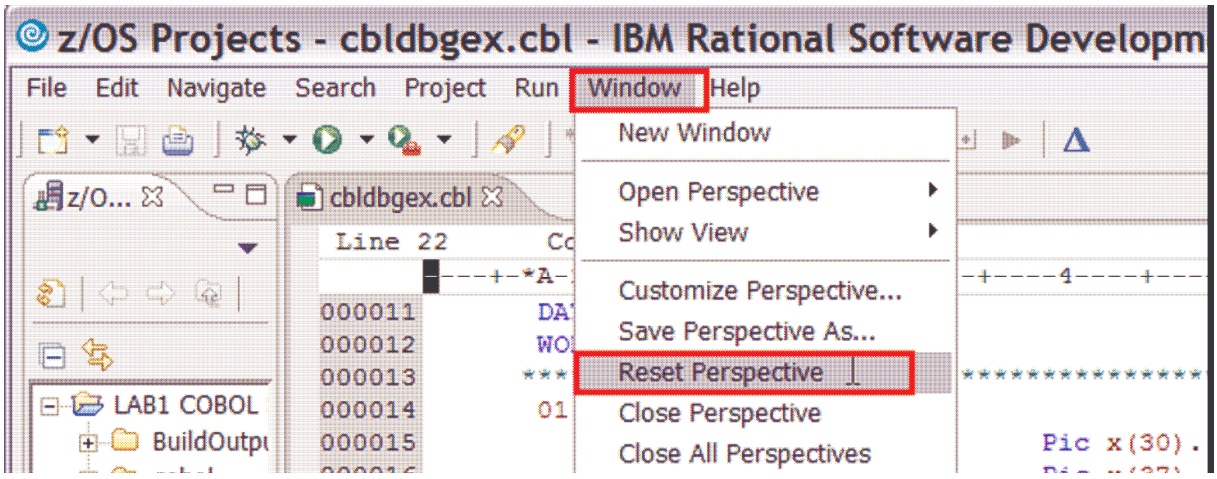
Betätigen Sie Ctrl + S, um die Änderung zu speichern.

Manchmal ist es vorteilhaft, außer dem Editorfenster die Umgebung der WDz-Oberfläche zu sehen. In diesem Fall führt man einen Doppelklick auf dem Namen cbldbgex.cbl aus, dann erhält man den anschließenden Screen:



Nach wiederholtem Doppelklick wird die ursprüngliche Größe wiederhergestellt.

Wenn Änderungen in der Größe, im Editor, in der View vorgenommen werden sollen oder Sie schließen einige Views und wollen danach wieder zu der Perspektive zurückkehren, so selektieren Sie Window → Reset → Perspective wie im nachfolgenden Screen:



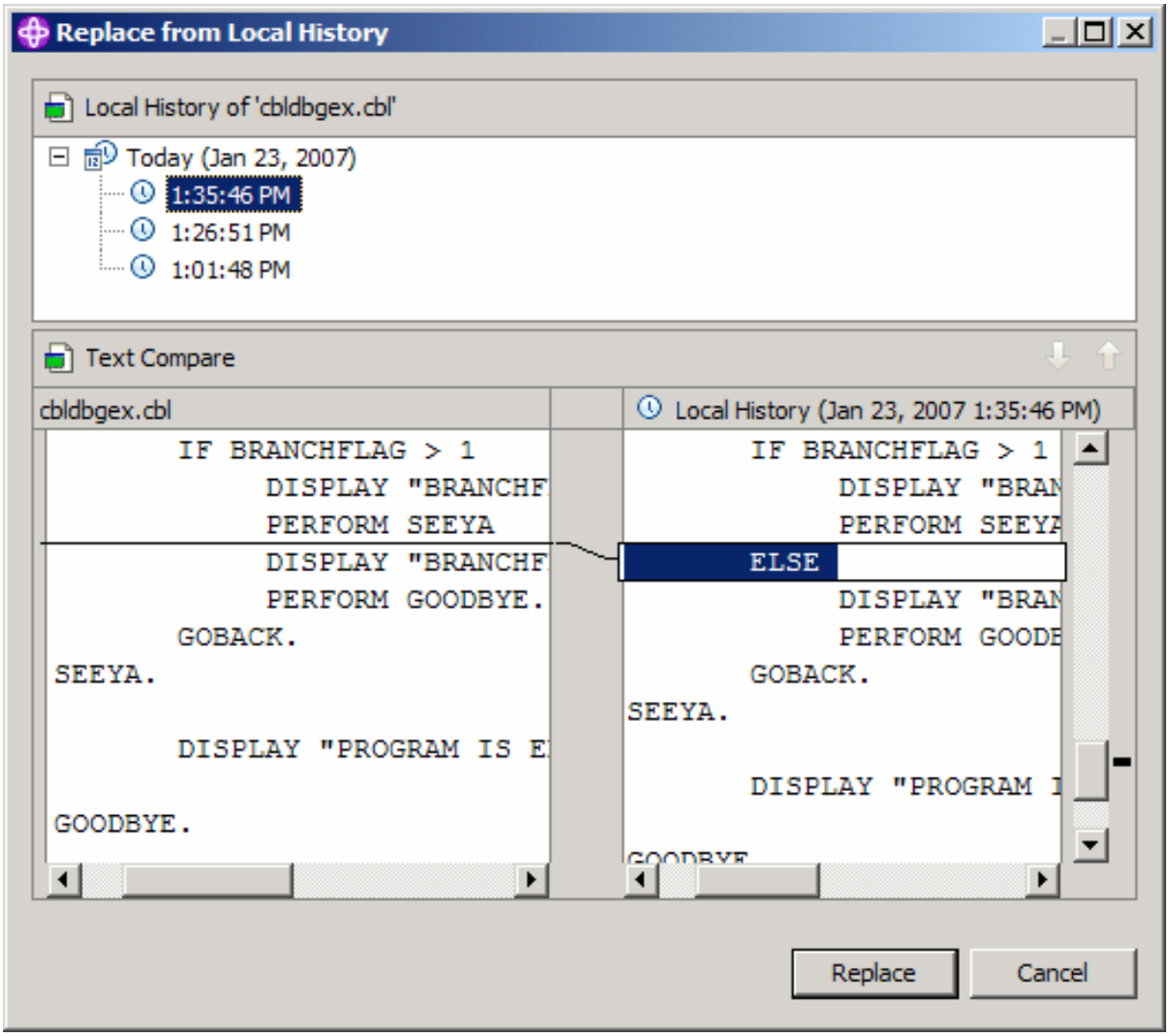
Die Nutzung lokaler File-Versionen

Die lokale History eines Files beim Erzeugen oder Modifizieren eines Files ist im WDz implementiert. Jedesmal, wenn Sie ein File editieren und abspeichern, wird eine Kopie davon gespeichert. Das erlaubt Ihnen zu jeder Zeit, das momentane File mit der vorhergehenden Version zu vergleichen oder das aktuelle File durch das vorhergehende zu ersetzen. Jeder Zustand in der lokalen History wird durch Datum und Zeit des gespeicherten Files identifiziert. Um die augenblickliche Version durch die vorhergehende zu ersetzen, gehen Sie folgendermaßen vor:

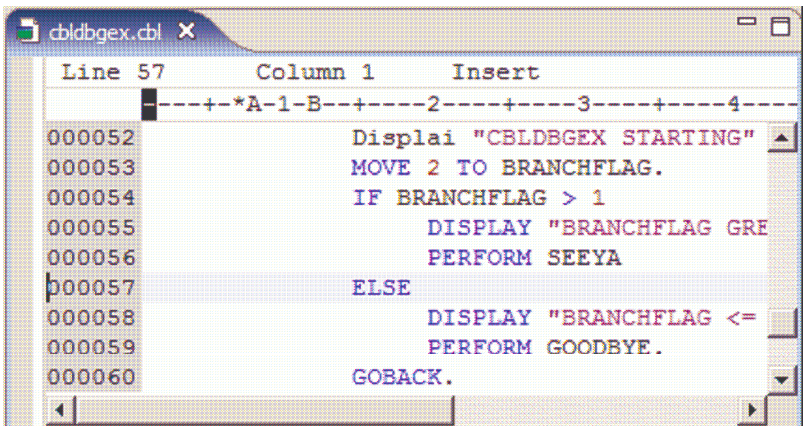
In der z/OS Project View selektieren Sie cbldbgex.cbl und klicken mit der rechten Mousetaste darauf. In dem Kontextmenue selektieren Sie „Replace“ mit → Local History.

Das „Replace“ von der Local History-Seite wird geöffnet. Für jede Save-Aktion (Ctrl + S) in einem früheren Schritt wird ein History Record angezeigt. Klicken Sie auf den letzten Record.

Es erscheint eine kleine Marke rechts. Das ist dann wichtig, wenn Sie mehrere Änderungen vorgenommen haben. Wenn Sie auf eine Marke klicken, bewegen Sie Sich auf diese Änderung. Ein Klick auf „Replace“, um die letzte Version vor dem Ablegen zurückzuspeichern.



Die vorhergehende Version wird zurückgespeichert:

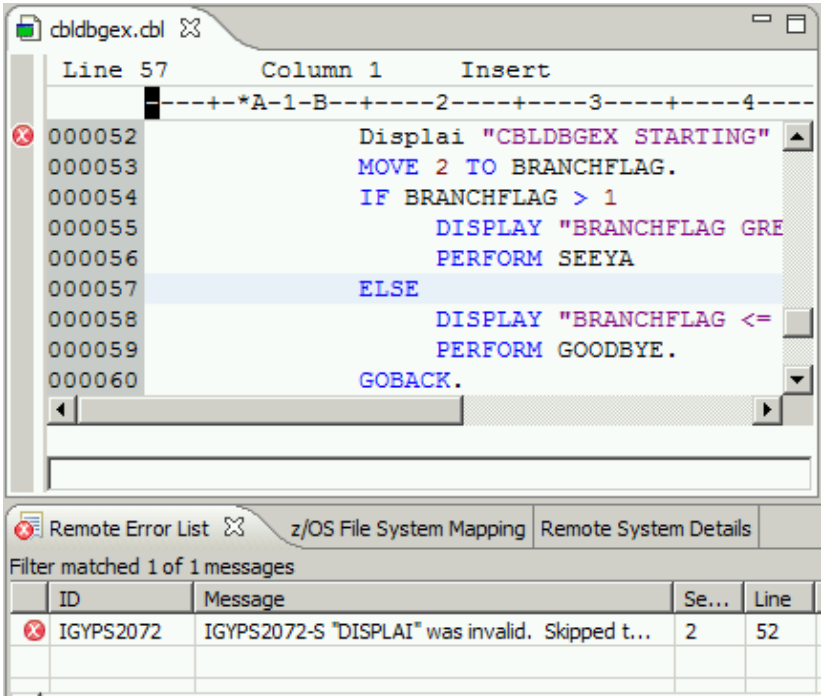


Dabei wurde das Wort „Displai“ nicht auf „Display“ geändert. Diese Änderung wurde vorgenommen durch das Rückspeichern des vorhergehenden History Record.

Schließen Sie den Editor durch Klicken auf „**X**“ neben dem File-Namen cbldbgex.cbl:

Prüfen der COBOL-Quell-File-Synthax

Benutzen Sie die z/OS Project View, rechter Mouseklick auf cbldbgex.cbl und Selektieren von Local Synthax Check des Kontext Menues.
Da es Compiler-Fehler gibt, werden diese in der Remote Error List View angezeigt. Ein Doppelklick auf einen Fehler bringt Sie zu dem Editor, in dem das fehlerhafte Statement angezeigt wird. Der folgende Screen enthält den entsprechenden Sachverhalt:

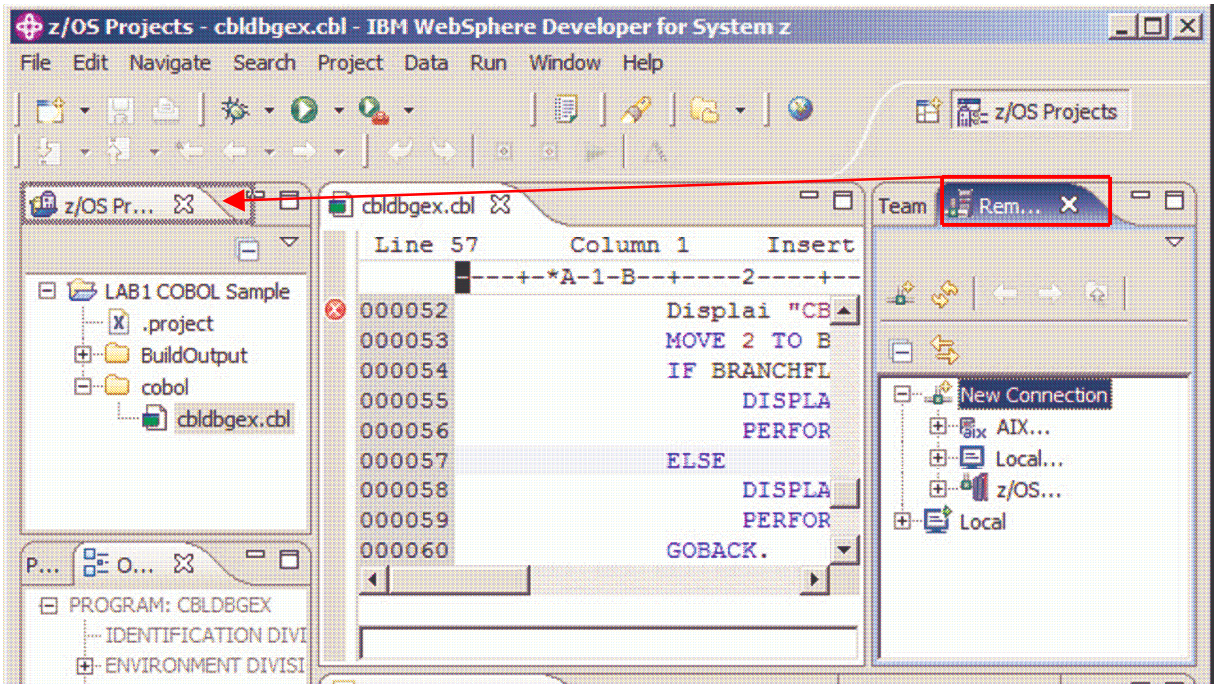


Aufgabe:	*Korrigieren Sie den Fehler in dem Statement der Zeile 52 (Displai). Speichern Sie Ihre Änderung.* *Nehmen Sie einen anderen Local Synthax Check vor, um sicherzustellen, dass die Übersetzung fehlerfrei läuft.* *Es sollte die Remote Error List View nach dem Ausführen des Local Synthax Check keine Fehler enthalten!*

