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 4.0 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 インスタンスの設定
- 7. Enterprise Server インスタンス開始と確認
- 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

例題プログラムに関連するリソースを用意します。

- 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例題プログラム 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。



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。





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。

MICRD'	B
Enterprise Developer for Ecli	pse にようこそ
	ブテースト・ステップ ファースト・ステップ ファースト・ステップの開始
Web リンース Web 上の語的US	
Copen Team Developer Perspective Team Developer パースペウザークを招見ます。このパースペウザー・ ペインロームのリンースへのアクセスを開始化するためのシール CODOL たんだドルゴッジックションを開始化するためのシールを用	プロは、 や、 内して つわり、バースペクディン型語をする。このバースペクディン型は、00801、 アプリケーションを開発するためのクールを提供します。
UEF.	Open PL/I Perspective RLIパースペクティング間をます。このパースペクティンでは、RLTアッリ ケーショング開発するためのケールを発生します。

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−ト(N) 検索	プロジェクト(P)) 👬	集(E)	ナビゲート(N)	検索	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)				プロジョ プロジョ	ェクトを開く(E) 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) クリーーン(N)	Ctrl+B			すべて プロジン ワーキ:	ビルド(A) ロクトのビルド(B) ング・セットのビルド	:(W)	Ctrl+B
~	999-2(N) 自動的にビルド(M)		\rightarrow		クリーン 自動的	/(N) りにビルド(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。

MF 設定	— D X
7ብሥያ入力	指令の確定 🗘 🔹 🗢 🔹
 Micro Focus へ COBOL Enterprise Server ISPF Dialog JCL PL/I エディク 指令の確定 Xディスプレイ 	指令の確定の設定 フィイルのスキャン時に設定する指令を選択します。 フィイルのスキャン時に設定する指令を選択します。 フィイルには、プロンゴクト設定と異なる指令のみが設定されます。 □推夸の自動電度を実行] 手動による指令の確定は、以下で1個以上が選択されている場合にのみ許可されます。 -SQL ☑ EXEC SQL を含むファイルに SQL 指令を設定する
 > サービスインダーフェイス > データベース デバッグ デゾブレート ビルダー リモート JRE 検索 > 統合化トレース機能 	 ☑ EXEC SQL を含まないファイルに SQL 指令の設定を解除する ○ CLS ☑ EXEC CLSを含むファイルに CLCS 指令を設定する ☑ EXEC CLSを含まないファイルの CLCS 指令の設定を解除する マクロ ☑ マクロ文を含むファイルにマクロ指令を設定する ☑ マクロ文を含まないファイルのマクロ指令の設定を解除する
> Mylyn > Oomph > Remote Systems > Server > Terminal Validation > Web	デフォルトの復元(1) 進用(1)
? • •	OK キャンセル



3.3 メインフレーム PL/I プロジェクトのインポート

1) 用意した例題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。		Ľ	5
<u>S</u> elect an import wizard: フィルタスカ			
 ◇ General ③ アーカイブ・ファイル △ ファイル・システム ○ フォルダーまたはアーカイブ由来のプロジェクト ◇ 既存プロジェクトをワークスペースへ □ 設定 			~
(?) 次へ(N) > 終了(E))	キャンセ	JL

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。

MF 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。			
 ・ディレクトリーの選択(①: C¥work¥JCLDEMO アーカイブ・ファイルの選択(Δ): ブロジェクト(P): 	~	参照(<u>R</u>)
JCLDEMO(C:\#work\#JCLDEMO)	選	すべて選択 択をすべて創 更新(<u>E</u>	R(<u>S)</u> 解除(<u>D</u>))
オプション コネストしたプロジェクトを検索(<u>H</u>) コプロジェクトをワークスペースにコピー(<u>C</u>) ロワークスペースに既に存在するプロジェクトを隠す(i)			
ワーキング・セット ローキング・セットにプロジェクトを追加(① ワーキング・セット(<u>の</u>):	/	新規(<u>W</u>) 選択(<u>E</u>).	**
⑦ <戻3(B) 次△(N) > 終7(D)		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。



- Fetched.pli
- b A fetcher.pli
- jcldemo.pli
- ▷ 🚵 xmldemo.pli
 ▷ 🚇 PL/I インクルードファイル
- a 🙋 JCL ファイル
 - 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。

フィルタ入力	ビルド構成	(; • ; ; • •
 > リソース ▲ Micro Focus ビルドパス 	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。

■ ビルド構成の管理	×
ビルド構成の管理 現在使用中のビルド構成の違択やビルド構成の作成と削除をおこないます。	*
C₂ @ ✔ ✔ ¥	
構成の名前 x64	使用中
x86	•
0	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¥debu	ıg



