Micro Focus Enterprise Developer チュートリアル

メインフレーム PL/I 開発: JCL

Eclipse 編

1. 目的

本チュートリアルでは、PL/I 言語で書かれたソースをオープン環境へ移行後、Eclipse を使用したプロジェクトの作成、コンパイル、JCL の 実行、デバッグまでを行い、その手順の習得を目的としています。

2. 前提

• Windows 開発環境に Enterprise Developer 2.3 for Eclipse がインストール済であること。

3. チュートリアル手順の概要

- 1. チュートリアルの準備
- 2. Eclipseの起動
- 3. メインフレーム PL/I プロジェクトのインポート
- 4. プロジェクトプロパティの確認
- 5. ビルドの実行
- 6. Enterprise Server の設定
- 7. Enterprise Server 開始と確認
- 8. JCL の実行
- 9. PL/I ソースのデバッグ
- 10. 終了処理



3.1 チュートリアルの準備

サンプルプログラムに関連する資源を用意します。

- 1) Eclipse のワークスペースで使用する「work」フォルダを C ディレクトリ直下に作成します。
- 2) 製品をインストールしたフォルダ配下に含まれているサンプルプログラム「JCLDEMO」フォルダを、作成した C:¥work ヘ コピーします。
 - 例) C:¥Users¥Public¥Documents¥Micro Focus¥Enterprise Developer¥Samples¥PLI-Eclipse¥JCLDEMO



3.2 Eclipse の起動

1) Micro Focus Enterprise Developer for Eclipse を起動します。

Enterpris	e Developer for Eclipse	j.	Adobe Reader XI	
Enterpris	e Developer for Visual Studio Enterpri	se De	eveloper for Eclipse の起動	→
▲ ドキュメント	- Eclipse		Micro Focus License Manager	→
לעעב‡א 😯	- Visual Studio 2012		Micro Focus Enterprise Developer	\rightarrow

2) 前項で作成した「C:¥work」をワークスペースへ指定して、[OK] ボタンをクリックします。

Twe I	ワークスペース・ランチャー	×
ワークスペースの選択		
Eclipse は、ワークスペースと呼ばれ このセッションに使用するワークスペー	るフォルダにプロジェクトを保存します。 ス・フォルダを選択してください。	
ワークスペース(<u>W</u>): C:¥work		♥ 参照(<u>B</u>)
□ この選択をデフォルトとして使用し	、今後この質問を表示しない(<u>U</u>)	OK キャンセル

3) [ようこそ]タブが表示されますので、[Open PL/I Perspective] をクリックして、PL/I パースペクティブを開きます。



4) PL/I パースペクティブ表示後、[プロジェクト] プルダウンメニューの [自動的にビルド] を選択して、これをオフにします。

) 編	集(E) ナビゲート(N) 検索	プロジェクト(P)) 8	編集(E) ナビゲート(N) 検	素 プロジェクト(P)
	プロジェクトを開く(E) プロジェクトを閉じる(S)			プロジェクトを開く(E) プロジェクトを閉じる(S)	
010	すべてビルド(A) プロジェクトのビルド(B) ワーキング・セットのビルド(W)	Ctrl+B		すべてビルド(A) プロジェクトのビルド(B) ワーキング・セットのビルド(W	Ctrl+B
~	クリーン(N) 自動的にビルド(M)		\rightarrow	クリーン(N) 自動的にビルド(M)	-

5) 既存ファイルのインポート時、自動的にコンパイル指令が指定される機能が用意されていますが、本チュートリアルではこれを 解除します。 [ウィンドウ] プロダウンメニューの [設定] > [Micro Focus] > [PL/I] > [指令の確定] > [指令の確定 を行う] チェックボックスをオフにして [OK] ボタンをクリックします。



- 3.3 メインフレーム PL/I プロジェクトのインポート
 - 1) 用意したサンプルプロジェクトをインポートします。 [ファイル] プルダウンメニューから [インポート] を選択し、インポートウィン ドウにて [General] > [既存プロジェクトをワークスペースへ] を選択後 [次へ] ボタンをクリックします。

「「「」 インポート	– 🗆 🗙
選択 アーカイブ・ファイルまたはディレクトリーから新規プロジェクトを作成します。	Ľı
 インボート・ソースの道択(<u>S</u>): フィルタ入力 ② General ③ アーカイブ・ファイル ④ ファイル・システム ③ 既存プロジェクトをワークスペースへ □ 設定 ▶ ▷ CVS 	<
(?) < 戻る(L) 次へ(N) > 終了(E)	キャンセル

2) [ルート・ディレクトリの選択] へ「C:¥work¥JCLDEMO」を指定すると、このフォルダに含まれるプロジェクトが表示されま す。チェックをオンにした状態で [終了] ボタンをクリックします。

