スタジオの使用法

Version 2024.1
2024-06-03

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

スタジオの使用法
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

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

図一覧

テーブル一覧

1

スタジオの概要

重要　InterSystems スタジオは非推奨になりました。バージョン 2024.2 以降は、従来の目的でスタンドアロン配布としてのみ利用できます。"非推奨の機能とサポート中止の機能" の "スタジオ" を参照してください。

スタジオは Windows システムで動作するクライアント・アプリケーションであり、InterSystems IRIS インスタンス上で ObjectScript コードを開発するために使用できる IDE の 1 つです。

スタジオは、アプリケーションの迅速な開発に役立つ以下のような機能を単一の統合環境で提供します。

- クラスとルーチンを作成するためのエディタ。
- ObjectScript、Java、SQL、JavaScript、HTML、XML 用に統合された構文のカラー表示機能およびチェック機能
- 共通リポジトリを使用してアプリケーション・ソース・コードを作成する開発者チームのサポート
- グラフィカルなソース・コード・デバッガ
- プロジェクト単位でのアプリケーション・ソース・コードの管理

スタジオは、Windows ベースのオペレーティング・システム上で実行されるクライアント・アプリケーションです。サーバが使用しているプラットフォームやオペレーティング・システムに関係なく、(スタジオの現在のバージョンと互換性のある) すべてのインターシステムズ・サーバに接続でき、SSL/TLS で保護された接続をサポートします。

注釈　スタジオ・クライアントでは、インターシステムズ・サーバが実行しているインターシステムズ製品と同じかそれ以降のバージョンを実行している必要があります。

1.1 スタジオ接続の有効化

スタジオが InterSystems IRIS サーバに接続するには、サーバが InterSystems IRIS サーバ・マネージャの接続リストに含まれている必要があります。

また、InterSystems IRIS サーバが適切に設定されている必要があります。"高度な Web サーバ構成" を参照し、接続するサーバの WebServerName、WebServerPort、および WebServerURLPrefix の設定を確認してください。また、`%Service_Bindings` サービスが有効になっていることも確認してください。

