記事
Mihoko Iijima · 2020年7月6日 18m read

%InstallerでInterSystems Cachéにアプリケーションをデプロイする

InterSystemsのテクノロジースタックを使用して独自のアプリを開発し、顧客側で複数のデプロイを実行したいとします。開発プロセスでは、クラスをインポートするだけでなく、必要に応じて環境を微調整する必要があるため、アプリケーションの詳細なインストールガイドを作成しました。この特定のタスクに対処するために、インターシステムズは、 %Installer（Caché/Ensemble） という特別なツールを作成しました。続きを読んでその使用方法を学んでください。

%Installer

このツールを使用すると、インストール手順ではなく、目的のCaché構成を記述するインストールマニフェストを定義できます。作成したい Caché 構成を記述します。必要な内容を記述するだけで、環境を変更するために必要なコードが自動的に生成されます。
したがって、マニフェストのみを配布する必要がありますが、インストールコードはすべてコンパイル時に特定の Caché サーバ用に生成されます。

マニフェストを定義するには、ターゲット構成の詳細な説明を含む新しいXDataブロックを作成してから、このXDataブロックのCaché ObjectScriptコードを生成するメソッドを実装します（このコードは常に同じです）。
マニフェストの準備ができたら、ターミナルまたはCaché ObjectScriptコードから、またはCachéのインストール中に自動的にアクセスできます。 マニフェストは%SYSネームスペースで実行する必要があります。 マニフェストは、システムパラメータ（スーパーサーバポート、OS、mgrディレクトリなど）とユーザが提供した任意のパラメータの両方を処理できます。
一般に、各インストールクラスは次の要件を満たしている必要があります。

- %occInclude.incへのリンクを含むこと
- Cachéサーバ構成のXDataブロックを含むこと
- ブロックには任意の有効な名前を付けること
- スタジオでブロックの名前の後に[XMLNamespace = INSTALLER]を追加すること
- ルートアイテム（1つのみである必要があります）は<Manifest>と呼ばれ、他のすべてのアイテムを含むように記述すること
- また、XDataブロックに必要なプログラムコードを生成するsetup()メソッドを実装すること

インストーラの実行

複数の 方法でインストールマニフェストを実行できます。

- ターミナルを使用して%SYSネームスペースで、またはCachéObjectScriptコードから以下実行します。

```
do ##class(MyPackage.MyInstaller).setup()
```

- Cachéのインストール中に自動的に実行されます。 これを行うには、インストーラのクラスを、Caché インストールパッケージ（すなわち、setup_cache.exeまたはcinstallが格納されている場所）のフォルダ内に格納されているDefaultInstallerClass.xmlにエクスポートする必要があります。 このクラスは、Cachéのインストール中に%SYSネームスペースにインポートされ、 setup() メソッドを介して実行されます。

例:

簡単な例を考えてみましょう。

ユーザー定義の名前で新しいネームスペースを作成、デフォルトのWebアプリを作成、作成されたネームスペースにコードをインポートするインストーラーを含む **App.Installer**というクラスを作成します。

```
 /// You can see generated method in zsetup+1^App.Installer.1
XData Install [ XMLNamespace = INSTALLER ]
{
<Manifest>
   <If Condition='(##class(Config.Namespaces).Exists("${Namespace}")=0)'>
      <Log Text="Creating namespace ${Namespace}" Level="0"/>
      <Namespace Name="${Namespace}" Create="yes" Code="${Namespace}" Ensemble="0"
Data="${Namespace}">
         <Configuration>
            <Database Name="${Namespace}" Dir="${MGRDIR}${Namespace}" Create="yes"/>
         </Configuration>
      </Namespace>
      <Log Text="End Creating namespace ${Namespace}" Level="0"/>
   </If>


   <Role Name="AppRole" Description="Role to access and use the App"
Resources="%DB_CACHESYS:RW,%Admin_Secure:U" />

   <Namespace Name="${Namespace}" Create="no">
      <CSPApplication Url="/csp/${Namespace}" Directory="${CSPDIR}${Namespace}"
AuthenticationMethods="64" IsNamespaceDefault="true" Grant="AppRole" />
      <IfDef Var="SourceDir">
         <Log Text="SourceDir defined - offline install from ${SourceDir}" Level="0"/>
         <Import File="${SourceDir}"/>
      </IfDef>
   </Namespace>
</Manifest>
}

///Entry point method, you need to call
/// At class compile time it generate Caché ObjectScript code from the manifest
/// After that you can run this installer from a terminal:
/// Set pVars("Namespace")="NewNamespace"
/// Set pVars("SourceDir")="C: \ temp \ distr \ "
/// Do ##class(App.Installer).setup(.pVars)
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 0, pInstaller As %Installer.Installer) As %Status [
CodeMode = objectgenerator, Internal ]
{
    Quit ##class(%Installer.Manifest).%Generate(%compiledclass, %code, "Install")
}
}
```