「「」 インポート	- 🗆 🗙
プロジェクトのインボート 既存の Eclipse プロジェクトを検索するディレクトリーを選択します。	
 ・・ティレクトリーの選択(I): C:¥work¥JCLDEMO アーカイブ・ファイルの選択(A): プロジェクト(P): 	参照(<u>R</u>) 参照(<u>R</u>)
JCLDEMO (C:¥work¥JCLDEMO 直接アクセス) 選択	べて選択(<u>S)</u> をすべて解除(<u>D</u>) 更新(<u>E</u>)
オプション 「オストしたプロジェクトを検索(<u>H</u>) 「プロジェクトをワークスペースにつビー(<u>C</u>) 「ワークスペースに既に存在するプロジェクトを隠す(<u>i</u>)	
ワーキング・セット □ ワーキング・セットにプロジェクトを追加(I) ワーキング・セット(<u>0</u>): v	選択(<u>E</u>)
⑦ < 戻る(B) 次へ(N) > 終了(E)	キャンセル

3) [PL/I エクスプローラー] にインポートしたプロジェクトが表示されます。

🔁 PL/I Explorer 🛛

JCLDEMO



4) [JCLDEMO] プロジェクトを展開すると PL/I ソースや JCL などが確認できます。

a 🐸 JCLDEMO

- ▲ PL/I ソースファイル ト ● fetched.pli
- Fetcher.pli
- jcldemo.pli
- k xmldemo.pli
- ⊿ 💯 JCL ファイル in fetcher.jcl
 - jcldemo.jcl
 - xmldemo.jcl

3.4 プロジェクトプロパティの確認

プロジェクトの設定値を確認していきます。

1) [JCLDEMO] プロジェクトを右クリックして [プロパティ] を選択するとプロパティウィンドウが表示されます。

64-bit 稼働が指定されていますが、ここでは 32-bit OS で実行することを考慮して 32-bit 稼働へ変更します。

① [Micro Focus] > [ビルド構成] で [構成の管理] ボタンをクリックして構成管理ウィンドウを表示します。

7ィルタ入力	ビルド構成	<> ▼ <> ▼ <
▷ リソース J Micro Focus ビルド パス ▷ ビルド構成	x64 [使用中]	✓構成の管理
▶ プロジェクト設定 Project Facets	出力ディレクトリ名: bin¥x64¥debug	参照

② [ビルドの構成管理] ウィンドウでは [x86] のチェックボックスをオンにして [完了] ボタンをクリックします。

₩E]	ビルド構成の管理	×
ビルド構想	成の管理 中のビルド構成の選択やビルド構成の作成と削除をおこないます。	
D. @	∕ √ ¥	
構成の名	前	使用中
x64 		 ✓
?		完了

③ [Micro Focus] > [ビルド構成] ウィンドウへ戻り [x86] へ変更されたことと、プロジェクト配下の「bin¥debug」 フォルダへ実行ファイルが出力されることを確認後 [適用] ボタンをクリックします。

ビルド構成
x86 [使用中]
出力ディレクトリ名: bin¥debug
出力ディレクトリ名: bin¥debug

④ [Micro Focus] > [ビルド構成] > [PL/I リンク設定] を選択して内容を確認すると、32 ビット稼働する実行可 能ネイティブライブラリをオブジェクトとして生成することがわかります。

Micro Focus ビルドパス J ビルド構成	x86 [使用中]	✓ 構成の
 > PL/I コンパイル設定 > PL/I リンク設定 > Pセンブラ コンパイラ > アセンブラ リンカ 	設定: 21ルタテキストを入力 注	
イベント イベント	設定	値
BMS ▶ IMS ▲ PL/I コンパイル設定		32 bit すべて実行可能ネイティブライブラリ JCL

 [Micro Focus] > [プロジェクト設定] > [PL/I コンパイル設定] を選択して内容を確認すると、サンプルの内容に沿って、 「システム」には「MVS」が設定されており、デバッグ実行用ファイルを生成することがわかります。

Micro Focus	設定:	
ビルドバス J ビルド構成 BMS	フィルタテキストを入力	
▷ PL/I コンパイル設定	設定	値
▷ PL/I リンク設定	▲ 一般的なオプション	
アセンブラ コンパイラ	システム	MVS
アセンブラリンカ	デバッグ用にコンパイル (-debug)	はい
1/121	リストファイルを出力 (-1)	いいえ
⊿ フロジェクト設定	最適化レベル (-opt)	-noopt
BMS	エンディアン (-bigendian)	
⊳ IMS	EXEC プリプロセッサ オプション (-optexec)	
▷ PL/I コンバイル設定	追加オプション	
アセンブラ コンパイラ	⊿ 高度	
アセンブラリンカ	Dock ⊐ kt → B / kitalte)	11113