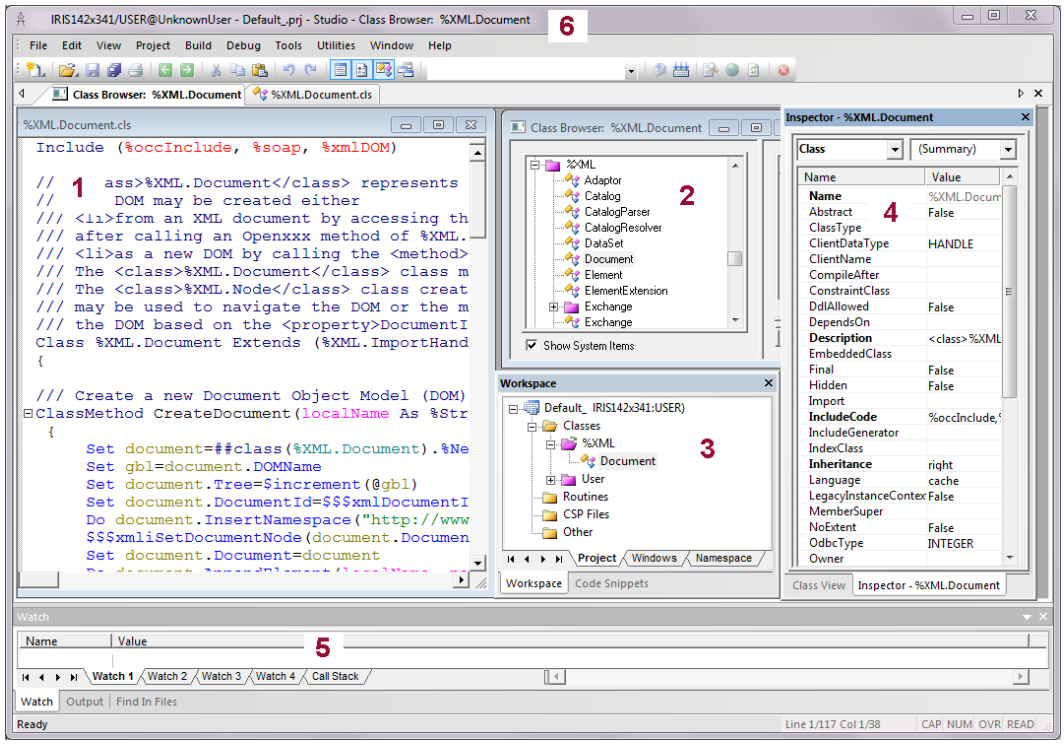
1.2 スタジオでの非 Latin1 エンコードの使用の有効化

スタジオでは非 Unicode エンコードが使用されるため、ロシア語などの言語で使用される非 Latin1 文字エンコードを処理するには、Windows OS レベルで言語エンコードを構成する必要があります。詳細は、Windows OS のドキュメントを参照してください。

1.3 スタジオ・ウィンドウの概要

スタジオのユーザ・インタフェースの主なコンポーネントは、以下のとおりです。

図 1-1: スタジオ・コンポーネント



1. エディタ・ウィンドウ：クラス・エディタはクラス定義の編集、ルーチン・エディタはルーチンおよびインクルード・ファイルの編集、CSP エディタは CSP 定義テキストの編集に使用します。InterSystems IRIS® での CSP ファイルの使用はお勧めしません。
2. [クラス・ブラウザ] ウィンドウ：既存のクラスを表示します。
3. [ワークスペース] ウィンドウ：3 つのタブがあり、現在のプロジェクトのコンテンツ、開いているすべてのウィンドウ、現在のネームスペースのコンテンツを表示できます。
4. クラス・インスペクタ・ウィンドウ：クラス定義内のキーワードを表示および変更します。
5. [ウォッチ] ウィンドウ：変数を表示します。
6. タイトル・バー：ConnectionName/Namespace@UserName - ProjectName.prj - Studio - ActiveDocument を表示します。アクティブ・ドキュメントが最大化されている場合は、名前は角括弧内に表示されます。

上図のウィンドウに加えて、スタジオには、一般的なタスクを実行するのに役立つウィザードやテンプレートが用意されています。これには、以下のものがあります。

- [ファイルから検索] ウィンドウ : 検索ウィンドウを表示します。
- 出力ウィンドウ : インターシステムズ・サーバから返された出力を表示します (クラスのコンパイル中に生成されたメッセージなど)。
- コード・スニペット・ウィンドウ : ユーザが作成したコード・スニペットを表示およびドラッグします。
- 新規クラス・ウィザード : 新規のクラスを定義します。
- 次のクラス定義にメンバを追加するクラス・メンバ・ウィザード : プロパティ、インデックス、リレーションシップ、メソッド、パラメータ、SQL トリガ、クエリ、プロジェクション、ストレージ、外部キー、および XData ブロック。
- 他のテクノロジからクラスを作成するウィザード : Java クラスおよび jar ファイル、XML スキーマ、SOAP クライアント・クラスからクラスを作成するウィザード、COM オブジェクトへのアクセスを可能にするウィザード、および .NET の DLL アセンブリ・ファイルからクラスを作成するウィザード。
- 次の要素を追加する HTML テンプレート : 色、テーブル、タグ、およびスクリプト。
- CSP フォーム・ウィザード : CSP ページ内のオブジェクトに結合された HTML フォームを作成します。InterSystems IRIS での CSP ファイルの使用はお勧めしません。

注釈　スタジオは Unicode をサポートしないコントロールで構築されているので、Unicode アプリケーションではありません。特に、[ファイルから検索] ウィンドウは Unicode をサポートしていません。