この例では、インストーラーは次のアクションを実行します。

- Namespace変数の値と同じ名前のネームスペースが存在するかどうかを確認します（明確にするために、Namespace変数がNewNamespaceに設定されていることを指定しましょう）
- そうでなければ、NewNamespaceという新しいネームスペースが作成されることをログに記録します。
- 新しいネームスペースを定義します。
    - 名前はNewNamespaceです
    - 新しいネームスペースを作成します
    - ルーチンデータベースはNewNamespaceです
    - Ensembleを有効にしないでください

    - ルーチンデータベースはNewNamespaceです
- 新しいデータベースを作成します
    - 名前はNewNamespaceです;
    - それを mgr/NewNamespaceフォルダーに作成します（MGRDIR変数はデフォルトで使用可能であることに 注 意してください）
- ネームスペースの作成を完了し、ログに記録されます。
- 新しいロールを次のように作成します。AppRole（リソース %DB_CACHESYS:RW および %Admin_Secure:U を設定）
- デフォルトのWebアプリケーション /csp/NewNamespace が作成されます（Grant 属性なり、AppRole が設定されます）
- SourceDir変数が定義されている場合、そこからすべてのファイルをNewNamespaceにインポートします。

このインストーラをターミナルで実行するには、次のコマンドを実行します。

```
Set pVars("Namespace")="NewNamespace"
Set pVars("SourceDir")="C:\temp\distr\"
Do ##class(App.Installer).setup(.pVars)
```

実行中、ターミナルは次の関連情報を表示します。

```
2016-02-17 19:26:17 0 App.Installer: Installation starting at 2016-02-17 19:26:17, LogLevel=0
2016-02-17 19:26:17 0 : Creating namespace NewNamespace
2016-02-17 19:26:17 0 : End Creating namespace NewNamespace
2016-02-17 19:26:17 0 : SourceDir defined - offline install from C: \ temp \ distr \
2016-02-17 19:26:18 0 App.Installer: Installation succeeded at 2016-02-17 19:26:18
2016-02-17 19:26:18 0 %Installer: Elapsed time .545148s
```

発生していることについてもっとの情報を取得するためには、LogLevelを指定します（0（デフォルト）から3（未加工）、数値が大きければ大きいほどより詳細な情報が得られます）。

```
Do ##class(App.Installer).setup(.pVars, 3)
```

次に、インストーラマニフェストで実行できることについて説明します。

利用可能なノード

マニフェストは次のアイテムで構成されています。

ノード	親ノード	属性（デフォルト値）	説明
Arg	Invoke、Error	Value – 引数の値	InvokeまたはError経由で呼び出されるたメソッドに引数を渡します
ClassMapping	Configuration	Package – マッピングされるパッケージ From – マッピングに使用されるソースデータベースの名前	データベースからこの構成項目を含むネームスペースへのクラスマッピングを作成します。