 [Micro Focus] > [プロジェクト設定] > [PL/I コンパイル設定] > [マクロ プリプロセッサ]を選択して内容を確認します。
 [追加オプション] には "-define debugit" が入力されています。この指定がある場合は、ソースコードに記述されている デバッグモード判断文で真となり、デバッガが起動します。

【デバッグモードの切り替え:下記値の有無】

⊿ 共	通マクロプリプロセッサ オプション	
	完全なリストを出力 (-full_list)	いいえ
	追加オプション	-define debugit

【ソースコード記述部: IF 文で真】

%if debu	git %then					
%DO;						
CALL	PLITEST('shlib	jcldemo.dll;env	JCLDEMO;br	START_DEBUG;br	AfterUndef;c','	',1);
STAR	T DEBUG:	-		_		
%END;	-					

"PLITEST"の引数や詳細に関しては下記の関連ヘルプを参照してください。

http://documentation.microfocus.com/help/topic/com.microfocus.eclipse.infocenter.enterprisedeveloper.eclipsewin/GUID-9460F14E-218C-46A5-9BCF-CF7FCF396519.html

ますはデバッガを起動しないで実行するため、[追加オプション]の値をクリアして [OK] ボタンをクリックします。

⊿	共通マクロプリプロセッサ オプション	
	完全なリストを出力 (-full_list)	いいえ
	追加オプション	

3.5 ビルドの実行

1) [プロジェクト] プルダウンメニューの [自動的にビルド] を選択して、これをオンすると自動的にビルドが実行されます。



2) [コンソール] タブでビルドの成功を確認します。

📃 コンソール 🛛 🔝 問題 📃 プロパティ
Micro Focus Build
os.init.unix:
init:
post.build.cfg.x86:
BUILD SUCCESSFUL Build finished with no errors.
Total time: 1 second

3) プロジェクトの bin¥debug フォルダ配下に目的の実行ファイルが作成されていることを確認してください。





3.6 Enterprise Server の設定

1) PL/I を実行するためのエンジンを搭載した Enterprise Server を作成します。 [サーバー エクスプローラー] タブの [ロ ーカル] を右クリックして [Administration ページを開く] を選択します。

😪 PL/I Explorer 📃 サーバー	・エクスプローラー 🛛 📄	R
▶ 🔜 ローカル [localhost:86]	新規作成(N)	•
	Administration ページを開く	Ctrl+F3

C:¥work¥JCLDEMO には Enterprise Server のサンプルが含まれており、これをインポートします。PL/I アプリケーションは 32 ビット稼働を指定したため、「C:¥work¥JCLDEMO¥plijcl_def」がインポート対象となります。64 ビットで稼働させる場合は「C:¥work¥JCLDEMO¥plijcl64_def」をインポートしてください。

Enterprise Server Administration 画面左側の [インポート] をクリックして、表示される下記項目へ前述のパスを 入力後、 [次へ] ボタンをクリックします。

サーバー情報のインポート (Page 1 of 4):



3) 画面の Page 2/4、3/4、ではそのまま [次へ] ボタンを、Page 4/4 では [OK] ボタンをクリックすると、[PLIJCL] とい う名前の 32 ビットアプリケーション稼働用 Enterprise Server が追加されます。





4) 設定を変更するため、[編集] ボタンをクリックします。



5) [サーバー] > [プロパティ] > [一般] タブで表示される画面の [構成情報] 欄を下記のように入力し、[Apply] ボタンを クリックします。

変更前;

[ES-Environment] JDEMO=C:¥Users¥Public¥Documents¥Micro Focus¥Enterprise Developer¥Samples¥"PLI-VS or PLI-Eclipse"¥JCLDEMO CODEWATCH_NOTIF=Y

変更後;

[ES-Environment] JDEMO=C:¥work¥JCLDEMO CODEWATCH_NOTIF=Y

自,情報

CODEWATCH_NOTIF 環境変数:

デバッガを使用する開発用サーバーに指定します。本番サーバーにはパーフォーマンスの観点から排除することを推奨します。

6) [サーバー] > [プロパティ] > [MSS] > [JES] > [一般] タブで表示される画面の各項目を確認します。

項目名	説明
メインフレーム サブシステム サポート有効	[MSS] タブ配下の設定をオン、オフ指定。
ジョブ入力サブシステム 有効	[JES] タブ配下の設定をオン、オフ指定。
JES プログラム パス	実行ファイルの所在値。
システムカタログ	カタログファイルの所在地とファイル名称。
データセットの省略時ロケーション	JCL などで指定するファイルのデフォルト所在地。

一般】 XAリソース (0) 】 MSS... (✔) MQ...