Änderungen im COBOL-Quellprogramm

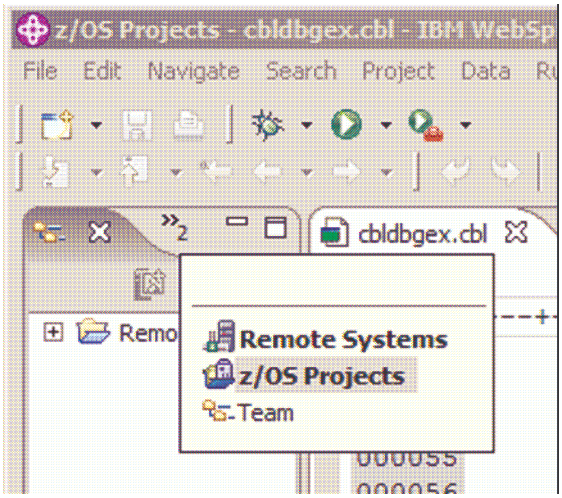
Eine der wichtigsten Funktionen des COBOL-Editors besteht in der Möglichkeit, Fehler im Programmcode vor der Übersetzung zu finden und damit dem Entwickler gezielt zu helfen. Zur besseren Erläuterung wird vorausgesetzt, dass etwas Logik zu dem COBOL-Programm hinzugefügt werden soll.

Als Beispiel werden COBOL-Statements wie DISPLAY, ACCEPT, MOVE und IF hinzugefügt. Hier einige Beispiele:
Da Sie die Views „Remote Systems“ und „Team“ in diesem Tutorial nicht benutzen, können Sie diese auf die linke Seite ziehen, um dadurch mehr Platz für das Editieren des COBOL-Programms zu gewinnen. Es können über Drag und Drop die beiden Views nacheinander zu z/OS Projects bewegt werden:



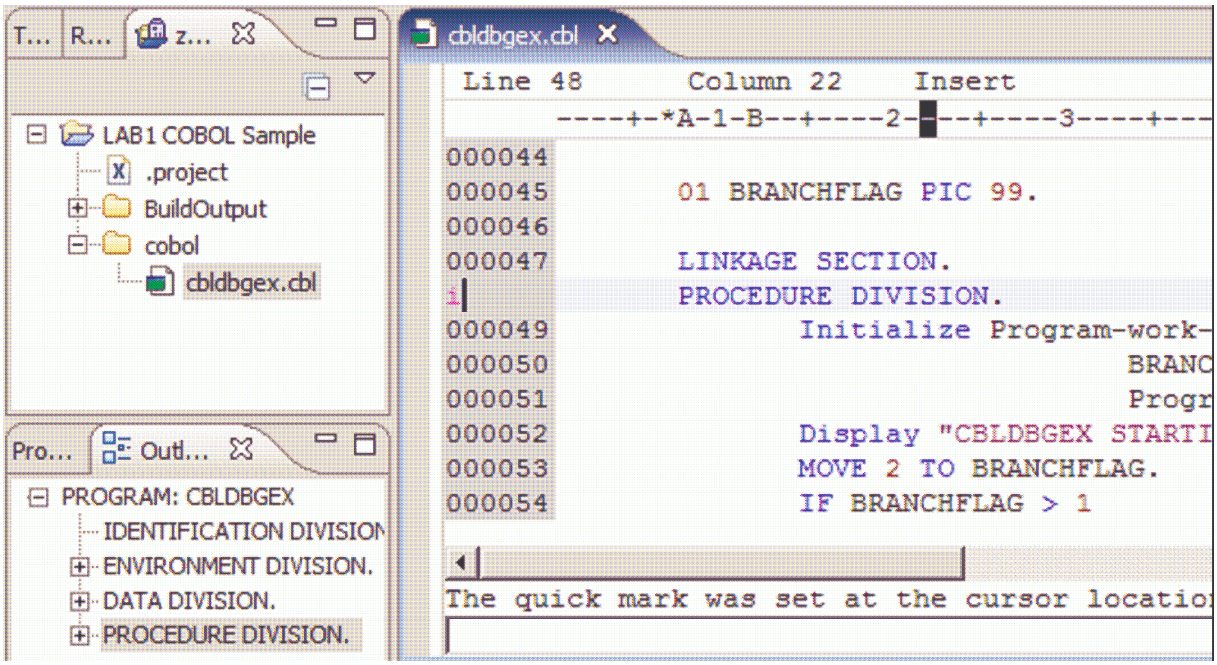
Abhängig von der Windows-Installation können Sie auf das Icon 

klicken und den z/OS Projects-Teil selektieren. Eine andere Möglichkeit bietet ein Klick auf den z/OS Projects-Teil:

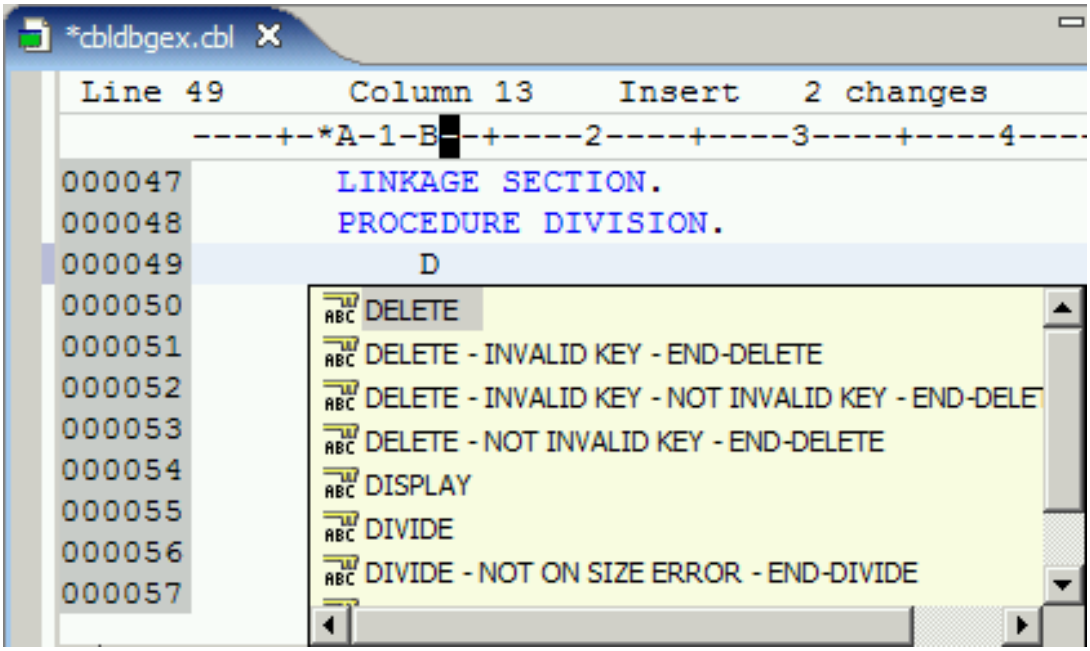


Um COBOL-Statements nach der PROCEDURE DIVISION einzufügen, ist die Outline View zu verwenden. Ein Klick auf die PROCEDURE DIVISION, um den Cursor auf dieser Zeile zu positionieren. Danach wird das Kommando „i“ in dem grauen Bereich dieser Zeile eingegeben. Anschließend wird „Enter“ betätigt, um eine leere Zeile zu

erhalten. Die Nutzung von Ctrl + Enter ist auch möglich, um eine neue Zeile zu öffnen und den Cursor auf diese Zeile zu bewegen.

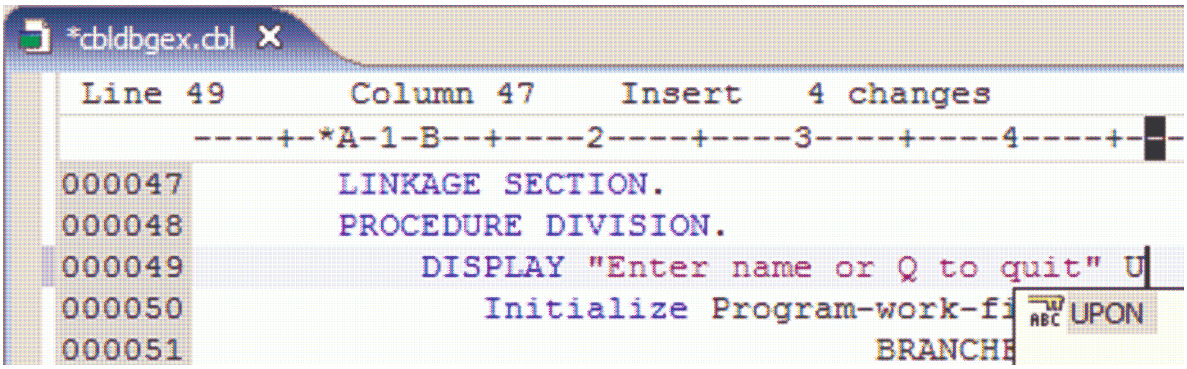


Eine sehr nützliche Funktion zur Code-Generierung stellt der Code-Assistent dar. Bewegen Sie den Cursor zu der Spalte 12 und betätigen Sie Ctrl + Space. Es erscheinen alle möglichen COBOL-Statements. Wenn Sie den Buchstaben „D“ eingeben, werden alle Statements gezeigt, die mit diesem Zeichen beginnen. Nach dem Doppelklick auf das Wort „DISPLAY“ ergibt sich folgender Screen:

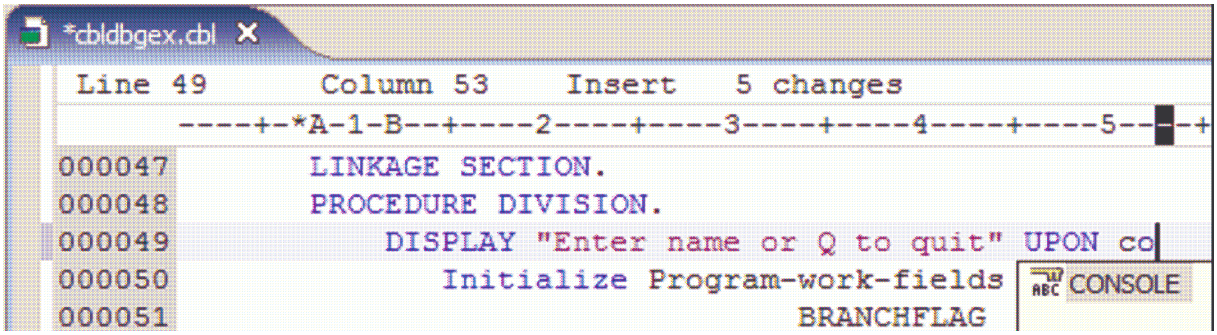


Danach wird der Cursor hinter das DISPLAY-Kommando bewegt und „Enter name or Q to quit“ eingegeben. Anschließend geben Sie ein Leerzeichen ein und drücken

wieder Ctrl + Space. Geben Sie ein „U“ ein und klicken doppelt auf „UPON“ (oder drücken „Enter“). Es erscheint der folgende Screen:

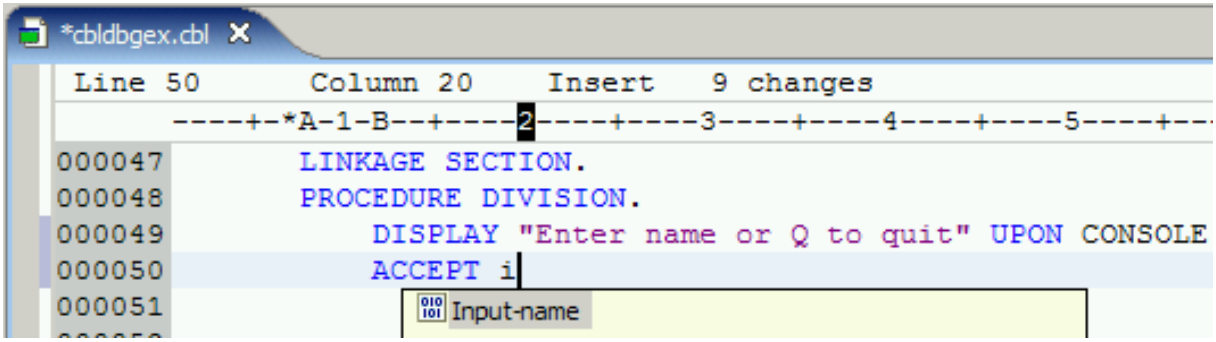


Betätigen Sie wieder Ctrl + Space, geben Sie „co“ ein und führen Sie in der Liste einen Doppelklick auf „CONSOLE“ aus.