1.4 コードの可視性

[ワークスペース] ウィンドウの **[ネームスペース]** タブには、現在のネームスペースにあるコードが表示されますが、以下の例外があります。

- **%SYS** ネームスペースでは、**%** で始まる名前のクラスとルーチンも含め、すべてのクラスとルーチンが表示されます。
- 他のネームスペースでは、**%** で始まる名前のクラスとルーチンを除き、そのネームスペースで利用できるすべてのクラスとルーチンが既定で表示されます。% で始まる名前のクラスとルーチンを表示するには、[開く] ダイアログ・ボックス (**[ファイル]→[開く]**) の **[システム・アイテムを含む]** チェック・ボックスにチェックを付けます。

1.5 プロジェクト

スタジオでは、プロジェクトを使用して、アプリケーションのソース・コードを管理できます。

プロジェクトは、クラス定義、ルーチン、インクルード・ファイルの集まりです。例えば、スタジオ・プロジェクトを作成して、1 つのアプリケーションに関するクラスをすべてまとめることができます。

ユーザが作成したプロジェクトであっても、最初にスタジオを開いたときに作成される既定のプロジェクトであっても、常にそのプロジェクト内で作業を行います。既定のプロジェクト名は、`Default_`yourusername (接頭語 `Default_` の後にユーザ名) となります。

1 つのプロジェクトのファイルはすべて、同じネームスペース (および同じインターシステムズ・サーバ) に存在する必要があります。個々のクラスまたはルーチンは、複数のプロジェクトと関連付けることができます。1 つのネームスペースには、複数のプロジェクトを配置することができます。

プロジェクトには、インターシステムズ・ネームスペースのクラス階層などの情報が保存され、クラスを編集する際にその情報を使用します。また、プロジェクトには、デバッグ情報（デバッグするアプリケーションの起動方法など）も保存されます。

1.6 クラス定義

クラス定義ではクラスを定義します。クラス定義は、クラス・メンバ（プロパティやメソッドなど）、キーワードと呼ばれるその他の項目、およびそれぞれの対応する値から成り、これらによってクラスの振る舞いの詳細を指定します。

クラス定義は、データベースのクラス・ディクショナリに保存されています。クラス定義をコンパイルし、実行可能コードを生成することによって、クラス定義に基づくオブジェクト・インスタンスを生成できます。クラスに対して生成された実行可能コードのソース・コードは、1 つまたは複数のルーチンから成ります。生成されたこれらのルーチンは、スタジオで表示できます。

クラス定義は、他のテクノロジで使用するために投影できます。SQL の場合、このプロジェクション（投影）は自動的に行われます。Java の場合、別のコンパイルの過程を経て、クラス定義に対応する Java クラスが生成されます。詳細は、"クラスへのプロジェクションの追加" を参照してください。

スタジオでは、クラス定義をクラス・エディタ・ウィンドウで表示し、編集することができます。また、クラス定義をクラス・インスペクタ・ウィンドウに表示し、キーワードやそれに対応する値を表形式で編集することもできます。

1.7 CSPファイル（従来）

従来のアプリケーションにも CSP ファイルを含めることができます。これは、タグベースの開発で使用されます。タグベースの開発モデルでは、開発者は Web アプリケーションによりアクセスされるディレクトリ構造に含まれる .csp ファイルを作成します。ファイルには、サーバとの通信を可能にする HTML タグと特殊タグの組み合わせが含まれています。CSP コンパイラはファイルを読み取り、それらからクラス定義を生成します。次いでクラス定義は実際のランタイム HTML を生成します。タグベースの開発の詳細は、Caché/Ensemble ドキュメントの "Tag-based Development with CSP" を参照してください。

InterSystems IRIS での CSP ファイルの使用（タグベースの開発）はお勧めしません。代わりに CSP クラスを作成できます。"CSP ベースの Web アプリケーションの作成" を参照してください。