メインフレーム サブシステム サポート有効: ☑ CICS (✔) JES... (✔) IMS... PL/I (✔) 一般 Initiators (1) Printers (0) ジョブ入カサブシステム有効: ☑ JESプログラムパス: \$JDEMO\bin\debug

システムカタログ: \$JDEMO\plijcl_base\catalog.dat

データセットの省略時ロケーション: \$JDEMO\plijcl_base

メインフレーム PL/I 開発: JCL Eclipse 編



7) [サーバー] > [プロパティ] > [MSS] > [JES] > [Initiators] タブでイニシエータ定義を確認します。 A ~ 9 までのク ラスに対するイニシエータが設定されています。

一般 Initiators (1)		Printers (0)			
A	Edit Initia	ator			
名前: INIT1					
Class:					
abcdefg	abcdefghijklmnopqrstuvwxyz0123456789				

8) [サーバー] > [プロパティ] > [MSS] > [PL/I] > [一般] タブで表示される画面の各項目を確認します。

項目名	説明		
PL/I 有効	[PL/I] タブ配下の設定をオン、オフ指定。		
Codewatch ソース パス	デバッグで使用するソースファイルの所在値。		
Codewatch STB パス	デバッグで使用するデバッグファイルの所在地。		
PL/I 構成ディレクトリ	プロジェクトの所在地。		
CICS (✔) JES (✔) IMS PL/I (✔) ► WO	k → JCLDEMO → bin → debug		
一般 名前	*		
PL/I有効: 🗹 🗃 fet	tched.adt		
Prompt for PLITEST attachment:	tched.def		
Codewatch ソース パス: 🚳 fet	tched.dll		
\$JDEMO de fei	tched.exp		
Reduntation of the state	tched.lib		
SJDEMO\bin\debug	ig fetched.obj		
📕 🔤 fet	- 🖉 fetched.obj.1.tlog		
PL/I構成デルクトリ: 🗿 fet	tched.pdb		
\$JDEMO	tched.stb		

9) 画面左上の [Home] をクリックして一覧画面に戻ります。



Home



3.7 Enterprise Server の開始と確認

- 1) サーバーエクスプローラ内に [PLIJCL] サーバーが表示されていることを確認します。表示されていない場合は [ローカル [localhost:86]] を右クリックし、[更新] を選択してリフレッシュしてください。
- 2) サーバーエクスプローラ内の [PLIJCL] サーバーを右クリックし、[プロジェクトに関連付ける] > [JCLDEMO] を選択します。 これにより Eclipse 内の [JCLDEMO] から実行される JCL は [PLIJCL] サーバーで処理されることになります。

プロジェクトに関連付ける・・	JCLDEMO

3) [PLIJCL] サーバーを右クリックして [開始] を選択します。



4) 下記ウィンドウが表示された場合は、ここではユーザーによる制限を行わないため [OK] ボタンをクリックします。

Enterprise Server サインオン		×
サーバーの接続詳細を	入力します:	
□サーバーを保護		
ユーザー名;		
パスワード:		
グループ:	デフォルト グループは空白	
✓ 資格情報の保存		
	OK キャンセル	

5) Enterprise Server Administration 画面へ移動して開始状態であることを確認後、[詳細] ボタンをクリックします。





6) [サーバー] > [診断] > [ES コンソール] で [PLIJCL] サーバーのコンソールログをリアルタイムにチェックすることができます。 また [Show Entire Log] をクリックしてログ全体を表示させることも可能です。

正常に開始されたことを確認します。

トレース	、 ダンプ
۵.	更新 ○ Show entries from 1 to 10 ● Show last 10 lines of 51 total entries
Entry	Event Show Entire Log
42	151102 11580971 12852 PLIJCL CASSI1600I SEP initialization completed successfully 11:58:09
43	151102 11580971 12296 PLIJCL CASSI1600I SEP initialization completed successfully 11:58:09
44	151102 11580993 12296 PLIJCL JES000042I SSTM not enabled: CICS 11:58:09
45	151102 11581015 12296 PLIJCL CASSI5001I PLTPI Phase 1 - No PLT Specified 11:58:09
46	151102 11581037 12296 PLIJCL CASSI5040I Active SEP memory strategy set to x'00000001', retain count 100 11:58:10
47	151102 11581140 12324 PLIJCL CASSI1744I MFPLI support loaded successfully 11:58:11
48	151102 11581140 12640 PLIJCL CASSI1744I MFPLI support loaded successfully 11:58:11
49	151102 11581140 12324 PLIJCL CASSI1600I SEP initialization completed successfully 11:58:11
50	151102 11581140 12840 PLIJCL CASBJ0005I Batch initiator started for job classes "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789" 11:58:11
51	151102 11581141 12324 PLUCL CASSI5021I PLTPL Phase 2 - No PLT Specified 11:58:11