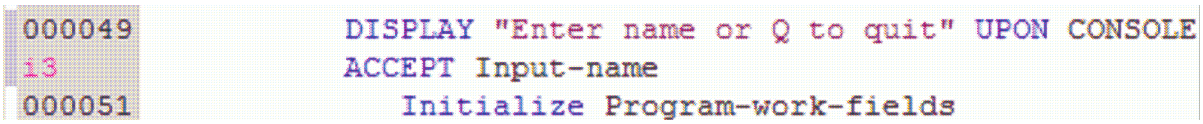


Die Benutzung von Ctrl + Space ist sehr hilfreich, wenn Sie die COBOL-Schlüsselworte benutzen. Wenn z. B. das ACCEPT-Kommando verwendet wird, um die eingegebenen Daten in die Konsole zu den Daten-Namen zu bewegen, kann diese Funktion bei großen Programmen extrem nützlich sein.

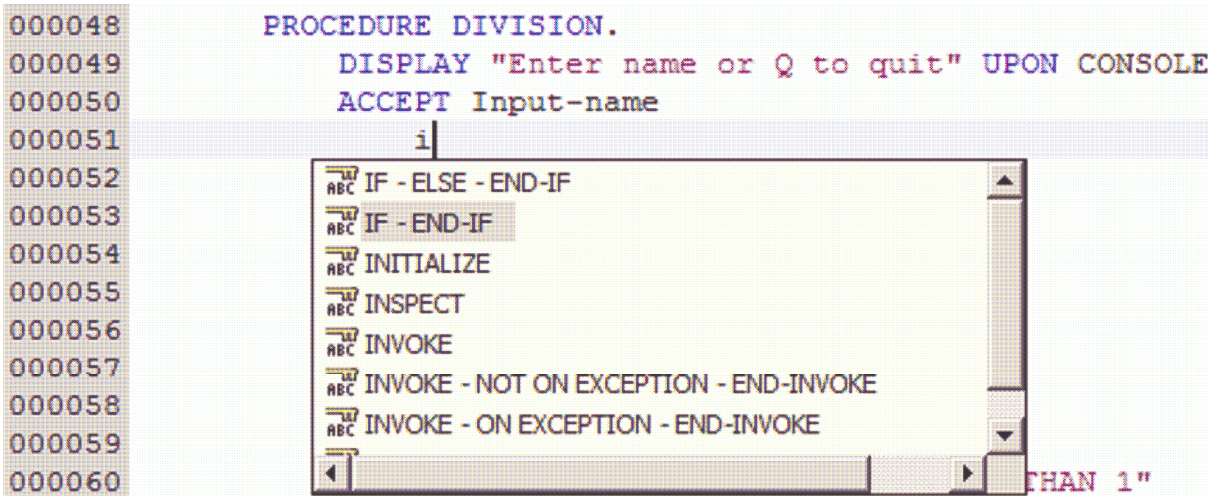
Geben Sie auf der linken Seite des Editors ein „i“ ein und betätigen Sie die Enter-Taste, um eine leere Zeile nach dem DISPLAY-Statement einzufügen. Dieses könnte durch Ctrl + Enter erreicht werden oder durch Drücken von Enter für eine neue Zeile, wenn der LPEX-Editor statt des ISPF-Editors benutzt wird.
Geben Sie das ACCEPT-Statement ein und verwenden Sie Ctrl + Space, um die Variable „Input-name“ einzufügen:



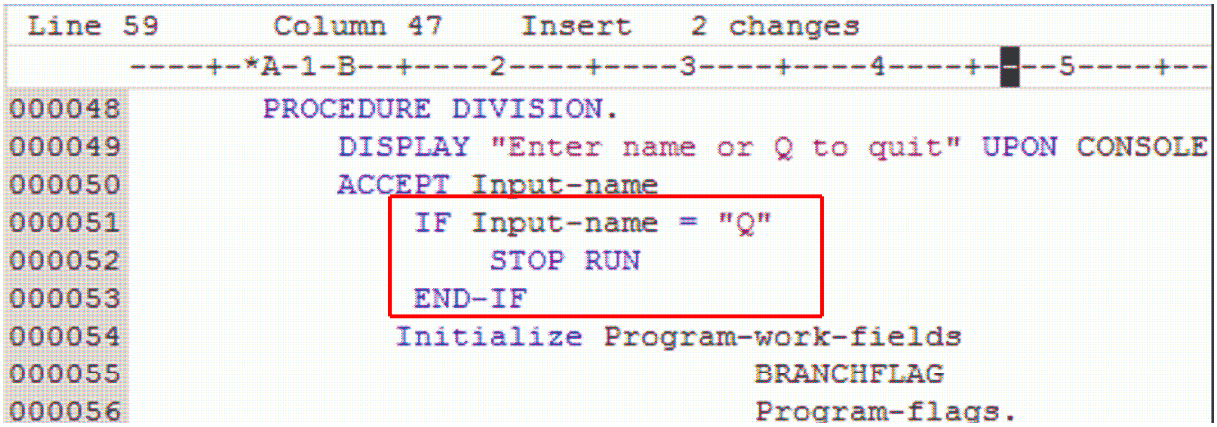
Geben Sie i3 ein und betätigen Sie die Enter-Taste. Als Ergebnis werden 3 leere Zeilen eingefügt:



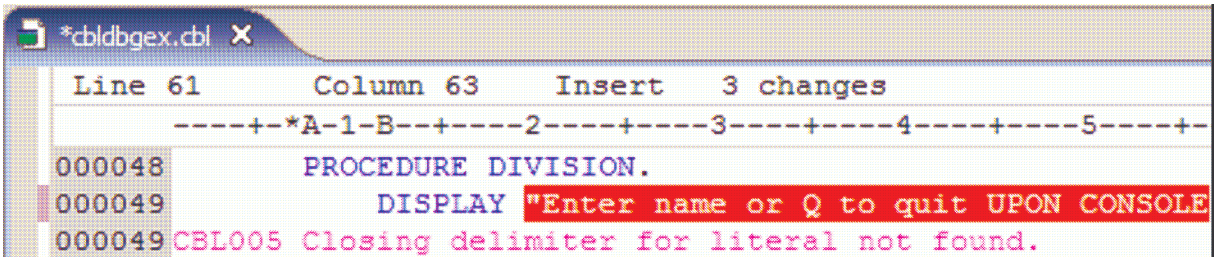
Anschließend wird ein „IF – END-IF“ Statement mittels Eingabe von „i“ und Auswahl des „IF – END-IF“ aus dem betreffenden Dropdown-Menue eingefügt. Es ergibt sich folgender Screen:



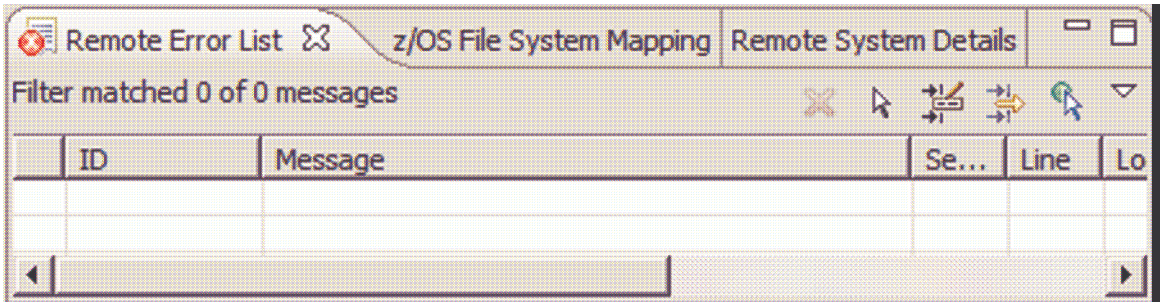
Das Statement wird vollständig erweitert und ist im nachfolgenden Screen dargestellt.



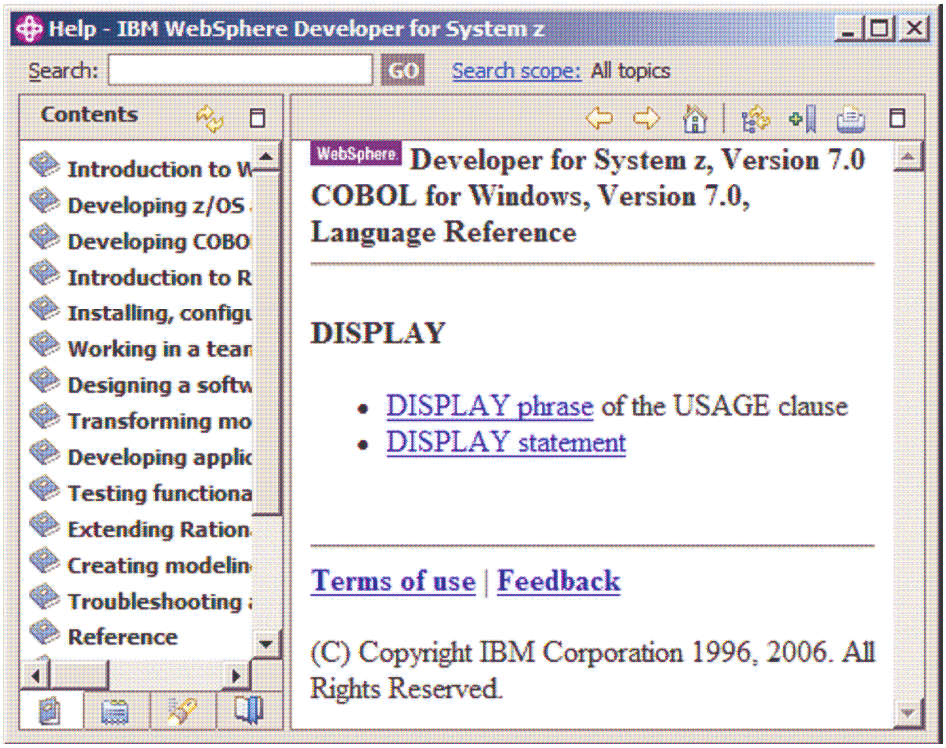
Da der COBOL-Editor einen intelligenten Editor implementiert, können einige Fehler schon zur Codierungszeit gefunden werden. Wenn Sie z.B. die ersten Hochkommas des Statements DISPLAY löschen und Enter drücken oder den Cursor zu einer anderen Zeile bewegen, wird ein Fehler wie folgt angezeigt:



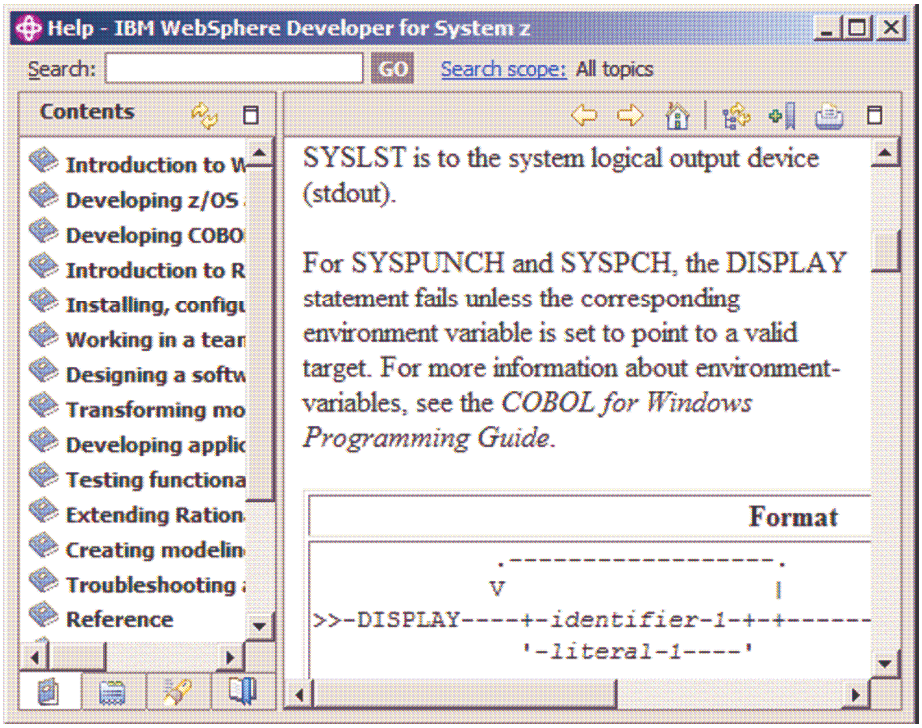
Korrigieren Sie den Fehler und speichern Sie das Ergebnis mit Ctrl + S ab. Der * auf der linken Seite des File-Namens verschwindet, wenn die Korrektur abgespeichert ist! Klicken Sie mit der rechten Mousetaste auf cbldbgex.cbl und selektieren Sie erneut „Local Synthax Check“. Ein Klick auf die View „Remote Error List“ zeigt, dass es keine Fehler in der Liste gibt. Falls doch noch Fehler eingetragen sind, korrigieren Sie diese und führen den Synthax Check wieder durch.