1.8 ルーチン・エディタ

ルーチン・エディタでは、構文をカラー表示する機能を使用して、特定のルーチンのソースを直接作成したり、編集することができます。また、インクルード・ファイルの編集にもルーチン・エディタを使用します。

1.9 複数のユーザへのサポート

スタジオは、オブジェクト・ベースの、クライアント・サーバ・アプリケーションです。スタジオで作成し編集できるソース・ファイル（クラス定義、ルーチン、インクルード・ファイル）はインターシステムズ・サーバに保存して、オブジェクトとして表示されます。

スタジオからソース・ファイルを保存する場合は、現在接続しているインターシステムズ・サーバにファイルを保存されます。スタジオで表示しているソース・ファイルがサーバ側で変更された場合、その変更が通知され、変更後のファイルをロードするかどうかを尋ねられます。

スタジオは、複数のユーザが同じソース・コンポーネントを同時に表示していることを自動的に検知し、並行アクセスを管理します。他のユーザが編集しているファイルを開こうとした場合は、他のユーザに使用されていることが通知され、そのファイルを読み取り専用モードで開くかどうかを尋ねられます。

1.10 ドキュメントのローカルでのインポートおよびエクスポート

通常、スタジオで作業しているすべてのドキュメントは (クラス定義やルーチンなど)、インターシステムズ・データベース (リモート・マシンに存在する場合があります) に保存されます。**[ツール]→[エクスポート]** と **[ツール]→[インポート]** を使用すれば、ローカル・ファイルのエクスポートとインポートを行うことができます。

クラス定義とルーチンは、XML ドキュメントとしてローカル・ファイルに保存されます。

1.11 デバッグ

スタジオには、ソース・レベルの GUI デバッガが用意されています。このデバッガは、スタジオが接続している同一インターシステムズ・サーバで実行中のターゲット・プロセスにアタッチします (または、起動してからアタッチします)。アタッチされたデバッガは、ターゲット・プロセスをリモート制御できるため、変数のウォッチ、コードのステップ実行、ブレークポイントの設定などを行うことができます。

通常、デバッガを使用するためにはプロジェクトを開く必要があります。プロジェクトには、ターゲット・プロセスのデバッグを開始するのに必要な情報 (ルーチン、メソッド、またはクライアント・アプリケーションの名前) が保存されています。また、プロジェクトには、以前のデバッグ・セッションで設定されたブレークポイントのリストが保存されます。

プロジェクトを開かずに、実行中のプロセスへアタッチし、デバッグすることもできますが、その場合、スタジオでは、前のセッションで設定されたブレークポイントを使用できません。デバッグに関する詳細は、"スタジオ・デバッガの使用"を参照してください。

1.11.1 オブジェクト・ベースのアプリケーションのデバッグ

現在、スタジオではソース・レベルでのみ、INT (ObjectScript ルーチン) のデバッグができます。クラスでステップ実行やブレークポイントの設定を行うには、対応する INT ファイルを開き、そのファイルでデバッグ・コマンドを使用します。

クラスに対して生成されたソース・コードを利用できるようにするには:

1. **[ツール]→[オプション]** を選択します。
2. 左側のペインで **[コンパイラ]→[フラグおよび最適化]** に移動します。
3. **[生成されたソース・コードを保存]** チェック・ボックスにチェックを付けます。

1.12 セキュリティ

インターシステムズのセキュリティ機能によって、スタジオの使用、インターシステムズ・サーバへのスタジオの接続、および SSL/TLS で保護された接続のサポートが制御されます。スタジオを起動すると、ログイン画面が表示されます。スタジオを使用するには、以下の特権を有するユーザとしてログインする必要があります。

- **%Development:Use** – %Development リソースに対する Use 許可は、さまざまな開発関連リソースへのアクセス権を付与します。
- **%Service_Object:Use** – **%Service_Object** リソースに対する Use 許可は、スタジオへのアクセスを制御する %Service_Bindings サービスへのアクセス権を付与します。