3.8 JCL の実行

1) [PL/I エクスプローラー] 内に存在する [jcldemo.jcl] をダブルクリックして内容を表示します。IDCAM などのユーティリ ティを使用してファイルを操作したのち、JCLDEMO プログラムを実行していることがわかります。

//*************************************								
//* Run th	//* Run the JCLDEMO Program							
//*******	***************************************							
//STEP100	EXEC PGM=JCLDEMO							
//SYSOUT	DD SYSOUT=*,HOLD=Y							
//SYSPRINT	DD SYSOUT=*,HOLD=Y,DCB=(RECFM=LSEQ)							
//B1079256	<pre>DD DISP=(,CATLG),SPACE=(CYL,(5,5),RLSE),</pre>							
11	<pre>DCB=(RECFM=FBA,LRECL=137,BLKSIZE=0),</pre>							
11	DSN=SYSAD.STREAM.TEST							

2) [PL/I エクスプローラー] 内に存在する [jcldemo.pli] をダブルクリックして内容を表示します。「debugit」 定義を基に 判断しているデバッグ文を含んでコーディングされていることがわかります。

```
ON UNDEFINEDFILE (NOFILE)
BEGIN;
PUT SKIP LIST('ON UNIT UNDEFINED FILE - NOFILE Triggered');
GOTO AfterUndef;
END;
%if debugit %then
%DO;
CALL PLITEST('shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;br AfterUndef;c','',1);
START_DEBUG:
%END;
```

前項でプロジェクトプロパティのデバッグ定義をクリアしたので、このデバッグロジックには入りません。



3) [PL/I エクスプローラー] から [jcldemo.jcl] を選択して右クリック後、[Enterprise Server ヘサブミット] を選択する と、この JCL が実行されます。



4) [コンソール] タブに下記が表示されますので、リンクをクリックします。

■ フロパティー 🛙 🕄 問題 🔲 プロパティー				
Enterprise Server				
JCLCM0187I JOB01001 JCLDEMO JOB SUBMITTED (JOBNAME=JCLDEMO, JOBNUM=01001) 15:22:57				
JCLCM0180I JOB01001 JCLDEM0 Job ready for execution. 15:22:58				
Processed "C:\work\JCLDEMO\jcldemo.jcl"				

5) この JOB 番号にかかわるスプール一覧が表示されます。先頭の [JESYSMEG] をクリックしてジョブログを確認します。

JOB01001	Name	e: JCLDEMO		Status: Outpo	ut Hold	
Release	Class	s: A		Priority: 00		
Update	Use	T: JESUSER		COND: 00004		
JCLCM0188I JO ONCODE 99140: JCLCM0182I JO	B01001 JCLDEM The UNDEFINE B01001 JCLDEM	0 JOB STAR DFILE cond 0 JOB ENDE	TED 15:22 ition was D - COND	2:58 s raised because a DD statement was not used CODE 0004 15:22:59	in (FILE=NOFIL	E) 15:22:59
	Status	Class DI	O Name	Step	Nbr.	Proc Step
Details	Hold	А <u>л</u>	SYSMSG		0	
Details	Ready	A <u>s</u>	SPRINT	STEP00	1	
Details	Hold	A <u>sy</u>	<u>(SPRINT</u>	STEP1	2	
Details	Hold	A <u>s</u>	SPRINT	STEP2	3	
Details	Hold	A <u>sy</u>	<u>(SPRINT</u>	STEP3	4	

- 6) ステップ名 [STEP60] から [STEP090] でリターンコードに [0004] が返却されていることがわかります。
 --> 15:22:59 JCLCM0191I STEP ENDED STEP60 COND CODE 0004
- 7) [STEP60] で何が発生したのか確認するために、右クリックで [前へ戻る] を選択し、スプール一覧から [STEP60] の [SYSPRINT] をクリックします。

Details	Ready	А	SYSPRINT	STEP60
---------	-------	---	----------	--------

.....