Eine andere nützliche Hilfe beim Codieren eines Programms bildet die Help Built-in-Funktion. Bewegen Sie den Cursor zu dem Statement DISPLAY und drücken Sie F1. Help wird geöffnet und zeigt das selektierte Statement. Dies ist sehr praktisch und kann für irgendein Statement verwendet werden. Nach kurzer Zeit wird das Help-Fenster geöffnet und es wird folgender Screen erscheinen:



Wenn Sie auf „DISPLAY“ klicken, erhalten Sie weitere Informationen:

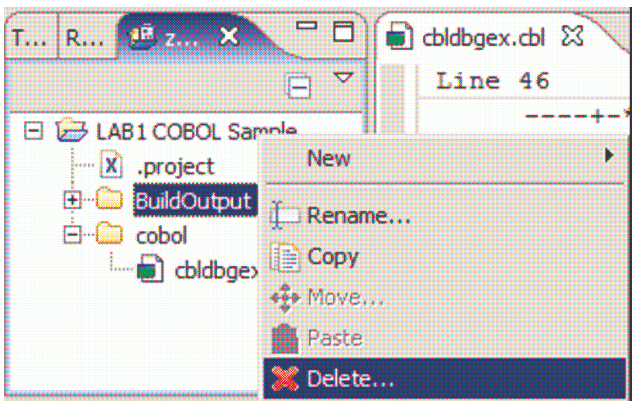


Anschließend wird das Help-Fenster geschlossen.

Übersetzen, Linken und Ausführen des lokalen COBOL-Programms

Erzeugen eines ausführbaren COBOL-Programms

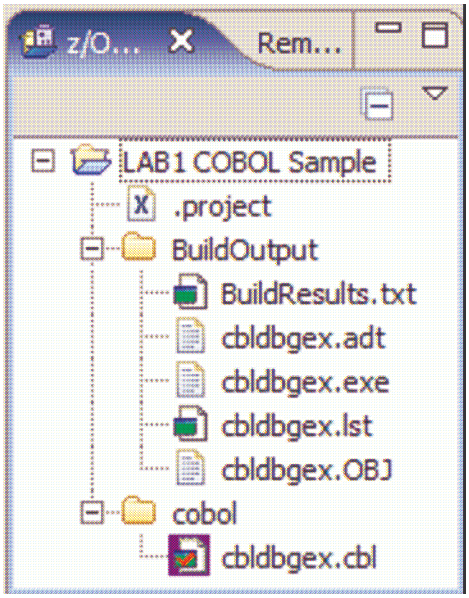
Nachdem Sie die Systhax des COBOL-Programms erfolgreich geprüft haben, kann letzteres lokal ausgeführt werden. Um ausführbaren Code zu erzeugen, wird der existierende Code gelöscht.
Selektieren Sie den Ordner „BuildOutput“ und benutzen Sie das Kontextmenue, um ihn zu löschen:



Der Grund dafür ist der, dass durch das Löschen des BuildOutput-Ordners gezeigt werden soll, wie er wieder erzeugt wird.

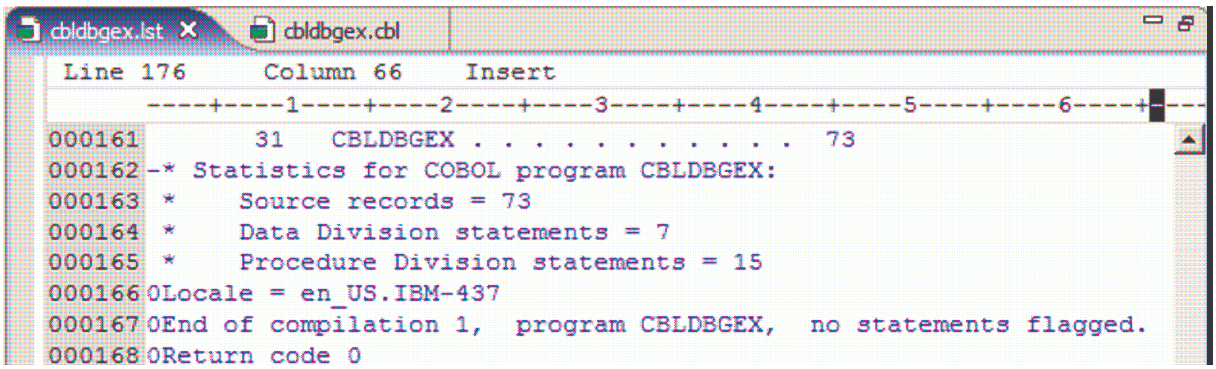
Klicken Sie auf „Delete“ in dem Pop-up-Fenster.

Benutzen Sie die z/OS Project View, expandieren Sie das LAB1 COBOL Sample, ein Klick mit der rechten Mousetaste auf cbldbgex.cbl und „Nominate as Entry Point“ selektieren. Wenn mehrere Programme in demselben Projekt enthalten sind, wird sonst auch eine DLL mit dem Projekt-Namen generiert. „Nominate as Entry Point“ ist die letzte Aktion. Da das Programm als Eintrittspunkt ausgewählt wurde, wird eine rote Prüfmarke in dem Icon des Programms erscheinen.
Um das Programm zu erstellen, klicken Sie auf das Projekt „LAB1 COBOL Sample“ und selektieren „Rebuild Project“.
Als Ergebnis wird „BuildOutput“ wieder erzeugt. In der Remote Error List View sollte kein Fehler auftreten. Es wird ein *.exe File generiert, optional könnte statt dessen ein DLL erzeugt werden. Letzteres wird für CICS TS benötigt!



Prüfen der lokalen COBOL-Übersetzung und des Link Edit-Schritts

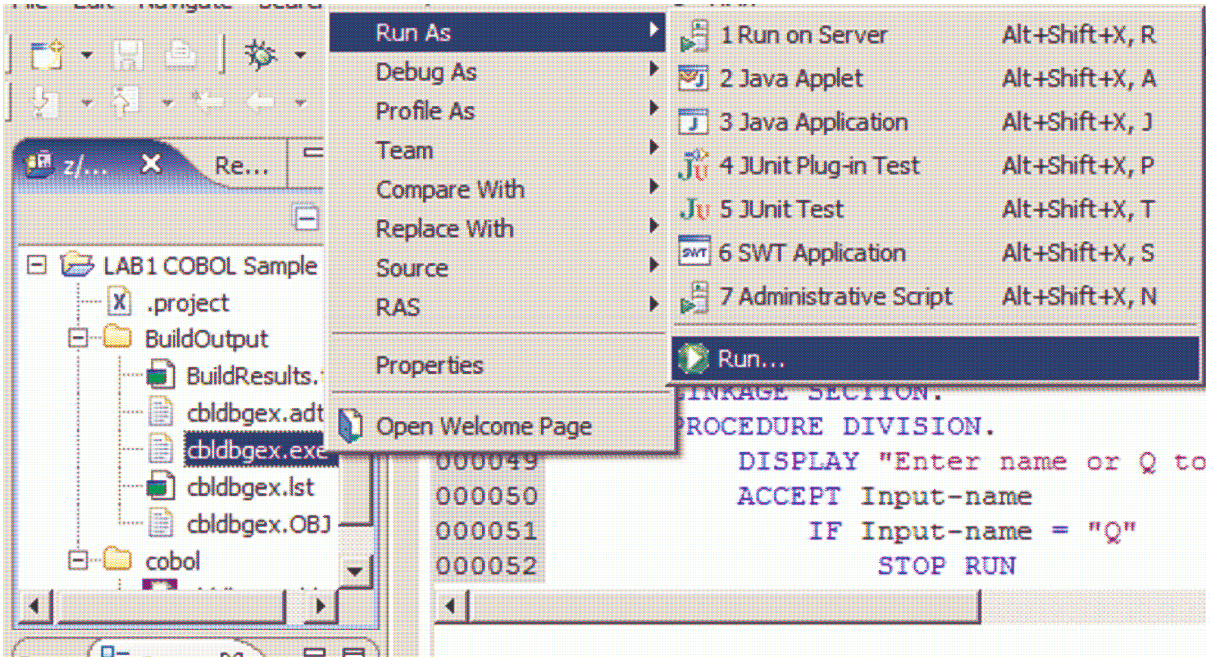
Der COBOL-Compiler speichert den Ausgang im BuildOutput-Ordner ab. Um sich das Übersetzungsergebnis anzusehen, klicken Sie auf cbldbgex.lst. Gehen Sie an das Ende der Liste und überprüfen Sie, ob dort Fehler aufgelistet sind. Benutzen Sie Ctrl + End, um an das Ende zu gelangen. Schließen Sie den Editor.



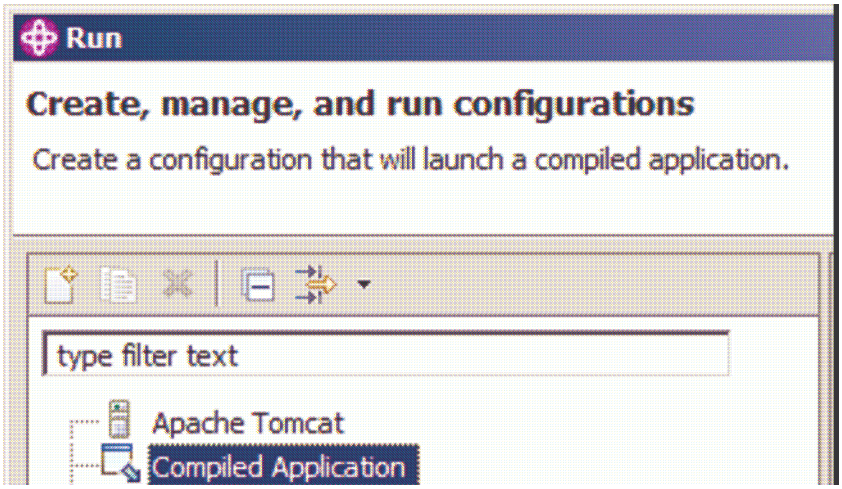
Erzeugen einer lauffähigen Konfiguration und Ausführung des COBOL-Programms

Da ein Programm oft für mehrere Läufe entwickelt wird, sollte eine Debug-fähige Konfiguration generiert werden. Letztere erleichtert die Ausführung und das Debugging eines spezifischen Programms. Um eine derartige Konfiguration zu erzeugen, wird das übersetzte COBOL-Programm folgendermaßen behandelt:

Rechter Mouseklick auf cbldbgex.exe und Run As → Run ... (nicht Run on Server)

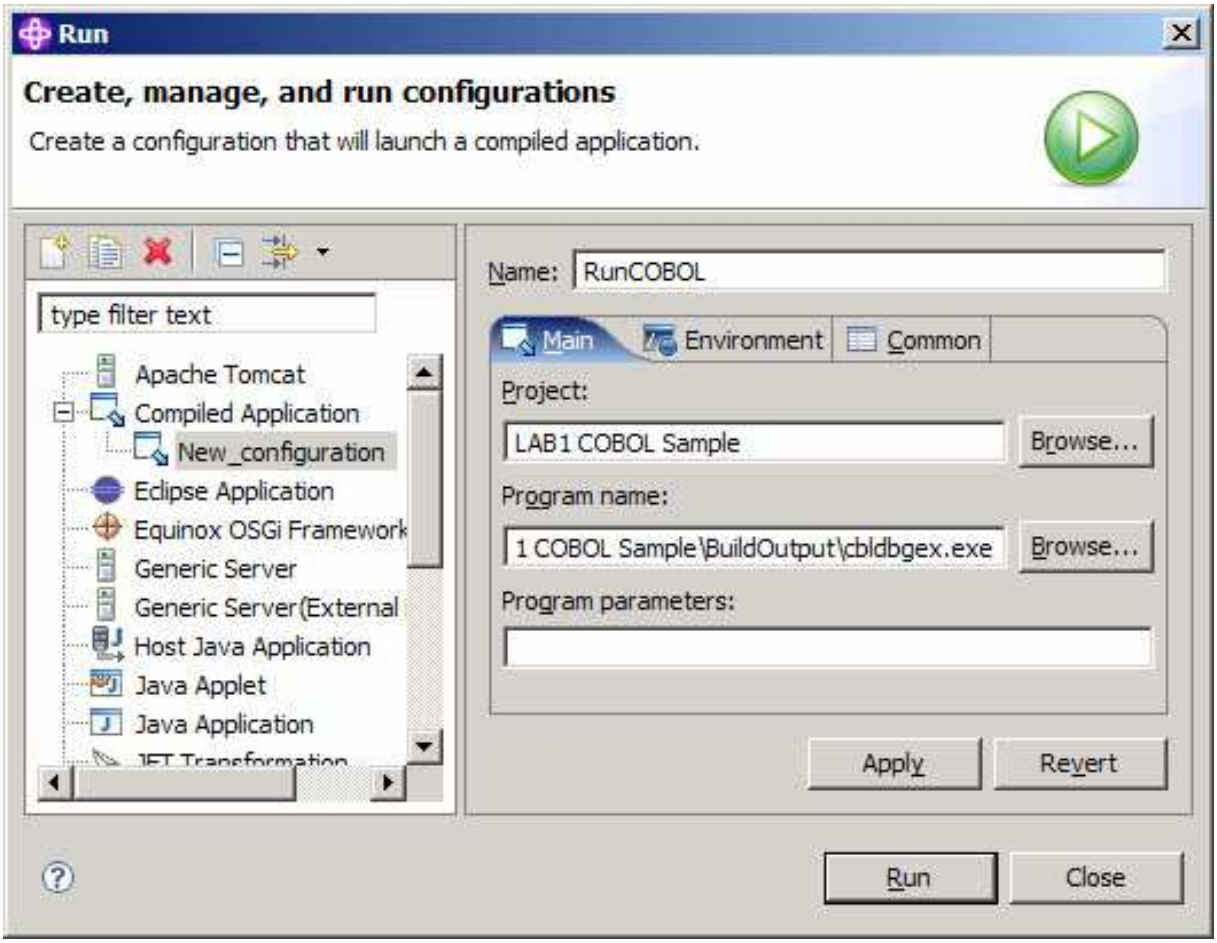


Selektieren Sie „Compiled Application“ und klicken Sie auf den New Button. Das ruft die notwendige Konfiguration und die Eingangsfelder auf, die auf der rechten Seite der Dialogbox angezeigt werden.