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実行ファイルタイプとして生成することがわかります。

 ▲ Micro Focus ビルドパス ▲ ビルド構成 	x86 [使用中]	✓ 構成σ.
BMS ▶ PL/I コンパイル設定 ▶ PL/I リンパイル設定 ▶ PL/I リンク設定 ▶ アセンブラ コンパイラ アセンブラ リンカ	設定: フィルタテキストを入力 匡	
イベント ▲ プロジェクト設定	設定	値
BMS ▷ IMS ▲ PL/I コンパイル設定	プラットフォーム ターゲット 出力タイプ 出力ファイル名	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、例題の内容に沿って、 「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。確認後、[OK] ボタンをク リックします。

▲ Micro Focus ビルドパス ▲ ビルド構成 BMS	設定: 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
イベント	リストファイルを出力 (-1)	いいえ
▲ プロジェクト設定	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 コンパイル設定	追加オプション	
▷ アセンフラ コンパイラ	⊿ 高度	
アセンフラリンカ	$ Z_{ij}\mathbf{k} = \mathbf{k}_{ij} = 2^{ij} + -\vec{R} = f_{ij} \mathbf{k}_{ij} \mathbf{k}_{ij} \mathbf{k}_{ij}$	LM AR

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。
 - ✓ ⇒ bin
 ✓ ⇒ debug
 fetched.adt
 fetched.def
 fetched.dil
 fetched.den

 fetched.den

3.6 Enterprise Server インスタンスの設定

1) PL/I を実行するためのエンジンを搭載した Enterprise Server インスタンスを作成します。 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 🛛	Ē	V 🗎
▷ 🔜 ローカル [localhost:86]	新規作成(N)		•
	Administrat	ion ページを開く	Ctrl+F3

 C:¥work¥JCLDEMO には Enterprise Server インスタンス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):

サーバーデータの復旧元ディレクトリの選択: 「file:/// ~ <mark>【C:\work\JCLDEMO\plijcI_def</mark>

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 インスタンスが追加されます。



全変 重要 アプリケーション稼働ビット数 = Enterprise Server インスタンス稼働ビット数である必要があります。

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 [動的デバッグを許可] 欄のチェックをオンにします。これにより、Eclipse からのデバッグが行えます。

動的デバッグを許可: 🗹

6) 前項と同じタブの [構成情報] 欄を下記のように入力し、[適用] 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

😭 情報

CODEWATCH_NOTIF 環境変数について:

CodeWatch デバッガを使用する際に開発用インスタンスに指定します。本番インスタンス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。ここでは Eclipse 上のデバッガを使用するため削除します。

7) [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、オフ指定します。
ジョブ入力サブシステム 有効	[JES] タブ配下の設定をオン、オフ指定します。
JES プログラム パス	実行ファイルのパスを指定します。
システムカタログ	カタログファイルのパスとファイル名称を指定します。
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パスを指定します。

CICS (✔) JES (✔) IMS PL/I (✔)
一般 イニシエータ(1) プリンター(0)
ジョブ入力サブシステム有効: 🗹
JES プログラム バス: \$JDEMO\bin\debug
レ システムカタログ: 「SIDEMONNING」 base/sateleg dat
データセットの省略時ロケーション:
\$JDEMO\plijcl_base



8) [サーバー] > [プロパティ] > [MSS] > [JES] > [イニシエータ] 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 1	ニシエータ(1)	プリンター (0)
A	イニシエータの	編集
名前:		
INIT1	×	
クラス:		
abcdefghijk	Imnopqrstuvwx	yz0123456789
説明:		
キャンセル	OK 肖	刂除

9) [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、オフ指定します。		
CodeWatch ソース パス	CodeWatch デバッガで使用するソースファイルパスを指定します。		
CodeWatch STB パス	CodeWatch デバッガで使用するデバッグファイルパスを指定します。		
PL/I 構成ディレクトリ	プロジェクトのパスを指定します。		
CICS (✓) JES (✓) IMS PL/I (✓) → wor -般 PL/I 有効: ✓ Prompt for PLITEST attachment: □ Codewatch ソース パス: SJDEMO Codewatch STB パス: SJDEMO PL/I 構成ディレクトリ: SJDEMO fet SJDEMO fet SJDEMO fet SJDEMO fet	k > JCLDEMO > bin > debug cched.adt tched.def tched.dll tched.exp tched.lib tched.obj tched.obj tched.obj tched.stb		