8) 最終行にワーニングが発生しており、JCL で指定した 100 件のレコードを下回ったため発生した警告と判断できます。

JCLAM0194W(04) - Number of records read was less than COUNT(00000100).

```
//SYSIN DD *
REPRO INFILE(IN) OUTFILE(OUT) COUNT(100)
```

9) Jcldemo.jcl の STEP60 から STEP090 に記述されている「COUNT(100)」を「COUNT(5)」へ修正して保存する と、自動的にビルドがかかります。ビルド成功を確認後、JCL を再実行します。

```
//SYSIN DD *
    REPRO INFILE(IN) OUTFILE(OUT) COUNT(5)
/*
```

10) 前項同様の手順で [JESYSMSG] 内容を確認すると、全てのステップが正常に終了していることがわかります。

> 16:31:06	JCLCM01911	STEP ENDED	STEP60 -	COND CODE 0000	

11) STEP100 では jcldemo.pli ソースから出力された内容が参照できますので、ソースコードと合わせて確認してみてください。

Details	Hold	Α	TABLE	STEP100
---------	------	---	-------	---------

12) 画面左上の [Home] をクリックして一覧画面に戻ります。



Home

13) 実行された JCL から作成されたカタログ情報を確認します。Enterprise Server Administration 画面へ移動して



14) $[\forall - n -] > [\exists 2 + n - n] > [ES = \exists 2 + n - n - n]$ $\pi = b + n - n - n = b + n = b + n - n = b +$

サーバー	リスナー (3)) "	-ビス <mark>(4)</mark>	ハンドラ (4))バ
プロパティ		۲Ì	診断	過去の統計
ESモニター&	ביאם-אים			



15) 画面左の中央部にある [Resources] 直下のコンボボックスから [JES] を選択後、表示された [Catalog] ボタンをク リックします。前項で確認したスプールに関しても [Spool] ボタンをクリックすることにより、全てが参照可能になります。

Resources	_
JES 🗸 🗸	
Spool	
Catalog	
Control	
Locks	
Alias	

16) [List] ボタンをクリックして、カタログ情報の一覧を表示します。

	Data CATALOG	Refresh
List *	Catalo	ged Only

17) JCL の実行により作成されたカタログ情報が参照できます。

DS Org DS	Name	
VSAM	SYSAD.CLUSTER.AIX	DCB
VSAM	SYSAD.CLUSTER.BASE	DCB
VSAM	SYSAD.CLUSTER.BASE.DATA	DCB
VSAM	SYSAD.CLUSTER.BASE.INDEX	DCB
VSAM	SYSAD.CLUSTER.PATH	DCB
D PS	SYSAD.QSAM.TESTFILE	DCB
□ ?	SYSAD.STREAM.TEST	DCB
D PS	SYSAD.TABLE5	DCB
PS	SYSAD.TABLE6	DCB
D PS	SYSAD.VBFILE	DCB
PS	SYSAD.VBOUT	DCB
VSAM	SYSAD.VSAM.ESDS.TESTFILE	DCB
VSAM	SYSAD.VSAM.KSDS.TESTFILE	DCB
VSAM	SYSAD.VSAM.KSDS2.TESTFILE	DCB
VSAM	SYSAD.VSAM.RRDS.TESTFILE	DCB

18) 画面右端の [DCB] をクリックするとカタログされたファイルの情報が表示され、変更も可能です。

DS Name:	SYSAD.CLUSTER.A	IX 🗹 Catalog
Physical File:	C:¥WORK¥JCLDEMC	¥PLIJCL_BASE¥SYSAD.CLUSTER.BASE.DA"
DS Org:	VSAM 🗸	RECFM: KS 🗸
Codeset:	ASCII 🗸	Created: 2015/09/18 16:31:06.27
LRECL:	00080	Referenced: 2015/09/18 16:31:06.29
BLKSIZE:	00000	
VSAM Type:	Alternate Idx	Key Start/Len: 00009/00004
VSAM Attr:	Non-unique Key	Max / Avg: 00080 / 00013
ShareOptions:	Cross 2 🗸	Cross System: 3 🗸
Display	Start: 1 for	10000 Codeset: ASCII 🗸 🗌 Details



19) 画面中央の各 DSN をするとデータが参照可能です。

CATALOG Entry Content-Type: text/plain RECORDO1 AIX9 DATA-010101010101 RECORDO2 AIX4 DATA-020202020202 RECORDO3 AIX5 DATA-030303030303 RECORD04 AIX4 DATA-040404040404 RECORD05 AIX7 DATA-050505050505 RECORD06 AIX2 DATA-060606060606

20) また、この画面からカタログの作成や削除も可能です。

List	*	*					
	New	Details	Delete				

3.9 PL/I ソースのデバッグ

- 1) 前項でクリアした、プロジェクトのマクロプロパティを再指定します。PL/I エクスプローラーから [JCLDEMO] プロジェクトを右 クリックして [プロパティ] を選択し、プロパティウィンドウを開きます。
- [Micro Focus] > [プロジェクト設定] > [PL/I コンパイル設定] > [マクロ プロセッサ] を選択して [追加オプション] へ 「-define debugit」を入力後、[OK] ボタンをクリックしてください。自動的に再ビルドが実行され、この値によりソースコー ドに記述されたデバッグ判定が真になります。