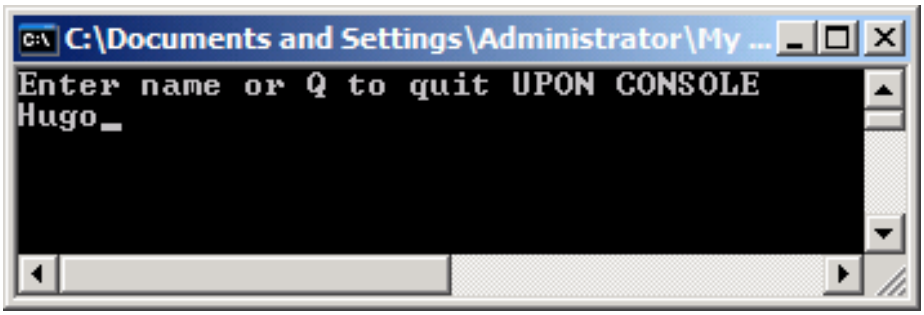


Geben Sie in dem Name-Feld einen Namen wie RunCOBOL ein, nutzen Sie den Browse Button, um cbldbgex.exe im BuildOutput-Ordner zu lokalisieren und klicken Sie auf „Open“.

Ihr ausführbarer COBOL-Modul sollte sich in C:\Documents and Settings\dev\IBM\rationalspd7.0\workspace\LAB1 COBOL Sample\BuildOutput\cbldbgex.exe befinden.
Klicken Sie auf „Apply“, um die Änderungen zu speichern. Anschließend klicken Sie auf „Run“. Die einzelnen Schritte sind im nachfolgenden Screen abgebildet.



Die Ausführung beginnt, und Sie werden den folgenden Dialog sehen. Geben Sie irgendwelche Daten (z. B. „Hugo“) oder „Q“ (Quit) in dem Dialog ein und betätigen Sie „Enter“.

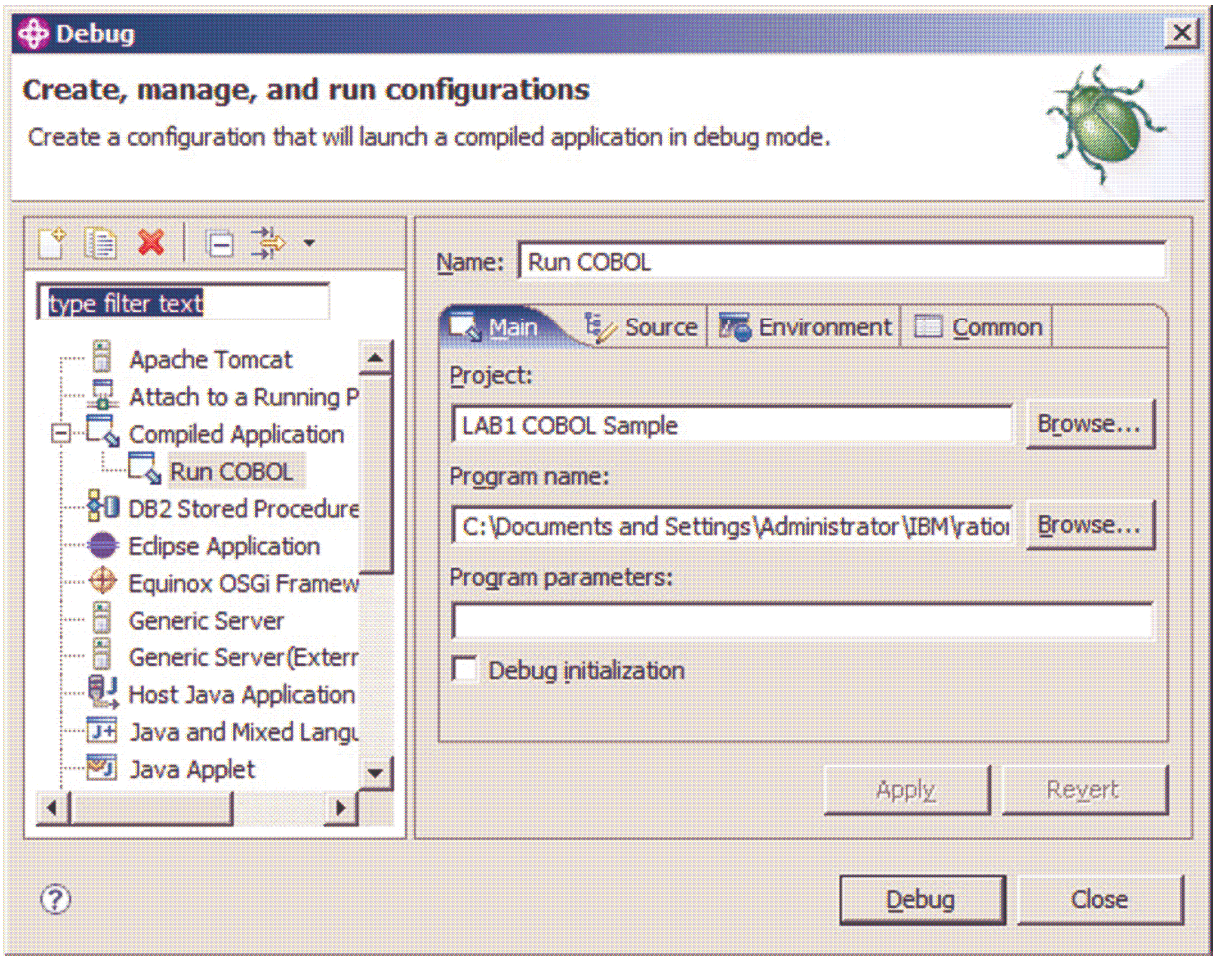


Um sicher zu sein, dass der Code ausgeführt wird wie vorgesehen, sollte der Code debugged werden.

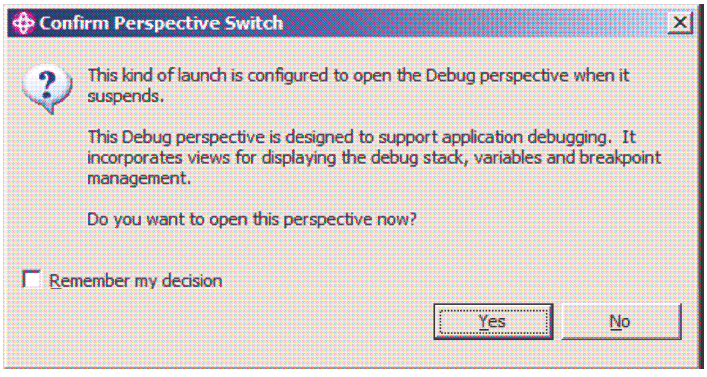
Testen und Debuggen des lokalen COBOL-Programms

Lokales Debuggen des COBOL-Programms

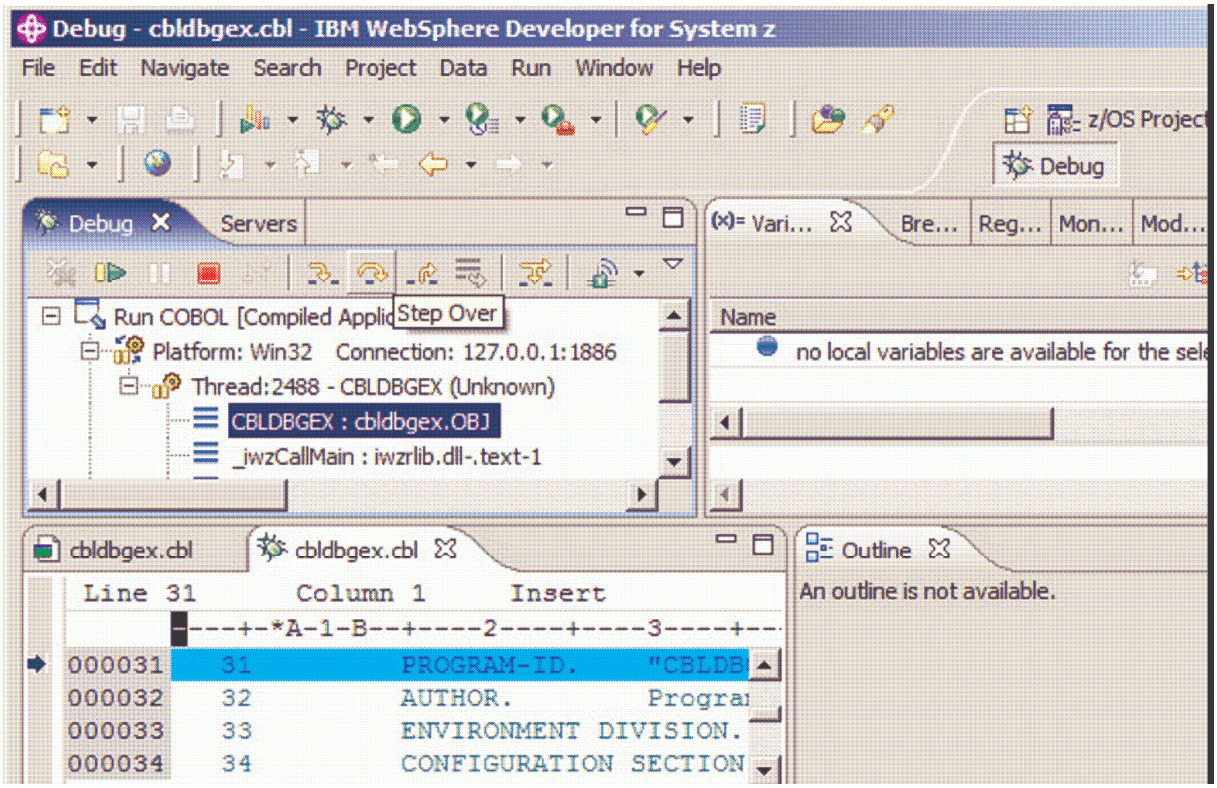
Da Sie bereits die entsprechende Konfiguration generiert haben, genügt ein rechter Mouseklick auf cbldbgex.exe und Selektieren von Debug As → Debug ... Wenn sich das Debug-Fenster öffnet, selektieren Sie „RunCOBOL“ und klicken auf „Debug“ entsprechend folgendem Screen:



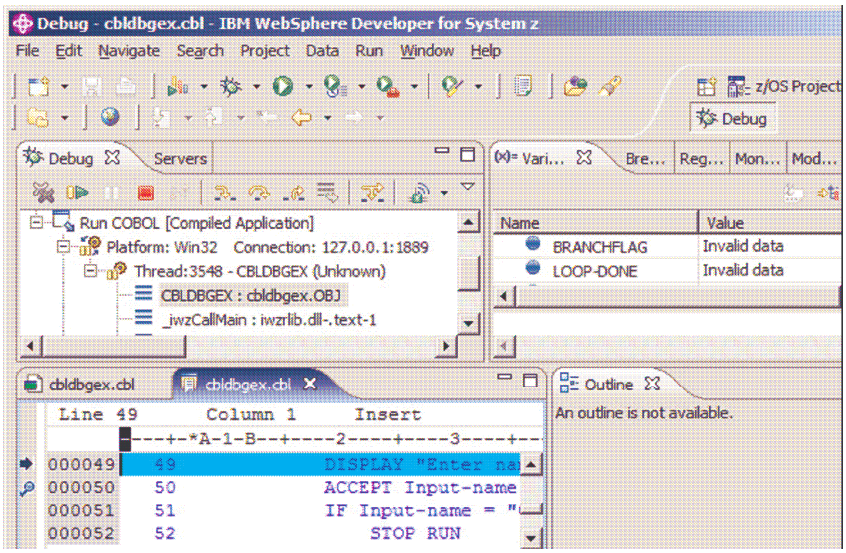
Im unteren Dialog werden Sie gefragt, ob Sie zur Debugger-Perspektive wechseln wollen. Klicken Sie auf „Yes“. Gegebenenfalls müssen Sie das Console/DOS-Fenster minimieren, um den Dialog zu sehen.



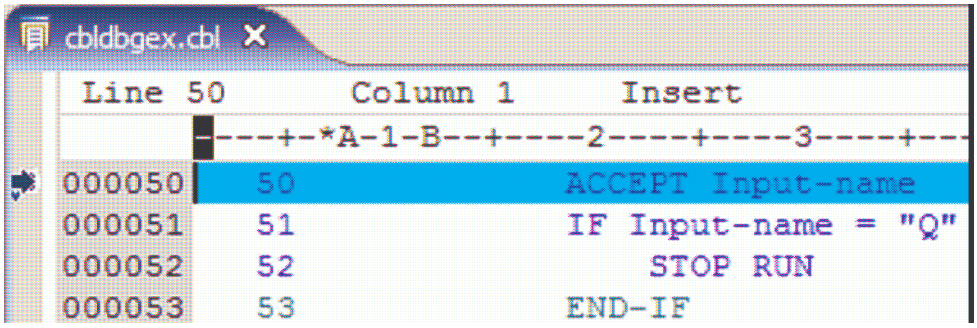
Die Debugger-Perspektive wird geöffnet, und der Debugger startet. Benutzen Sie die Debug View und klicken Sie auf das „Step Over“ Icon oder auf die Taste F6 (nicht Step Into benutzen!).