また、デフォルト・データベースへの Read または Write 許可を持つ場合のみ、ネームスペースへの接続が可能になります。

ユーザがこれらのさまざまな特権を付与される方法は、インスタンスのセキュリティ・レベルによって異なります (以下の一覧で説明します)。

- 最小限のセキュリティが設定されたインスタンスでは、UnknownUser を含むすべてのユーザがすべてのネームスペースに対するすべての特権とアクセス権を持ちます。スタジオのログイン画面が表示された場合、**[ユーザ名]** と **[パスワード]** フィールドを空白のままにするか、ユーザ名とパスワードのペアとして _SYSTEM と SYS を入力します。
- 標準のセキュリティが設定されたインスタンスでは、指定された特権を明示的に付与される必要があります。これは、これらの特権を持っているロールに割り当てることによって確立されます。
- ロックダウン・セキュリティが設定されたインスタンスでは、スタジオへのアクセスを制御するサービス (**%Service_Bindings**) が既定で無効に設定されています。既定では、すべてのユーザはスタジオへのアクセス権を持っていません。

スタジオの認証設定を変更するには:

1. InterSystems IRIS ランチャーを使用して管理ポータルを開きます。
2. **[システム管理]→[セキュリティ]→[サービス]** の順に選択します。
3. **[表示] または [サービス定義編集]** セクションで **[実行]** をクリックします。
4. [%Service_Bindings] をクリックします。
5. **[許可された認証方法]** のチェック・ボックスにチェックを付けるかチェックを外します。

注釈　　スタジオへのアクセス権は、インストール以降に行った既定の設定に対する変更に影響される場合があります。

1.13 ソース・コントロール・フック

スタジオには、カスタム・フック (ドキュメントがロードまたは保存されるとき、必ずインターシステムズ・サーバで実行されるコード) を実装するメカニズムがあります。通常、これらのフックは、ソースやリビジョン管理システムへの接続を実装するのに使用されます。詳細は、"InterSystems IRIS とソース・コントロール・システムの統合" を参照してください。

1.14 コマンド行からのスタジオの実行

スタジオをシステムのコマンド行から実行するには、install-dir¥bin ディレクトリ内の Cstudio.exe コマンドを使用します。このコマンドとそのパラメータでは大文字と小文字が区別されます。

パラメータ	説明
?	ヘルプ情報を表示します。
/Servername=ServerName	ServerName で指定した名前のサーバに接続します。
/Server=cn_iptcp:127.0.0.1[51773]::	ip address[port] のサーバに接続します。
/Namespace=User	User で指定したネームスペースに接続します。サーバも定義する必要があります。
/Project=MyProject	MyProject で指定したプロジェクトを開きます。サーバとネームスペースも定義する必要があります。
cn_iptcp://127.0.0.1:51773/User/test.int	ルーチン test.int をロードします。cn_iptcp は、大文字と小文字が区別されるプロトコル識別子です。
/files="tag+1^myroutine.int",User.Class1.cls	指定したドキュメントを開いて、指定した位置にカーソルを配置します。サーバとネームスペースも定義する必要があります。
/pid=123	プロセスにアタッチします。サーバとネームスペースも定義する必要があります。
/fastconnect=127.0.0.1[51773]:USER:_SYSTEM:SYS	ip address[port]:USER:username:password で指定した内容に基づいて、レジストリ内の接続定義を使用せずに接続します。

2
クラス定義の作成

スタジオでは、クラス定義を作成して、編集できます。クラス定義では、特定のクラスのメンバ (メソッド、プロパティ) や特性 (スーパークラス) などのコンテンツを指定できます。

スタジオでは、以下のツールを使用して、クラス定義を作成できます。

- ウィザードを使用して、クラスやクラス・メンバをすばやく生成
- クラス・インスペクタを使用して、表形式でクラスの特性を表示および編集
- クラス・エディタを使用して、クラス定義を直接編集。クラス・エディタは、構文のカラー表示や構文のチェックなどを行える、フル機能のテキスト・エディタです。複数のオプションを選択できるコード補完ドロップダウン・メニューも備えています。

