顧客のデータベースへのコンパイル済みコードの追加

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

顧客のデータベースへのコンパイル済みコードの追加

InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

顧客のデータベースへのコンパイル済みコードの追加

このページでは、コンパイル済みのコードを顧客のデータベースに追加する方法について説明します。これにより、リコンパイルする必要のない新しいコードを顧客に提供できるようになります。

重要　この手順は、実働環境で使用する前に、特定のクラスを使用してテスト環境でテストしておくことを強くお勧めします。

1 要件

要件は以下のとおりです。

- InterSystems IRIS® のバージョンは、コードを作成してコンパイルするシステムと、そのコードをインストールするシステムで同じにする必要があります。
- SQL 区切り識別子の設定は、両方のシステムで同じ設定にする必要があります。

2 コンパイル済みコードの配置

コンパイル済みコードを配置するには、以下の操作を実行します。

1. プロジェクトを作成して、クラス定義、ルーチン、およびグローバルを含めます。そのためには、%Studio.Project クラスのメソッドを使用します。
2. プロジェクトに含まれるすべてのコードがコンパイルされていることを確認します。
3. %Studio.Project のメソッドを使用して、プロジェクトへの OREF を取得します。
4. プロジェクトの DeployToFile() インスタンス・メソッドを呼び出します。以下に例を示します。

   ObjectScript

   ```
   set sc=projectoref.DeployToFile("c:\test\myglobal.xml",,1)
   ```

   3 番目の引数に 1 を指定しているため、生成されたファイルにはソース・コードと中間コードは含まれません。このメソッドのシグニチャと詳細は、以下のとおりです。

   ```
   method DeployToFile(file As %String,
                       qspec As %String,
                       removesource As %Boolean = 0) as %Status
   ```

   このプロジェクトに属しているパッケージ化済みのコードを含むファイルを生成します。file はファイルの名前です。qspec は標準コンパイルの修飾子を格納する文字列です。詳細は、$SYSTEM のリファレンスで “フラグおよび識別子” を参照してください。/exportselectivity 修飾子が目的に応じて指定されていることを特に確認する必要があります。また、生成されたルーチンのソース・コードを保持するかどうかを制御する k フラグも指定できます。

removesource は、ブーリアン値です。removesource が 1 の場合は、ルーチンとメソッドのソースおよび中間コードがグローバルに含まれなくなります (k フラグの設定は影響しません)。

配置モードの詳細は、"クラスの定義と使用" の "クラスを配置モードに入れる方法" を参照してください。

5. 顧客には、このファイルを指示と共に提供します。顧客はターミナルを使用して、適切なネームスペースに切り替え、以下に示すように %Studio.Project の InstallFromFile() メソッドを呼び出す必要があります。

```
set sc=##class(%Studio.Project).InstallFromFile("c:\test\myglobal.xml")
```

DeployToFile() は、このクラス内の関係によって参照されるすべての親クラスと子クラスを、導入するアイテムのリストに自動的に追加します。

3 DeployToFile() と InstallFromFile() の制限事項

DeployToFile() メソッドと InstallFromFile() メソッドには、以下の制限があります。

- DeployToFile() および InstallFromFile() メソッドは、クラス・プロジェクション (Java ファイルに投影するクラスなど) を処理しません。
- これらのメソッドは、依存関係のチェックを一切実行しないため、サブクラスが自動的に含まれることはありません。すべての必要なクラスをパッケージに挿入する責任は開発者にあります。

  例えば、InstallFromFile() は、コードのインストール先になる場所をチェックしません。また、スーパークラスがインストールされていないシステムに、サブクラスを導入してしまう可能性もあります。

重要　この手順は、実働環境で使用する前に、特定のクラスを使用してテスト環境でテストしておくことを強くお勧めします。

4 例

以下は、簡単なルーチンの例を示しています。このルーチンは、プロジェクトを作成しますが、保存はしません。プロジェクトを保存していなくても、DeployToFile() を呼び出すことができます。

```
 ; deployexample
 set p=##class(%Studio.Project).%New()
 do p.AddItem("Sample.Customer.cls")
 do p.AddItem("Sample.Person.cls")

 do p.DeployToFile("c:\test\myglobal.xml",,1)
```

この簡単なルーチンは、デモンストレーションを目的としているため、エラー・チェックが含まれていません。

5 実行中のプロセスに対する影響

このセクションでは、現在コードを実行中のシステムに新しいコンパイル済みバージョンのコードをロードした場合、実行中のプロセスに与える影響について説明します。

- 現在実行中のルーチンとクラス・メソッドは、古いコードを継続して使用します。コンパイルの完了後に、このルーチンまたはクラス・メソッドが同じプロセスから再度呼び出されると、新しいコードをロードするようになります。

- ルーチンまたはクラス・メソッドが別のルーチンまたはクラス・メソッドを呼び出してから呼び出し元に戻った場合、プロセスは呼び出しコードの古いバージョンを継続して使用します。例えば、ルーチン A がルーチン B を呼び出した後で、ルーチン A に制御が戻されたとします。ルーチン A の実行中 (またはルーチン B の実行中) にルーチン A をリコンパイルすると、ルーチン A に制御が戻されたときに、プロセスはルーチン A の古いコードを継続して使用します。
- 開いているプロセスは、メモリ内バージョンの OREF を継続して使用します。
- インスタンス・メソッドはオブジェクトのインスタンスに依存します。オブジェクトがインスタンス化されるときには、どのオブジェクトも既存のインスタンス・メソッドのバージョンを使用します。オブジェクトは、OREF が破棄されて、新しいオブジェクトがインスタンス化されるまで、そのコードを継続して使用します。そのため、オブジェクトをインスタンス化してから新しいバージョンのクラスをインポートしても (またはクラスを完全に削除しても)、そのインスタンスのメソッドを継続して呼び出すことができます。つまり、そのオブジェクトがインスタンス化されたときに存在していたクラスのバージョンを使用することになります。

OREF を照会することで、インスタンスが最新バージョンのクラス・コードを使用しているかどうかを確認できます。

ObjectScript

```
Write oref.%ClassIsLatestVersion()
```

%ClassIsLatestVersion() メソッドは、インスタンスが最新バージョンのクラス・コードを使用している場合に 1 を返します。それ以外の場合は、0 を返します。

上記の説明は、特定のインスタンスのコードをリコンパイルしたときにも当てはまります。

注釈　ここに示した説明は、自分がクラスをコンパイルしているインスタンスに適用されます。それとは別の ECP 経由 (ミラー メンバ) のインスタンスの場合、ルーチンまたはクラスは一時的に矛盾した状態になってエラーをスローします。