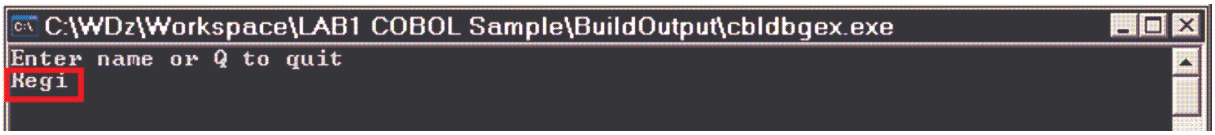
Addieren Sie einen Breakpoint zu dem ACCEPT Statement. Zu diesem Zweck bewegen Sie die Mouse vor die Zeilen-Nummer und führen einen Doppelklick, wie unten angegeben, aus. Den Breakpoint zeigt ein kleiner Kreis an:



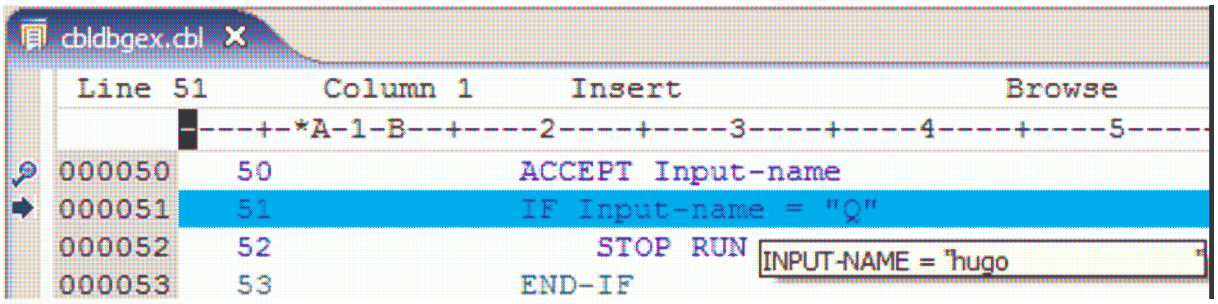
Der Breakpoint bewirkt, dass die Ausführung eines Threads an dieser Stelle unterbrochen wird. Die Ausführung des Programms wird weitergeführt bis zum nächsten Breakpoint. Klicken Sie auf das Resume Icon oder drücken Sie die Taste F8. Die Ausführung wird an dem ACCEPT Statement gestoppt:



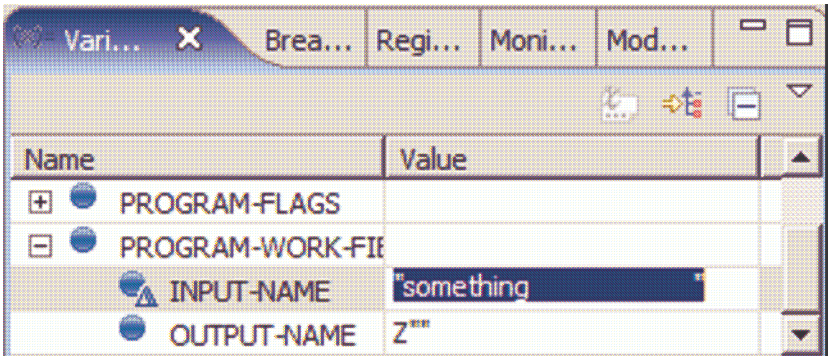
Das Statement wird ausgeführt, indem Sie auf das „Step Over“ Icon klicken oder die Taste F6 betätigen. Die Kommandos werden wirkungslos, wenn Sie zum DOS-Fenster gehen und eine Antwort eingeben.
Stellen Sie die ursprüngliche Größe des DOS-Fensters wieder her und beantworten Sie die Frage, wie unten in dem Screen angegeben. Geben Sie irgendetwas ein wie z.B. Ihren Namen, betätigen Sie die Enter-Taste und minimieren Sie das DOS-Fenster wieder, um das Debugging fortzusetzen (Fenster nicht schließen!):



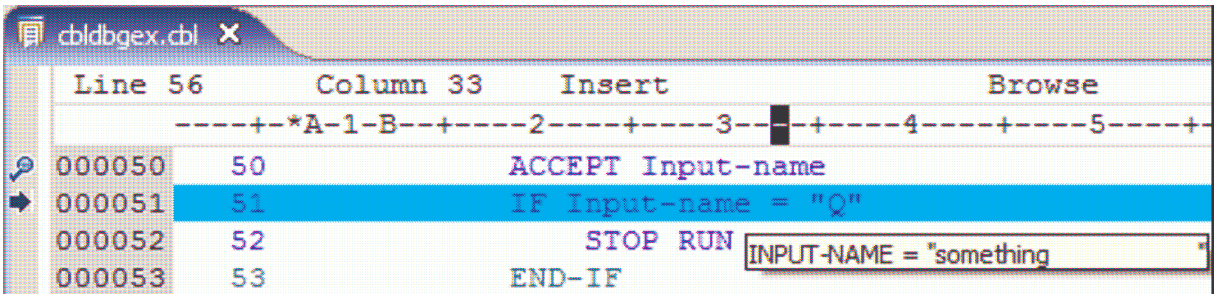
Um die Variablen-Inhalte wie z.B. INPUT-NAME zu überprüfen, bewegen Sie den Cursor zu dem Daten-Namen und warten einige Sekunden. Der Inhalt wird, wie unten dargestellt, erscheinen:



Sie können das Feld selektieren und auf das Kontextmenue „Monitor Expression“ klicken. Ein Variablen- Wert wird auch in der Variablen View gezeigt. Klicken Sie auf das Variablen-Feld, expandieren Sie „PROGRAM-WORK-FIELDS“ und ändern den „INPUT-NAME“ zu „something“. Klicken Sie auf „INPUT-NAME“ und überschreiben Sie den Wert.

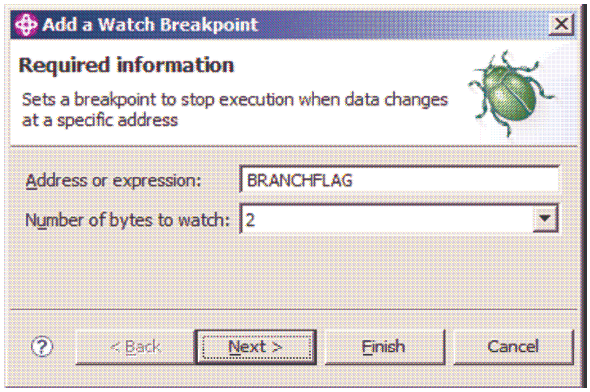


Wenn Sie den Cursor über den Input-Namen der Variablen bewegen, dann hat sich sein Wert verändert:

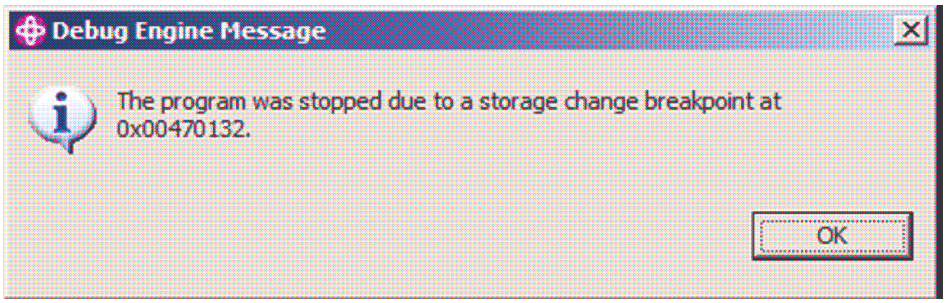


Klicken Sie auf die Breakpoint-Tabelle, um zu sehen, ob alle Breakpoints dem Programm zugeordnet sind.

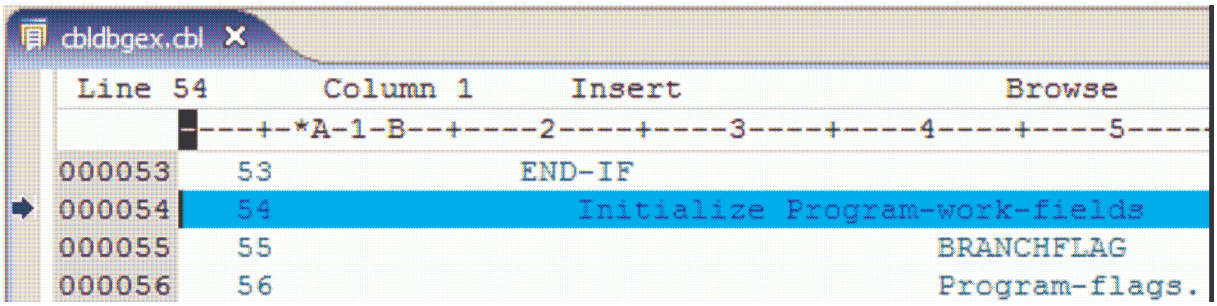
Es wird nun ein Watchpoint hinzuaddiert: Klicken Sie mit der rechten Mousetaste in die Breakpoint View und selektieren Sie „Add Breakpoint → Watch...“ Geben Sie im Feld „Address“ BRANCHFLAG und im Feld „Number of bytes“ die Ziffer „2“ ein, da gewünscht wird, an diesem Feld zu stoppen, wenn es modifiziert wird. Anschließend klicken Sie auf „Next“ und „Finish“.



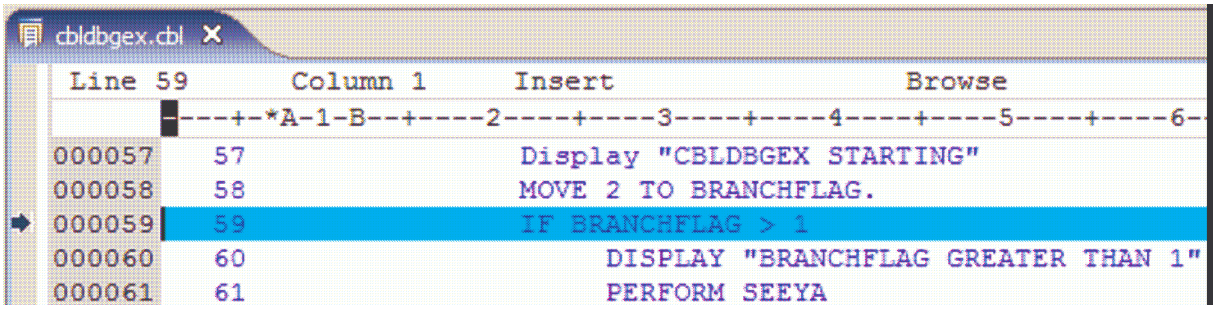
Klicken Sie auf das Resume Icon oder betätigen Sie die Taste F8. Sie werden eine Nachricht erhalten, die besagt, dass der Speicher geändert wurde:



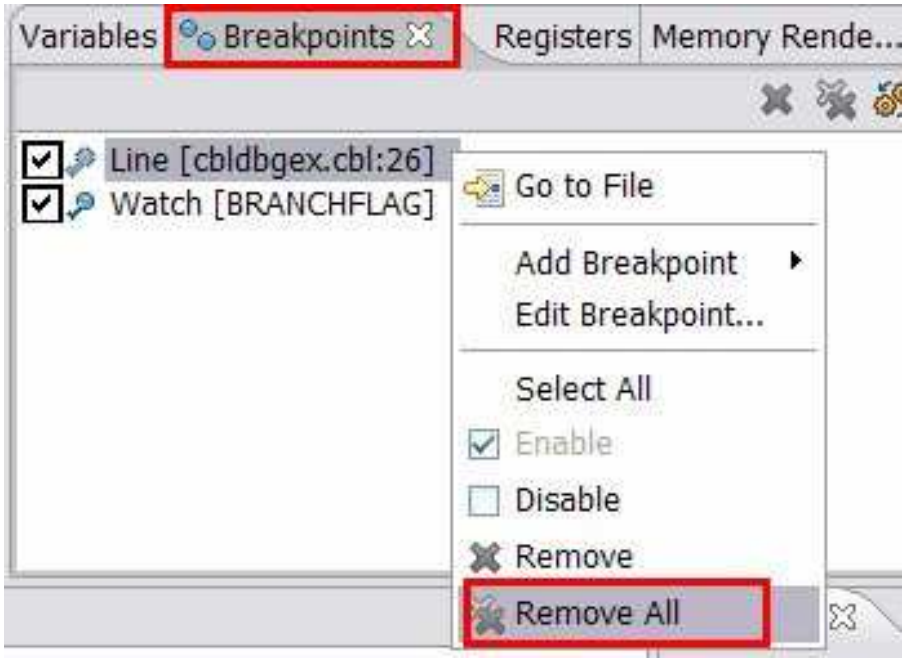
Der Grund ist die Initialisierung des COBOL-Statements, das zu BRANCHFLAG bewegt wird, und dieses Feld wird geändert. Klicken Sie auf „OK“. Die Ausführung wird, wie unten gezeigt, gestoppt:



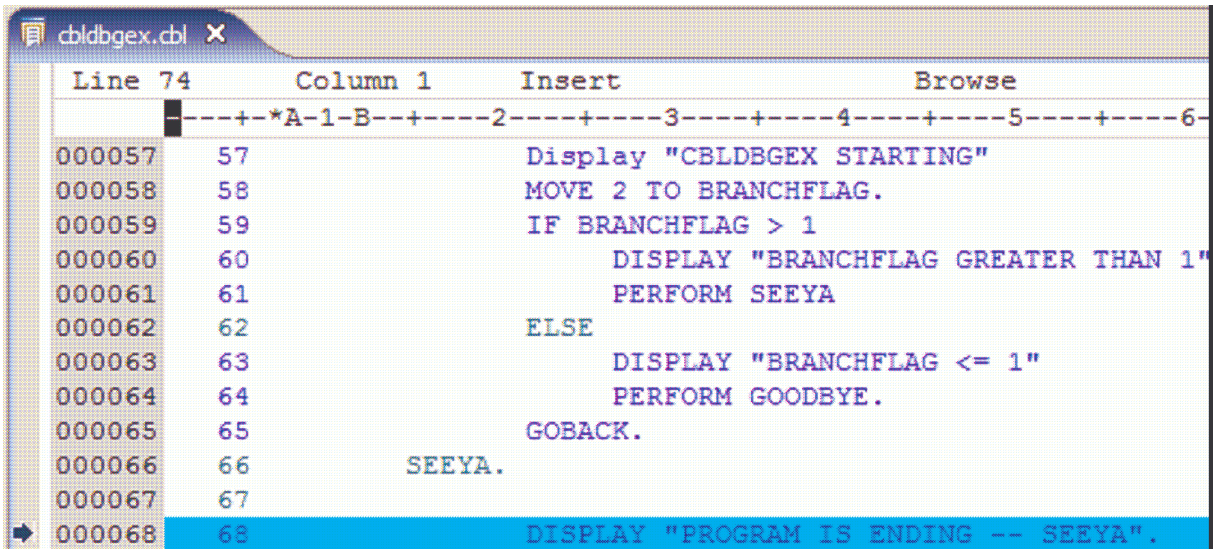
Klicken Sie wieder auf das Resume Icon oder betätigen Sie die Taste F8. Sie werden wieder eine Nachricht über die Speicher-Änderung erhalten. Sie müssen auf „OK“ klicken, da das Feld BRANCHFLAG wieder verändert wurde (hat den Wert 02).