これらのツールはすべて切り替えて使用できます。スタジオでは、これらのツールは自動的に同期化されて表示されます。

ここでは、クラス定義の作成の概要について説明します。これ以降の章のほとんどでは、プロパティ、メソッド、パラメータなどのクラス・メンバの作成方法について説明します。

2.1 新規のクラス定義の作成

スタジオでは、新規クラス・ウィザードを使用して、新規のクラス定義を作成できます。

注釈　スタジオでクラス定義を作成する前に、プロジェクトを開いている必要があります。クラス定義を作成するとき、スタジオとInterSystems IRIS® サーバ間では、さまざまなやり取り (クラスのリストの参照やクラスのコンパイル要求など) が行われます。スタジオでは、これらのサーバとのやり取りの詳細を管理するために、プロジェクトが必要になります。

2.1.1 新規クラス・ウィザード

新規クラス・ウィザードを開くには、**[ファイル]→[新規作成]→[一般]** を選択して **[　　]** を選択します。

新規クラス・ウィザードでは、情報の入力が促されます。**[完了]** ボタンはいつでも選択できます (この場合、指定していない情報には既定値が設定されます)。

2.1.1.1 名前および説明ページ

新規クラス・ウィザードでは、以下の情報の入力を求められます (クラス名やパッケージ名以外は、後で値を変更できます)。

パッケージ名

新規のクラスが属すパッケージ。既存のパッケージ名を選択するか、新規の名前を入力します。新規の名前を入力した場合、クラス定義を保存するときに新規のパッケージが自動的に作成されます。プロパティ名に使用できる句読点は、ドット (.) と先頭のパーセント記号 (%) だけです。

パッケージの詳細は、"パッケージのオプション" を参照してください。

クラス名

新規のクラス名。これは、有効なクラス名である必要があります。また、定義済みの既存のクラスと同じ名前を付けることはできません。クラス名は、後に変更できないことに注意してください。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規クラスに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントに表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

2.1.1.2 クラス・タイプ・ページ

新規クラス・ウィザードでは、生成するクラスのタイプが尋ねられます。定義済みの既存のクラスを拡張 (継承) するか、以下のオプションの 1 つを選択することで、新規のクラスを生成します。

Persistent

永続クラスの定義を生成します。永続オブジェクトはデータベースに保存できます。

Serial

シリアル・クラスの定義を生成します。シリアル・オブジェクトは、アドレスなどの複雑なデータ型を生成するために、永続オブジェクト内に組み込まれます。

Registered

登録クラスの定義を生成します。登録オブジェクトは、データベースに保存されません。

抽象

スーパークラスを持たない抽象クラスの定義を生成します。

データ型

データ型クラスの定義を生成します。データ型クラスは、ユーザ定義のデータ型を生成するのに使用されます。

CSP (HTTP イベントの処理に使用)

%CSP.Page クラスの定義を生成します。CSP クラスは、CSP イベント処理クラスを生成するために使用します。これは、CSP ページを生成するため、または HTTP イベントに応答する (例えば、XML サーバを作成する) ためのプログラム的な方法です。InterSystems IRIS での CSP ファイルの使用はお勧めしません。

Extends

既存のクラスを拡張します。[Extends] にチェックを付けて、既存のスーパークラスの名前を入力 (またはリストから選択) します。

2.1.1.3 データ型クラスの特性ページ

新規のデータ型クラスを生成する場合、新規クラス・ウィザードでは、データ型クラス特有の項目の入力が促されます。これには、以下のものがあります。

クライアント・データ型

クライアント・アプリケーション内でこのデータ型を表すために、クライアントで使用されるデータ型。

ODBC データ型

このデータ型を表すために、ODBC または JDBC で使用されるデータ型。このデータ型を ODBC/JDBC ベースのアプリケーションに表示する型を選択します。