Compile	Namespace	Class – コンパイルするクラスの名 Flags – コンパイルフラグ(ck) IgnoreErrors – エラーを無視(0)	コンパイラクラス。$System.OBJ.Compile(Class, Flags)を呼び出す
Configuration	Namespace		ネームスペースにデータベースの作成使用されます。タグを閉じるとマッピングがアクティブになり、CPFファイルを更新します。
CopyClass	Namespace	Src — ソースクラス Target — ターゲットクラス Replace —ソースクラスを削除(0)	ソースクラス定義をターゲット定義にコピーまたは移動します。
CopyDir	Manifest	Src —ソースディレクトリ Target —ターゲットディレクトリ IgnoreErrors —エラーを無視(0)	ディレクトリをコピーします。
CopyFile	Manifest	Src —ソースファイル Target —ターゲットファイル IgnoreErrors —エラーを無視(0)	ファイルをコピーします。
Credential	Production	Name – アクセス認証情報の名前 Username – ユーザー名 Password – ユーザーパスワード Overwrite – アカウントが既に存在する場合は上書き	アクセス資格情報を作成または上書きします。
CSPApplication	Namespace	AuthenticationMethods – 有効な認証方法 AutoCompile – 自動コンパイル(CSP設定) CSPZENEnabled – CSP / ZENフラグ ChangePasswordPage – パスワードページを変更するパス CookiePath – セッションCookieパス CustomErrorPage – カスタムエラーページへのパス DefaultSuperclass – デフォルトスーパークラス DefaultTimeout – セッションタイムアウト Description – 説明[文字列の折り返しの区切り]Directory – CSPファイルへのパス EventClass – イベントクラ	Webアプリケーションを作成または変更します。詳細については、ドキュメントおよびSecurity.Applicationsクラスを参照してください

		ス名 Grant – システムへのログイン時に割り当てられたロールのリスト GroupById – Idプロパティによるグループ InboundWebServicesEnabled – 受信Webサービスフラグ IsNamespaceDefault – 名前空間デフォルトアプリケーションフラグ LockCSPName – CSP名のロックフラグ LoginClass – ログインページへのパス PackageName – パッケージ名プロパティ PermittedClasses – 許可されたクラスプロパティ Recurse – 再帰フラグ(サブディレクトリを提供) (0) Resource – Webアプリにアクセスするために必要なリソース ServeFiles – サービスファイルプロパティ ServeFilesTimeout – 静的ファイルをキャッシュする時間(秒単位)。 TwoFactorEnabled – 2段階認証 Url – Webアプリケーションの名前 UseSessionCookie – セッションにCookieを使用する	
Database	Configuration	BlockSize – データベースのバイト単位のブロックサイズ ClusterMountMode – データベースをクラスターの一部としてマウント Collation – 並べ順序 Create – 新しいデータベースを作成するかどうか(yes / no / overwrite)(はい) Dir – ディレクトリ Encrypted – 暗号化データベース EncryptionKeyID – 暗号化キーのID ExpansionSize – MB単位のサイズ InitialSize – 初期サイズ MaximumSize – 最大サイズ MountAtStartup – 起動時にマウント MountRequired – 起動時に	データベースを作成または変更します。詳細については、 ドキュメント 、 Config.Databases および SYS.Database クラスを参照してください

		データベースを正常にマウントする必要があることを指定します。 Name – データベース名 PublicPermissions – パブリック権限 Resource – リソース StreamLocation – このデータベースに関連付けられたストリームが移動するディレクトリ	
Default	Manifest	Name – 変数名 Value – 変数値 Dir – 変数値（これがフォルダ/ファイルへのパスの場合）	変数値を設定します（未設定の場合）
Else	Manifest, Namespace	If文がfalseのときに実行されます。	
Error	Manifest	Status – エラーコード Source – エラーのソース	例外をスローします。${}および#{}構文は使用できないことに 注 意してください。
ForEach	Manifest	Index – 変数名Values – 変数の値のリスト	コレクション制御 ループ
GlobalMapping	Configuration	Global – グローバル名 From – マッピングするデータベースの名前 Collation – 並べの順序 (Caché 標準)	グローバルをマッピングします。
If	Manifest, Namespace	Condition – 条件付きステートメント	条件付きif文
IfDef	Manifest, Namespace	Var – 変数名	変数がすでに設定されている場合に使用される条件付きif文
IfNotDef	Manifest, Namespace	Var – 変数名	変数がすでに設定されていない場合に使用される条件付きif文
Import	Namespace	File – インポート用のファイル/フォルダー Flags — コンパイルフラグ (ck) IgnoreErrors —エラーを無視 (0) Recurse – 再帰的にインポート (0)	ファイルをインポートします。次を呼び出します。[文字列の折り返しの区切り] $System.OBJ.ImportDir(File,,Flags,,Recurse) and $System.OBJ.Load(File,Flags)
Invoke	Namespace	Class – クラス名	さまざまな引数を指定して