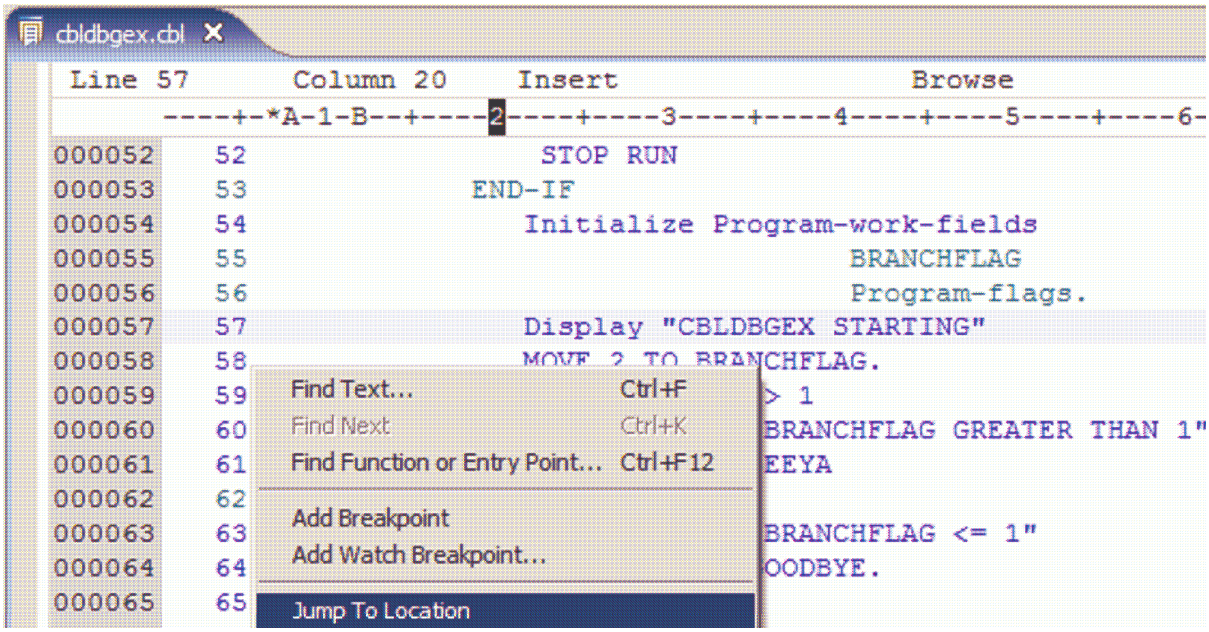
Entfernen Sie den Breakpoint (Sie können den Breakpoint auch „disabled“ setzen anstatt ihn zu entfernen; das erlaubt Ihnen, diesen später wieder „enable“ zu setzen). Gehen Sie zu Breakpoints View und selektieren „Remove All“ vom Kontextmenue (rechter Mouseklick):



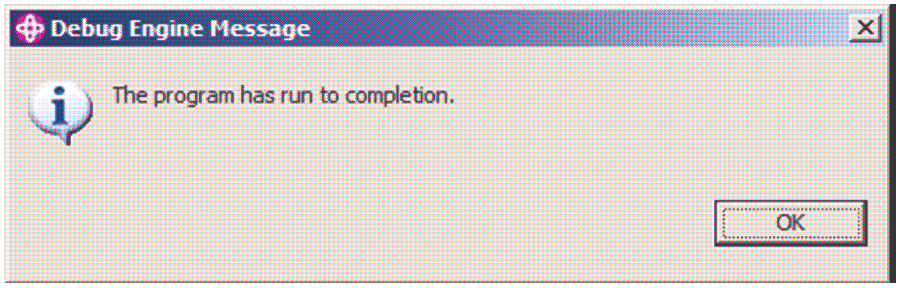
Klicken Sie auf das „Step Over“ Icon oder betätigen Sie die Taste F6, bis Sie die Zeile wie unten erreichen:



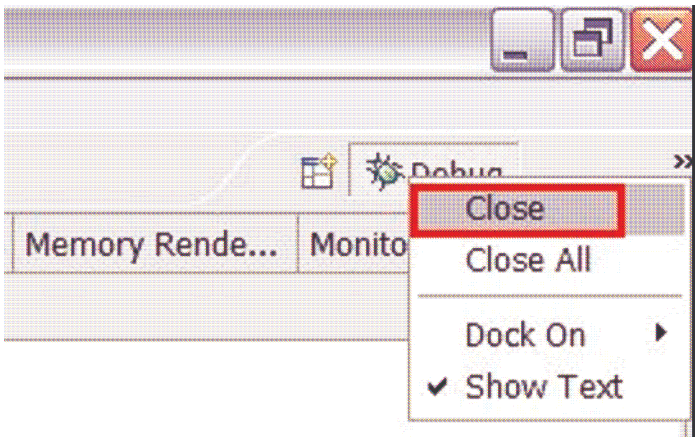
Scrollen Sie zurück zur COBOL-Zeile 58 und benutzen Sie das Kontextmenue, selektieren Sie „Jump To Location“. Die Ausführung wird zurück zu der Zeile 58 bewegt.



Spielen Sie etwas mit dem Debugger. Sie können dieselben Debugging-Funktionen für Batch, CICS, IMS oder DB2-Programme benutzen. Sie bewegen sich im z/OS durch die Installierung des interaktiven Debuggers auf diesen Plattformen. Wenn gewünscht, kann das entsprechende Icon oder die Taste F8 benutzt werden. Nach dem Programm-Ende wird eine Nachricht wie unten angezeigt:



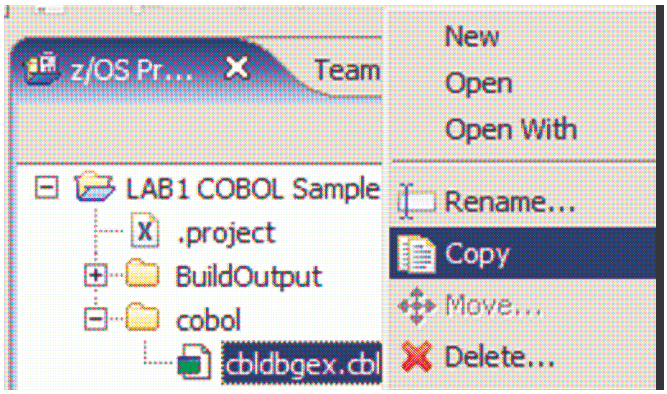
Klicken Sie auf „OK“ und schließen Sie die Debug-Perspektive (unten rechts).



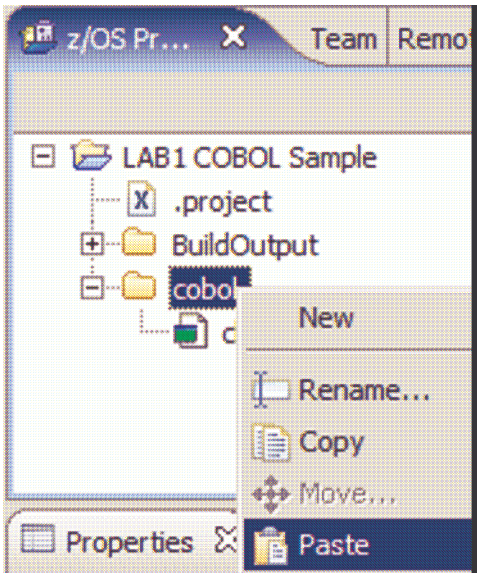
Visueller Vergleich zweier COBOL-Programme

Vergleich zweier Versionen desselben Files

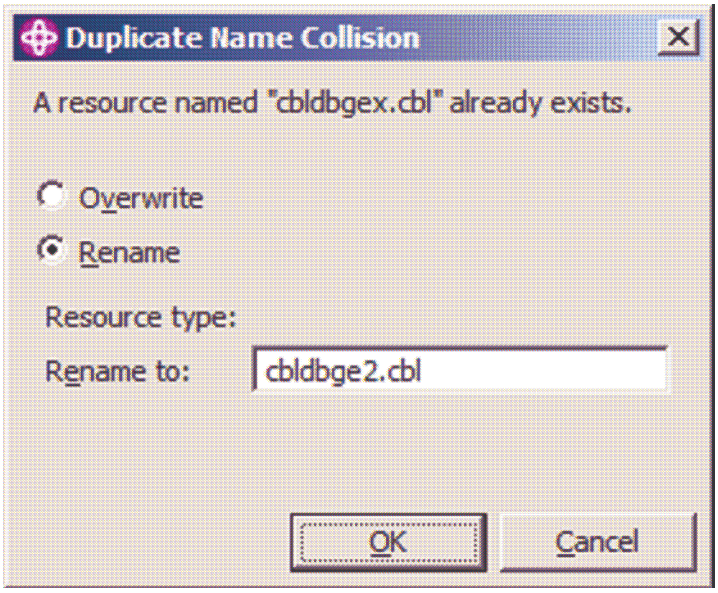
Es gibt Situationen, in denen Sie zwei Files haben (können COCOL, JCL, etc. sein), die Sie vergleichen möchten, z.B.:
Benutzen Sie die z/OS Project View und fertigen Sie eine Kopie von cbldggex.cbl, wie unten gezeigt, an. Dazu klicken Sie auf die rechte Mousetaste und selektieren „Copy“:



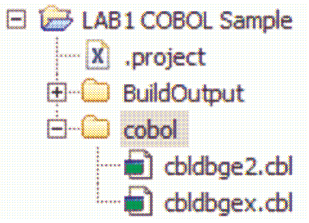
Klicken Sie mit der rechten Mousetaste auf den Ordner „cobol“ und selektieren Sie „Paste“:



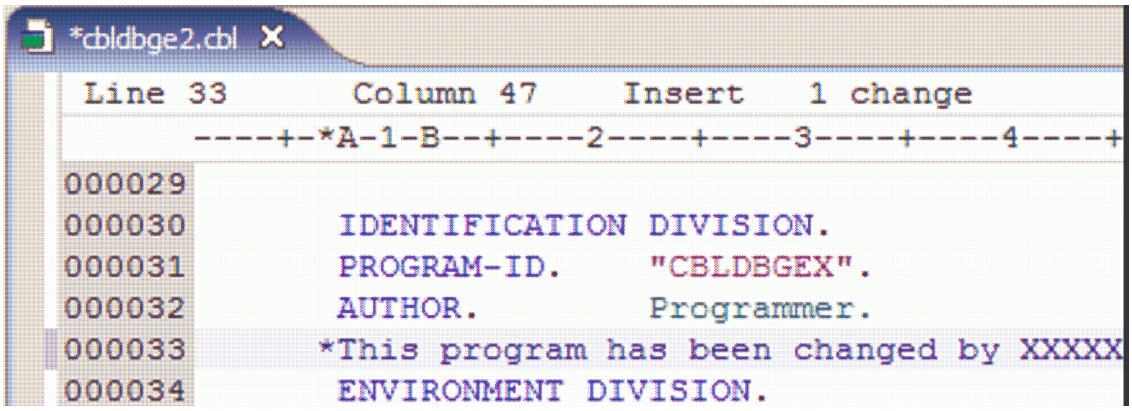
Da dieser Name schon existiert, erscheint eine Fehlermeldung. Selektieren Sie „Rename“ und ändern Sie den Namen zu cbldbge2.cbl. Klicken Sie auf „OK“ wie unten dargestellt:



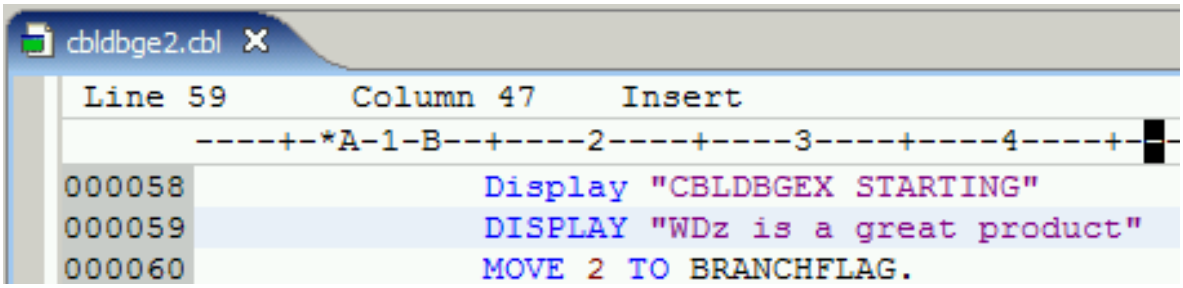
Ihr COBOL Directory-Baum sollte wie folgt aussehen:



Anschließend werden einige Änderungen an dem Quellprogramm cbldbge2.cbl vorgenommen. Editieren Sie cbldbge2.cbl (Doppelclick) und fügen Sie eine Kommentarzeile unterhalb von „AUTHOR“ durch Nutzung des Line Commands „i“ ein:

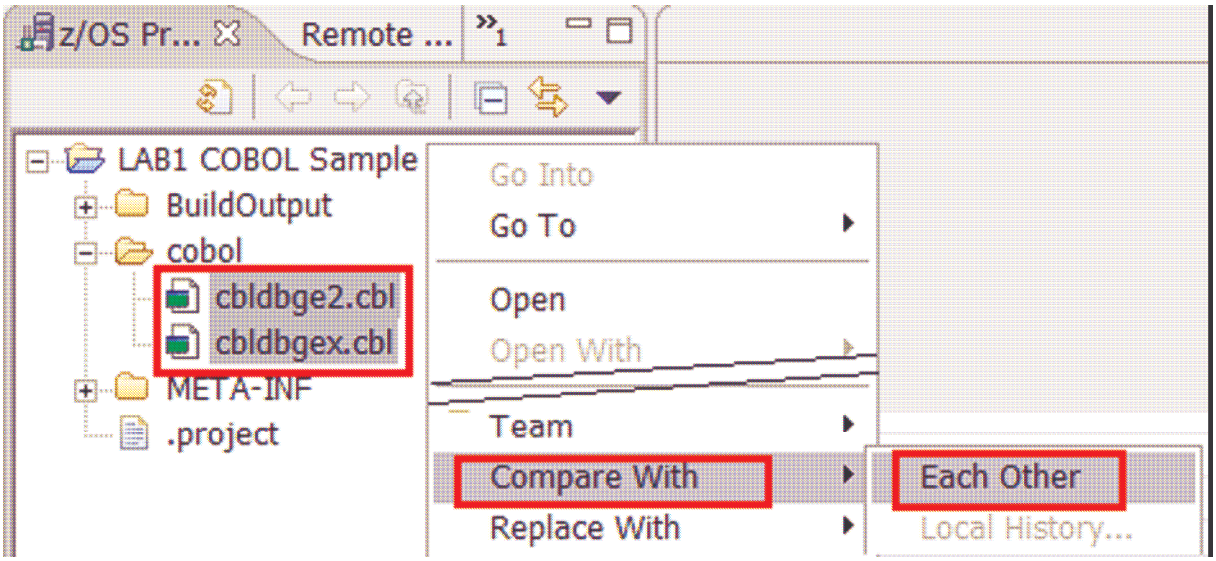


Addieren Sie zusätzlich ein neues Statement, wie unten angezeigt, hinzu:

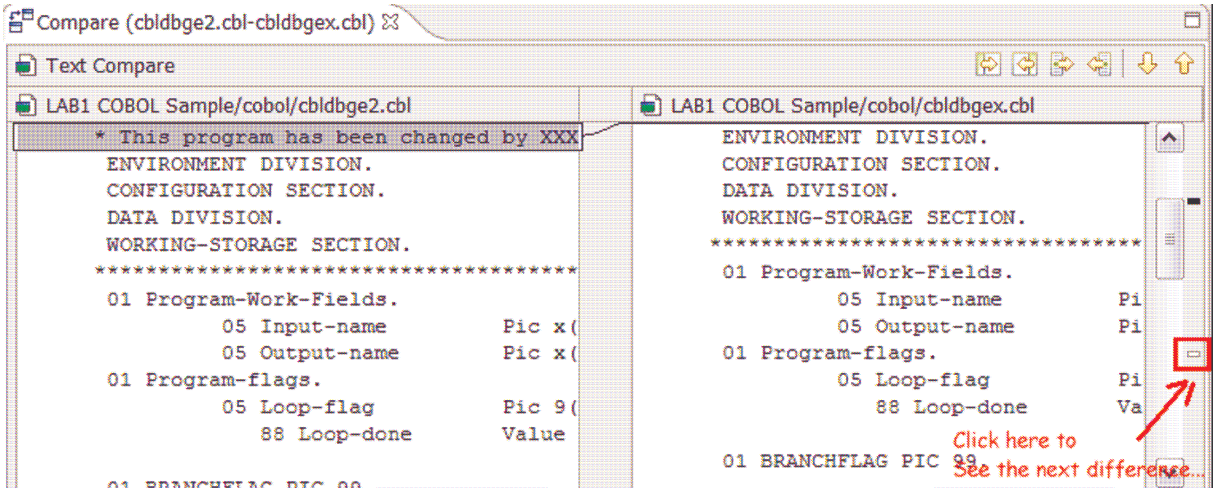


Speichern Sie das editierte File mittels Ctrl + S ab und schließen Sie den Editor.

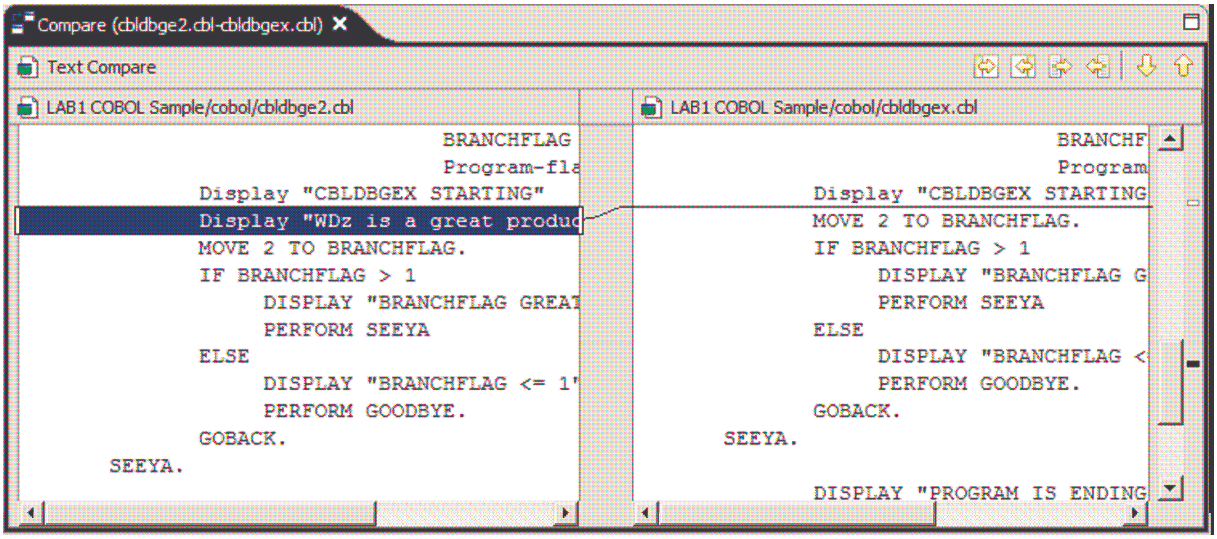
Diese beiden Programme cbldbgex.cbl und cbldbge2.cbl sollen nun verglichen werden. Selektieren Sie beide Programme. Selektieren Sie im Kontextmenue Comprae With → Each Other:



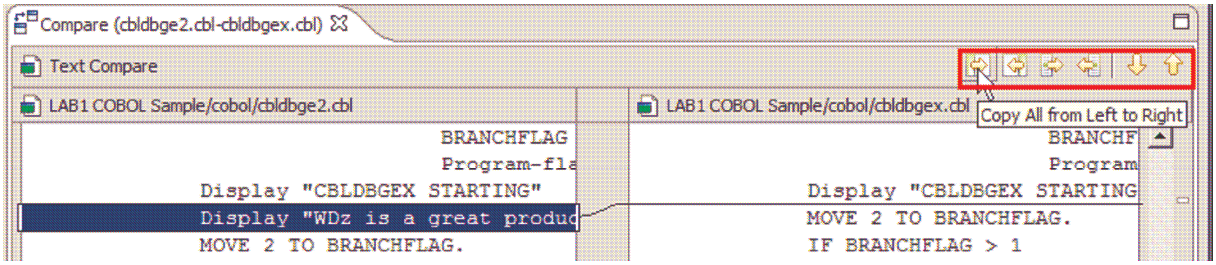
Sie werden beide Programme nebeneinander wie im unteren Screen sehen:



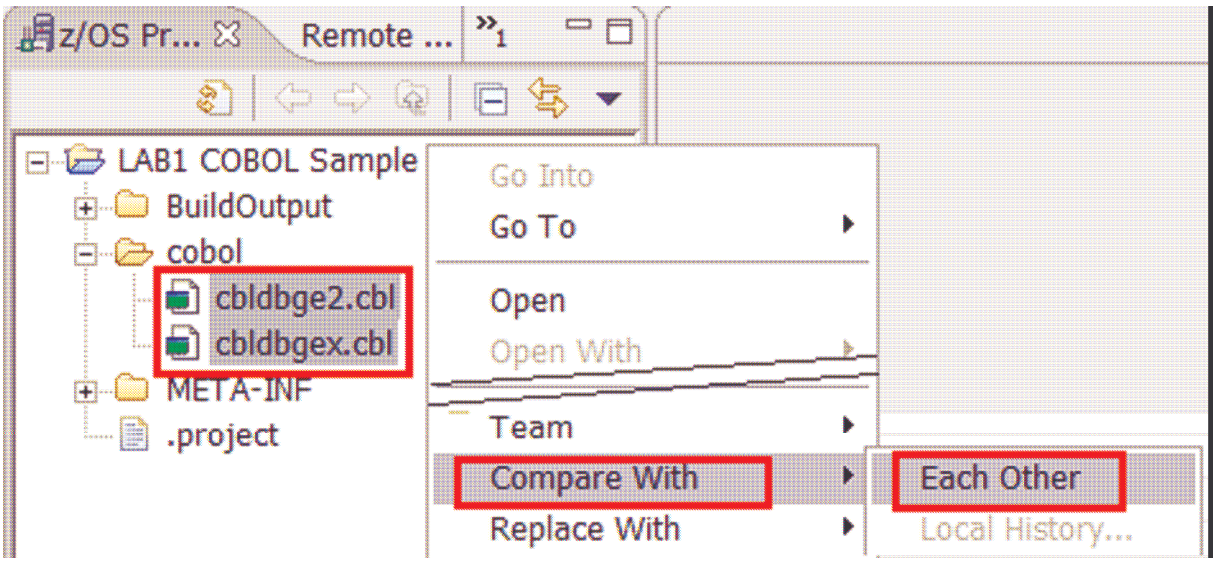
Klicken Sie auf das Icon rechts, um den Unterschied festzustellen:



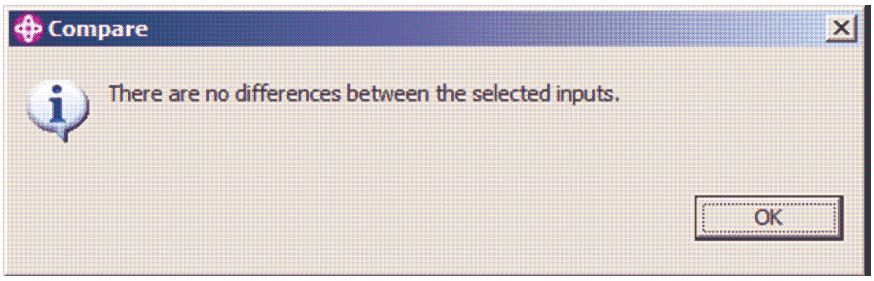
Dieses stellt eine nützlich Funktion dar, die lokal oder innerhalb von z/OS benutzt werden kann.
Sie werden in der Lage sein, die Unterschiede durch Verwendung der Icons, die sich rechts oben befinden, zu beseitigen.
Klicken Sie auf das erste Icon (Copy All from Left to Right):



Speichern und schließen Sie den Vergleich. Führen Sie erneut den Vergleich durch wie unten:



Sie werden die untenstehende Nachricht erhalten, da keine Unterschiede mehr zwischen beiden Programmen vorhanden sind.



Aufgabe:	*Entwickeln Sie ein COBOL-Programm (ca. 20-30 Code-Zeilen). Speichern Sie dieses im Ordner „Workstation COBOL“ ab. Überprüfen Sie die File-Synthax, benutzen Sie die Funktionen des Editors. Übersetzen und linken Sie das COBOL-Programm, führen Sie es aus. Benutzen Sie für die Testhilfe die Debugging-Funktionen durch das Setzen von Breakpoints. Führen Sie den visuellen Vergleich zweier ähnlicher COBOL-Programme durch.*