3) 再度 jcldemo.jcl を実行すると Codewatch デバッガが自動的に立ち上がってきます。

<u>F</u> ile	<u>E</u> dit	View Actions Help						
⇔	0	P D Src: C:#work#jcldemo#jcldemopli	Env:	J	CLDEMO	¥	Find	
	116:	ON UNDEFINEDFILE (NOFILE)	^	IF	Owner			From
	117:	BEGIN:		۳.				riom
	118:	PUT SKIP LIST('ON UNIT UNDEFINED FILE - NOFILE Triggered');			Current			JCLDEMO¥125
	119:	GOTO AfterUndef;		1	JCLDEMO			lpi_main+0×33
	120:	END;		1	lpi main			0x374b7f0
	121:				0-2745760			
	122:	Xif debugit Xthen		IH	0.0740710			
	123:	XDO;	_					
_	124:	CALL PLITEST('shiib joidemo.dii;env JCLDEMO;br START_DEBUG;br AfterUndef;c	,11,1					
•		START_DEBUG:						
	126:	XEND;						
	407							

4) ステップインを行うことにより、ステップ実行が可能です。



5) ソースコード内の変数へマウスオーバーして右クリックし、 [View contents of '変数名'] を選択すると、値がポップアップウィンドウで参照できます。



6) 変数値を常に監視したい場合には、変数へマウスオーバーして右クリックし、 [Add '変数名' to Evaluation panel] を 選択すると、右側に表示されているパネルへ常に表示されるようになります。

ggle breakpoint able breakpoint

7) ステートメントの行をダブルクリックすることによりブレークポイントの設定が可能です。ブレークポイントには、該当行の左端に 赤丸が表示されます。解除も同様にダブルクリックを行います。

PRFNAMESO = DDP_STRUCB.PREF_NAME;

8) 何度か [Continue] アイコンをクリックして最後まで実行されると、デバッガが終了し、Codewatch が終了します。





9) ソースコードへ記述したマクロを有効にしてデバッグする方法のほかに、PLIDEBUG.DAT というファイルを使用して、ソースコ ードには何も記述せずにデバッグすることが可能です。このファイルを Micro Focus クラシック データファイル ツールを使用 して編集します。下記のようにツールを起動させます。

	SQL Option for DB2	×	퉬 データツール	\mathbf{F}	🐌 IBM DB2	١I
	データ接続	+	\mu 構成	ъ	🐌 Tablet PC	١
<u> </u>	HCO for DB2 LUW		Enterprise Developer for Eclipse		🔎 Adobe Reader XI	
r _e ,	HCO for SQL Server		Enterprise Developer for Visual Studio 2012	- 1	🐌 IBM WebSphere MQ 🛛 🔰	١
X	クラシック データファイル ツール		🔤 ドキュメント - Eclipse	- 1	Micro Focus License Manager	١
MF	データファイル ツール	1	<u>- ドナーバト Viewel Church</u> 2012		퉬 Micro Focus Enterprise Developer 👘	
_					·	

10) [ファイル] プロダウンメニューから [開く] を選択して、C:¥work¥JCLDEMO 直下にある [PLIDEBUG.DAT] ファイル を選択し、[開く] ボタンをクリックします。

X		Open		×	
ファイルの場所(1):	JCLDEMO	•	← 🗈 📸 ▼		
Ca	名前	^	更新日時	種類	
最近表示した場所	퉬 bin 퉬 plijcl_base		2015/09/18 10:57 2015/09/18 17:05	ファイル フォルダー ファイル フォルダー	
	plijcl_def		2015/09/18 9:27	ファイル フォルダー	
テスクトップ	pijcio4_der		2015/09/18 9:2/	ノデイフレ ノオフレター	
ライブラリ		т	2015/09/18 17:05	DAT ファイル DAT ファイル	
コンピューター					
ネットワーク	<			>	
	ファイル名(N):		•	開((<u>0</u>)	
	ファイルの種類(工):	_	キャンセル		
	最近使用したフォルダ	-			
	開((<u>A</u>)	自動	•		

11) 注意喚起を行うことなくファイルへ書き込まれる旨の確認ウィンドウが表示されますので、[OK] ボタンをクリックします。

	注記	? ×
注記 索引ファイルおよび相対 変引ファイルおよび相対 更新のために編	ファイルへの編集は直ちに反明 課したデータファイルを上書き(ありません。	ゃされます。 呆存する必要
🗆 このメッセージは今後表示しない(_))	<u> </u>