		Method – メソッド名 CheckStatus – 返されたステータスを確認 Return – 結果を変数に書き込む	クラスのメソッドの呼び出し、実行結果を返します
LoadPage	Namespace	Name – CSPページへのパス Dir – CSPページのあるフォルダ- Flags — コンパイルフラグ (ck) IgnoreErrors — エラーを無視 (0)	$System.CSP.LoadPage(Name, Flags)と$System.CSP.LoadPageDir(Dir, Flags)を使ってCSP fileを読み込みます。
Log	Manifest	Level – 0 (最小) から3 (詳細) までのロギングレベル テキスト – 長さ32,000文字までの文字列	ロギングレベルが「level」属性の場合、ログにメッセージを追加します
Manifest			ルートアイテム。マニフェストで唯一のルートアイテム。他のすべてのアイテムを含みます。
Namespace	Manifest	Name – ネームスペースの名前 Create – 新しい名前空間を作成するかどうか (はい/いいえ/上書き (はい)) Code – プログラムコードのデータベース Data – データベース Ensemble – ネームスペースに対してEnsembleを有効化他のすべての属性Ensemble Webアプリケーションに適用可能	インストラクションを定義します。
Production	Namespace	Name – プロダクション名[AutoStart – プロダクションの自動起動	Ensembleプロダクションを構成します。
Resource	Manifest	Name – リソース名 Description – リソースの説明 Permission – パブリック権限	リソースを作成または変更します。
Role	Manifest	Name – 役割の名前 Description – 役割の説明 (カンマを含めることはできません) Resources – 役割に与えられるリソース。「MyResource:RW,MyResource1:RWU」として表されます RolesGranted – 対応するロ	新しいロールを作成します。

		ールを付与するかどうか	
RoutineMapping	Configuration	Routines – ルーチン名 Type – ルーチンタイプ(MAC、INT、INC、OBJまたはALL) From – ソースデータベース	ルーチンの新しいマッピングを作成します。
Setting	Production	Item – 構成な項目[文字列の折り返しの区切り]Target – パラメータタイプ:項目、ホスト、アダプタ[文字列の折り返しの区切り]Setting – パラメータ名[文字列の折り返しの区切り]Value – パラメータ値	Ensembleプロダクションの項目を構成します Ens.Production:ApplySettings メソッドを呼び出します
SystemSetting	Manifest	Name – 構成パッケージのclass.property Value – 属性	Configパッケージの属性を設定します(Modifyメソッドを使用)。
User	Manifest	Username – ユーザ名 PasswordVar – パスワードを含む変数 Roles – ユーザの役割のリスト Fullname – フルネーム Namespace – 開始ネームスペース Routine – 開始ルーチン ExpirationDate – ユーザが非アクティブになる日付 ChangePassword – システムに次回ログインしたときにパスワードを変更する Enabled – ユーザがアクティブかどうか	ユーザを作成または変更します。
Var	Manifest	Name – 変数名 Value – 変数値	変数に値を割り当てます。

変数

ユーザ提供の変数

属性 っては、マニフェストの実行時に展開される式(文字列)を含めるこができます。
次のように、展開できる式には3つのタイプがあります。

- ${<Variable_name>} – マニフェストの実行時に計算される変数(ユーザ定義または環境変数;間接参照)の値。
- ${#<Parameter_name>} – コンパイル時にインストラクタのクラスから指定されたパラメータの値に置き換えられます。
- #{<Cache_ObjectScript_code>} —指定されたCaché

ObjectScript文の値は、マニフェストの実行中に処理されます。必要に応じて引用符を付けてください。

パラメータ値はコンパイル時に定義されるため、変数またはCaché ObjectScript文の一部にすることができます。
変数はCaché ObjectScriptコードの前に解釈されるため、Caché ObjectScript文で使用できます。
例：#{$ZCVT("${NAMESPACE}","L")}