SQL カテゴリ

SQL カテゴリは、このデータ型で論理演算を実行するときに、InterSystems SQL エンジンによって使用されます。

2.1.1.4 永続クラス、シリアル・クラス、登録クラスの特性ページ

新規の永続クラス、シリアル・クラス、または登録クラスを生成する場合、新規クラス・ウィザードでは、永続クラスまたはシリアル・クラス特有の項目の入力が促されます。これには、以下のものがあります。

所有者

(オプション) 永続クラスの場合、新規クラスの所有者となる SQL ユーザ名を入力します。このユーザ名は、このクラスが SQL 経由で使用されるときの権限を制御します。このフィールドが空欄の場合、所有者は既定の _system が使用されます。

SQL テーブル名

(オプション) 永続クラスの場合、このクラスに対応する SQL テーブルに使用する名前を入力します。このフィールドを空白にした場合、SQL テーブル名はクラス名と同じ名前になります。クラス名が有効な SQL 識別子でない場合、ここに SQL テーブル名を入力する必要があります。

XML 対応

(オプション) このオプションを選択した場合、クラスは XML 対応になり、そのクラス自体を XML ドキュメントに投影できるようになります。また、Web サービス・メソッドにも使用できるようになります。これは、クラスのスーパークラス・リストに %XML.Adaptor クラスを追加する方法と同じです。

詳細は、"XML ツールの使用法" と "Web サービスおよび Web クライアントの作成" を参照してください。

Zen DataModel

InterSystems IRIS では、この機能はサポートされていません。

データ生成

(オプション) このオプションを選択した場合、新規クラスは自動的なデータ生成をサポートします。これは、クラスのスーパークラス・リストに %Library.Populate クラスを追加する方法と同じです。

自動データ生成によって、ランダムなデータを簡単に生成し、クラスの動作をテストすることができます。クラスを生成するには、クラスをコンパイルし、そしてクラスの Populate メソッド (**%Library.Populate** クラスから継承) を実行します。以下にターミナルの使用例を示します。

ObjectScript

```
Do ##class(MyApp.Person).Populate(100)
```

詳細は、"Populate ユーティリティの使用" を参照してください。

2.1.1.5 CSP クラスの特性ページ

新規の CSP クラスを生成している場合、新規クラス・ウィザードでは、以下の値の入力を求められます。

コンテンツ・タイプ

CSP クラスで使用されるコンテンツ・タイプを指定します。利用できるオプションは、**HTML** または **XML** です。このオプションを使用して、新規クラスの CONTENTTYPE パラメータ値を、**text/html**、もしくは **text/xml** に設定します。このオプションは、後で変更できます。

InterSystems IRIS での CSP ファイルの使用はお勧めしません。

2.1.2 新規クラス・ウィザード実行の結果

新規クラス・ウィザードを実行した後、スタジオには新しいクラス・エディタ・ウィンドウが表示されます。クラス・エディタ・ウィンドウには、新規のクラス定義が追加されています。以下に例を示します。

Class Definition

```
/// This is a Person class
class MyApp.Person extends %Persistent
{
}
```

このクラス定義を InterSystems IRIS データベース内に保存したり、プロパティやメソッドなどのクラス・メンバを追加したり、クラス・インスペクタを使用してクラス定義を編集したりできます。

2.2 クラス定義をオープンする

ワークスペース・ウィンドウの [プロジェクト] タブでクラスを選択し、そのクラスをダブルクリックすると、以前保存したクラス定義をオープンし、クラス・エディタ・ウィンドウで表示できます。

オープンするクラス定義が現在のプロジェクトのものではない場合は、最初に、**[プロジェクト]→[クラス追加]** を選択して、現在のプロジェクトに追加しておきます。