12) 空の内容が表示されますので、用意されているデータフォーマットを連結させます。[ファイル] プルダウンメニューから [データフ ァイル エディタ] > [レコードレイアウトのロード] を選択して、C:¥work¥JCLDEMO 直下にある [PLIDEBUG.str] ファ イルを選択し、[開く] ボタンをクリックします。

	JCLDEMO 💌
	名前
	🌗 bin
	\mu plijcl_base
📝 ファイル(F) 編集(E) 表示(V) ファイル(F)	\mu plijcl_def
レコードレイアウトのロード(L) データファイル エディタ(D)	\mu plijcl64_def
レコードレイアウトの関連付け(A)	PLIDEBUG.str

13) 右クリックを行い、[索引レコードの挿入] を選択します。レコードレイアウトの選択ウィンドウでは [PLIDEBUG_LAYOUT-DEFAULT] を選択して [OK] ボタンをクリックします。 索引レコードの挿入ウィンドウでは、そ のまま [OK] ボタンをクリックします。

	レコートドレイアウトの選択 ? ×
	レコードレイアウト レコードレイアウトの選択(B) レイアウト名 ハデドルグハイトを使用 8 PLIDEBUG_LAYOUT-DEFAULT 104
16進表示(<u>H</u>) Alt+F2 表示の同期(<u>S</u>) Ctrl+Y	選択したレコードルイアウトは新規のレコートを初期化するために使用されます。 条件値が必要なフィールドはすべて事前に設定されます。
貼り付け Shift+Ins	ОК + +ууди
索引レコードの挿入(X)Ctrl+I	→

14) 各項目へ値を入力します。

Micro Focus データファイル ツール - [PLIDEBUG.DAT (可変 長さ 索引)]							
アテイル(E) 編集(E) 表示(V) ファイル(E) 検索(S) オブション(Q) ツール(I) ウィンドウ(W) ヘルプ(H)							
JCLDEMO Y 1 shlib JCLDEMO.so;env.	PLIDEBUG_LAYOUT-DEFA	ULT	V1791 OK	<u>*</u>			
	7ィール・名	形式	値				
	I PLIDEBUG_LAYOUT 2 PGM_NAME	char(8)					
	2 FILLER1	char(12)	JOOEDEmo				
	2 DEBUG_ACTIVE	char(1)	Y				
	2 FILLER2	char(1)					
	2 DEBUG_IYPE	char(1)	1				
	2 DEBUG_INIT	char(1)	shlib JCLDEMO.so;env jcldemo;br %en	try;br %exit [det			
	1						
ያ PLIDEBUGDAT (可変 長さ 索引) 🗙 🛃 PLIDEBL	CDAT (可変 長さ 索引)	×					
(4) 出力							
準備OK	N/A 1/3-1*	長 104 (104 to	0104) フィールド2 / 8 行 0.列 0	OVR I			

MICRO FOCUS

項目名	值
PGM_NAME	JCLDEMO
DEBUG_ACTIVE	Y
DEBUG_TYPE	1
DEBUG_INIT	shlib JCLDEMO.dll;env jcldemo;br %entry;br %exit [det;q];c

入力した内容はソースコードに記述されている内容と同様になっています。

%if debugit %then
%DO;
CALL PLITEST('shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;br AfterUndef;c','',1);
START_DEBUG:
%END;

15) 前項同様に jcldemo.jcl を実行すると Codewatch が起動され、前項とは違いソースコードの先頭からデバッグされま す。

<u>F</u> ile	<u>E</u> dit	<u>V</u> ie	ew <u>A</u>	ctions	<u>H</u> elp							
\$	9	[+]	{ }	Src:	C:¥work¥jcldemo¥jcldemopli	~	Env:	JC	LDEMO 🗸	Find:		산
	1:	4	seesees * Coor	kakakakaka Aright (**************************************		^		Owner		From	
	3:	- 7	* A11	rights	reserved. */				Current		JOLDEMO¥%ENTRY	
	4:	- 7	aototototo	keledededede	alapalapalapalapalapalapalapalapalapala			3	JOLDEMO		lpi_main+0x33	
•	5:	J	CLDEM	D: PROC	JCLPARM) OPTIONS(MAIN);			2	lpi_main		0×374b7f0	
	6:							1	0×374b7f0			

3.10 終了処理

1) サーバーエクスプローラ内で [PLIJCL] を右クリックして [停止] を選択し、開始中のサーバーを停止します。

⊳		PLIJCL	
⊳	1	新規作成(N)	
⊳	2	停止	

2) [PLIJCL] サーバーの停止状態を確認後に、Eclipse を終了します。

WHAT'S NEXT

- メインフレーム PL/I 開発: CICS Eclipse 編
- 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。