10) 画面左上の [Home] をクリックして一覧画面に戻ります。





3.7 Enterprise Server インスタンスの開始と確認

- 1) サーバーエクスプローラ内に PLIJCL インスタンス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。
- サーバーエクスプローラ内の PLIJCL インスタンスを右クリックし、[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 プロジェクトから実行される JCL は PLIJCL インスタンス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 インスタンスを右クリックして [開始] を選択します。

۵ 🚪	PLIJCL
▷ 🚪	新規作成(N)
Þ 📩	開始

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。

WF	Enterprise Server サインオン				
サーバーの接続詳細をノ	く力します:				
ユーリー石・ 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 インスタンス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 【 ESコンソール 】 CSコンソール 】
曲值	● 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 12298 PLIJCL CASSI5001I PLTPI Phase 1 - No PLT Specified 11:58:09
46	151102 11581037 12298 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 12640 PLIJCL CASBJ0005I Batch initiator started for job classes "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789" 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。



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。CodeWatch デバッガを使用 する際に利用するデバッグ文が含まれていますが、ここでは Eclipse を使用したデバッグを行うため利用しません。





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。



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。

J0001000	Name	JOLDE	MO		Status: Complete
Hold	Class	: A			Priority: 00
Update	User	JESUSE	R		COND: 0004
Delete	File	STXRFC	IR/t00000027.	t	
JCLCM0188I J000	1000 JCLDEMO JC	B START	ED 13:33:54		
CASMG0001I MPL	IR05309E ONCODE	99140: T	he UNDEFINED	ILE condition was raised because a DD stat	ement was not used
JCLCM0182I J000	1000 JCLDEMO JC	B ENDER) - COND CODE	0004 133357	
	Status	Class	DD Name	Step	
Details	Hold	А	JESYSMSG		
Details	Ready	А	SYSPRINT	STEP00	
Details	Hold	А	SYSPRINT	STEP1	

- 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 JCLCM01911 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	Α	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。

× 1	10 01 00	T FOLIOMAN ANT	ATED EUDED	OTEDAA	OONE OODE	0000
>	тк-зт-нк.		STEP ENDED	STEPKIL -		
	10.01.00	JOLOWO ISTI	OTLI LINDLD		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 A	TABLE STEP100
----------------	---------------

12) Enterprise Server Administration 画面へ移動後、画面左上 [Home] をクリックして一覧画面に戻ります。



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。[詳細] ボタンをクリックします。



14) $[\forall - n -] > [\exists 2 n - n] > [ES \exists 2 n - k \exists 2 n - n - n]$ $\pi p > k - n - n]$

サーバー	リスナー (3)	サービス	(4) ハンドモ	ラ <mark>(4)</mark> パ
プロバティ	コントロール		ī)過去G	D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	*	Cataloged 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. AIX 🗹 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 V Details	



19) カタログ情報一覧画面に中央部に表示されている DS Name をクリックするとデータが参照可能です。

CATALOG Entry
Content-Type: text/plain

RECORD01	AIX9	DATA-010101010101
RECORD02	AIX4	DATA-020202020202
RECORD03	AIX5	DATA-03030303030303
RECORD04	AIX4	DATA-040404040404
RECORD05	AIX7	DATA-050505050505
RECORDO6	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	*	Cataloged Only	
	New	Details	Delete

3.9 PL/I ソースのデバッグ

JCL から実行される PL/I プログラムをデバッグします。

1) [実行] プルダウンメニューの [デバッグの構成] を選択します。

<u>У</u> -	-7	ナビゲート(N)	検索	プロジェクト(P)	実行(R)
R	実行	テ点をリセット			
Q	実行	₸(R)		Ctr	1+F11
16	デバ	「ッグ(D)			F11
	実行	亍履歴(T)			•
	実行	₸(S)			•
	実行	亍構成(N)			
	デバ	「ッグ履歴(H)			+
	デバ	「ッグ(G)			•
	デバ	、ッグの構成(B)			

2) 左側のツリービューから [PL/I Enterprise Server] を選択して、左上の [新規の起動構成] アイコンをクリックします。



3) [PL/I プロジェクト] へ対象となる JCLDEMO プロジェクトを入力し、[Enterprise Server] へ実行させる PLIJCL イ ンスタンスを指定します。[デバッグの種類] は「JCL」タブを選択した状態で、[デバッグ] ボタンをクリックします。