システム変数

次の変数は常に使用可能です。

変数	説明	サンプル値
SourceDir	(インストーラの実行時にのみ使用可能) インストール(setup_cache.exeまたはcinstall)が実行されるディレクトリ。	/InterSystems/distr/
ISCUpgrade	(インストーラの実行時にのみ使用可能) 新規インストールなのか、アップグレードなのかを示します。この変数は、0(新規インストール)または1(アップグレード)のいずれかです。	0 (インストール) 1 (アップグレード)
CFGDIR	INSTALLDIRを参照。	/InterSystems/Cache/
CFGFILE	CPFファイルへのパス	/InterSystems/Cache/cache.cpf
CFGNAME	インスタンスの名前	CACHE
CPUCOUNT	CPUコアの数	4
CSPDIR	CSPディレクトリ	/InterSystems/Cache/csp/
HOSTNAME	Webサーバの名前	SCHOOL15
HTTPPORT	Webサーバのポート	80
INSTALLDIR	Cachéのインストールディレクトリ	/InterSystems/Cache/
MGRDIR	マネージャディレクトリ(mgr)	/InterSystems/Cache/mgr/
PLATFORM	オペレーティングシステム	UNIX
PORT	Cachéスーパーサーバポート	1972
PROCESSOR	プラットフォームの名前	x86-64
VERSION	Cachéのバージョン	2015.1.1

デバッグ

ノード属性値としてどのような値が割り当てることができるのか理解に苦しむことがあります。これを把握するには、setupメソッドの生成されたintコードを確認してください。

ほとんどの場合、メインコールは %Installer.Installer クラスのオブジェクトである

tInstaller<ElementName>に対して行われ、次に、システムメソッドを直接呼び出します。
または、ノード属性またはプロパティである
%Installer.<ElementName>クラスのコードを確認することができます。プログラムコードは、
%OnBeforeGenerateCode、%OnGenerateCode、%OnAfterGenerateCode メソッドで生成されます。デバッグ
のため、インストーラへの呼び出しをトランザクションにラップすることをお勧めします。たとえば、
Caché内で行われたすべての変更を簡単に元に戻すため
に TSTART/TROLLBACK
コマンドを使うことができます（ただし、新しいデータベースファイルの作成など、外部の変更は元に戻りません
）。

最後に、LogLevel を 3 に設定することを忘れないでください。

例:

MDX2JSONプロジェクトは インストーラを提供します 。
プロジェクトをインストールするには、MDX2JSON.Installer
クラスを含む installer.xml
ファイルを任意のネームスペースにインポートします。管理ポータルから、またはファイルをスタジオにドラッグ
＆ドロップすることによってインポートを実行できます。
次に、ターミナルで次のコマンドを実行します。

```
do ##class(MDX2JSON.Installer).setup()
```

その結果、CachéはGitHubリポジトリからアプリケーションファイルをロードし、デフォルトの MDX2JSON
ネームスペース /MDX2JSON データベースにインストールを実行し、MDX2SJONパッケージを
%All(SAMPLESにマップし、MDX2SJONグローバルを %Allにマップし、SAMPLESネームスペースは /MDX2JS
ON で定義されるRESTアプリケーションを作成ます。 これらの手順はすべてターミナルで確認できます。

MDX2JSONインストーラの詳細については、プロジェクトの readmeを参照してください 。

追加の例

ドキュメントからの例 Samplesネームスペースの Sample.Installerクラス
CacheGitHubCIプロジェクトは、 インストーラを提供します 。
SYSMONダッシュボードプロジェクトは、 インストーラを提供します 。
DeepSee監査プロジェクトは、 インストーラを提供します 。

概要

%Installerは、InterSystems Caché および Ensemble
に基いてアプリケーションを配布およびデプロイするための便利なツールです。

参考: ドキュメント

#システム管理 #ツール #デプロイ #Caché #Ensemble #InterSystems IRIS #InterSystems IRIS for Health

ソースURL: https://jp.community.intersystems.com/post/installer%E3%81%A7intersystems-cach%C3%A9%E3%81%AB%E3%82%A2%E3%83%97%E3%83%AA%E3%82%B1%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%82%92%E3%83%87%E3%83%97%E3%83%AD%E3%82%A4%E3%81%99%E3%82%8B