病の作成、管理、および実行 Enterprise Server アブリケーションへの接続	テ きとデバッグ			Ś
	名前(U): 新規構成(4) パー般 モッソース 日 共通(C) Enterprise Server 上でデバッグセッシ ・ PL/1 プロジェクト(P) ICLDEMO) 「シーデバッグシンボル」 rョンを開始して、PL/I ブ	ログラムの起動を待機しま	す。 参照
Ju JUnit Ju JUnit Ju JUnit Ju JUnit Ju JUnit Odejs Application ● Odej Ju-J0-7-9 ■ QU JU-J0-79 ■ PU/J70-532 ■ PU/J70-532 ■ PU/J70-532 ■ PU/J07542 ■	・ Enterprise Server 接版 localhost ・ デパッグの確認 ・ デパッグの確認 ・ CCE / ALL MS Web サービ・ ・ CLE 設定 (空白の場合はす/ ジョブ名: ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	サーバー: PLIJCL ス) Java] べての JCL ジョブをデバッ	Í)	参照
Khino JavaScript Standalone V/8 V/M A V// マン			前回保管した状態に	戻す(⊻) 適用(⊻)

4) デバッグタブで [アタッチ待機] 状態になったことを確認します。



- 5) PL/I エクスプローラー内の jcldemo.jcl を右クリックして [Enterprise Serverへのサブミット] を選択し、 JCL を実行 します。
- 6) 再度、パースペクティブの切り替え確認ウィンドウが表示されますので、[はい] ボタンをクリックし、デバッグ用のパースペクティ ブを開きます。



7) 少し待つとデバッグセッションが開始して、プログラムのステップ実行が可能になります。[F5] キーもしくは [実行] プルダウン メニューから [ステップイン] を選択してステップを進めることができ、 [式] タブでは使用している変数の値が確認できます。



work - $\vec{\tau}/(\tau \vec{\tau}$ - JCLDEMO/jcldemo.pli - Eclipse			-		×
ファイル(E) 編集(E) ソース リファクタリング ナビゲート(N) 検索 プロジェクト(E) 実行(E) ウィンドウ(M) ヘルプ(H)					
	III 174	🖉 🕶 👷 🗢 🎧 🕶	や ◆ ・ ☆ ・ 24ック・アクセス ■ ■ 品 ■	ල ල ල	
巻デバッグ ☆ 巻 Servers 後 夢 マ 中 ロ	いー変数 💁 ブレークポイント 😚 式 😂		🖄 🐗 🖻 🍦 💥 🍇 🤻		8
 ● 読 転荷構成(4) [PU/I Enterprise Server]	名前 > 約" 'jclparm' > 約" 'iNREC"		it Parm	Í	• 🐌
■ JCLDEMO (行: 162)	IN BASE KEY	RECORD	201		
_lpi_main	IN_FILL1				
system/MVS	IN_AIX_KEY	AD(9			
	IN_FILL2				v .
	<			>	
🖹 jeldemo,pli 🕄 📕 サーバー: localhost 💿 jeldemo.jel			א גאפאלי 😰 😵 🖓 ד		
🔓 jcldemo.pli			 vbfile_buff char (400) varying 	-	*
·····1·····2·····3····4·····5·····6·····7·····8··			B1079256 FILE PRINT		
 OPEN FILE(BASEDAT) SEQUENTIAL; OPEN FILE(PATHDAT) SEQUENTIAL; 		^	 Debug_commands char (1024) Display_Address char (100) vary PLITEST_Flags fixed bin (31) init < 層合>	varying ing init ! (3)	
<pre>e /***********************************</pre>		-	 → 二二 → <置名> → <置名> → <置名> 		
<pre>put skip list('Reading via BASE');</pre>			 <置名> <要名> 		
READ FILE(BASEDAT) INTO(INREC);			 <置名> 		
DO INTLE(NORE_BASE): OUT_BASE_[~	 <置名> <置名> <置名> <置名> 		Ļ
¢		> <	-	>	
🖸 🕹 🖉 🖉 😓 😓 🖘			🗙 🗟 📓 🖻 🛃 🖬 - 📬 -		
Enterprise Server					
Processed "C:\work\JCLDEMO\jcldemo.jcl"					2

- 8) 希望のステップの左端をダブルクリックすることにより、ブレークポイントを設定することも可能です。
 - ▶ PL/I行ブレークポイント
 PRE_BASE);
- 9) 先に進める場合は画面上部の再開アイコンをクリックします。



10) デバッグを終了させるため、画面上部の終了アイコンをクリックします。



11) デバッガが停止状態になったのを確認後、右クリックして [終了したエントリをすべて削除] を選択し、これを削除します。



12) PL/I パースペクティブへ戻るには画面右上の PL/I アイコンをクリックします。





3.10 終了処理

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

⊳		PLIJCL
\triangleright	1	新規作成(N)
⊳	2	停止

2) PLIJCL インスタンス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。