オープンするクラス定義が、他のユーザによって編集されている場合、そのクラス定義を読み取り専用でオープンするかどうかが確認されます。

2.3 クラス定義の編集

新規に生成したクラス定義、または既存のクラス定義の特性は変更できます (クラス名とパッケージ名は除く)。これは、以下の 2 つの方法で行うことができます。

- クラス・インスペクタを使用して、クラスまたはクラス・メンバのキーワードの値を変更する方法
- クラス・エディタを使用して、クラス定義の値を変更する方法

クラス・キーワードとそれぞれの意味のリストは、"クラスの定義と使用" を参照してください。クラス定義の詳細は、"クラス定義言語" リファレンスを参照してください。

2.4 クラス定義の保存および削除

クラス定義を変更した場合、以下のいずれかの方法で InterSystems IRIS データベースに保存します。

- **[ファイル]→[保存]** を使用して、現在のウィンドウのコンテンツを保存します。
- **[プロジェクトを保存]** を使用して、現在のプロジェクト内の変更されたすべてのクラス定義を保存する方法

クラス定義を削除するには、ワークスペース・ウィンドウで、クラスを選択して、**[編集]→[クラス削除]** classname を選択します。クラスと生成されたすべてのファイルが削除されます。

2.5 クラス定義のコンパイル

スタジオでは、以下の方法でクラス定義をコンパイルできます。

- **[ビルド]→[コンパイル]** または **[コンパイル]** アイコン  を使用する方法。これは、変更されたすべてのクラス定義を保存し、実行中のエディタ・ウィンドウに表示されている現在のクラス定義をコンパイルします。
- **[ビルド]→[すべて再ビルド]** または **[すべて再ビルド]** アイコン  を使用する方法。これにより、変更済みのオープン・クラス定義がすべて保存され、現在のプロジェクトのクラスがすべてコンパイルされます。

注釈　**[ツール]→[オプション]** ダイアログの **[コンパイル]** タブを使用して、クラスをコンパイルする方法を指定できます。

2.5.1 増分コンパイル

スタジオでは、クラスの増分コンパイルを実行できます。この機能は、**[関連最新アイテムをスキップ]** オプションにチェックを付けると有効になります。このオプションを探すには、**[ツール]→[オプション]** ダイアログの **[コンパイラ]→[フラグおよび最適化]** タブを開きます。

このオプションを有効にし、1 つ以上のメソッドのソース・コードに変更を加えた場合、**[ビルド]→[コンパイル]** を選択すると、変更されたメソッドのみをコンパイルできます(オーバーライドするには **[ビルド]→[すべて再ビルド]** を使用します)。クラス・インタフェース (プロパティやメソッド・シグニチャなど) やストレージ定義を変更した場合はすべて、完全コンパイルが必要です。

増分コンパイルは通常、完全コンパイルよりも処理が高速であるため、開発中のメソッド (アプリケーション・ロジック) に部分的変更を行う場合の処理をスピードアップできます。

増分コンパイルは、以下のように動作します。

1. クラス・コンパイラは、実装が変更されたすべてのメソッドを検索し、その実行時コードを MyApp.MyClass.5.INT などの新規のルーチンに配置し、このルーチンをコンパイルします。

2. 次に、クラス・コンパイラは、コンパイルされたメソッドの新規の実装を使用できるように、クラスの実行時のクラス記述子を変更します。アプリケーションがこのメソッドのうちの 1 つを呼び出すとき、新規のコードは送信され、実行されます。
3. 残りのクラス定義 (コンパイルされたメタ情報、永続クラスのストレージ情報、実行時 SQL 情報) は、変更されません。変更されたメソッドの以前の実装は、実行時コードに残されますが、実行はされないことに注意してください。

完全な (増分でない) コンパイルが実行されるとき、増分コンパイルされたメソッドを含むすべての余分なルーチンが削除されます。アプリケーションを配置するときは、事前にすべてのクラスで完全コンパイルを実行し、余分なルーチンを削除しておきます。

2.6 クラス定義の名前を変更する

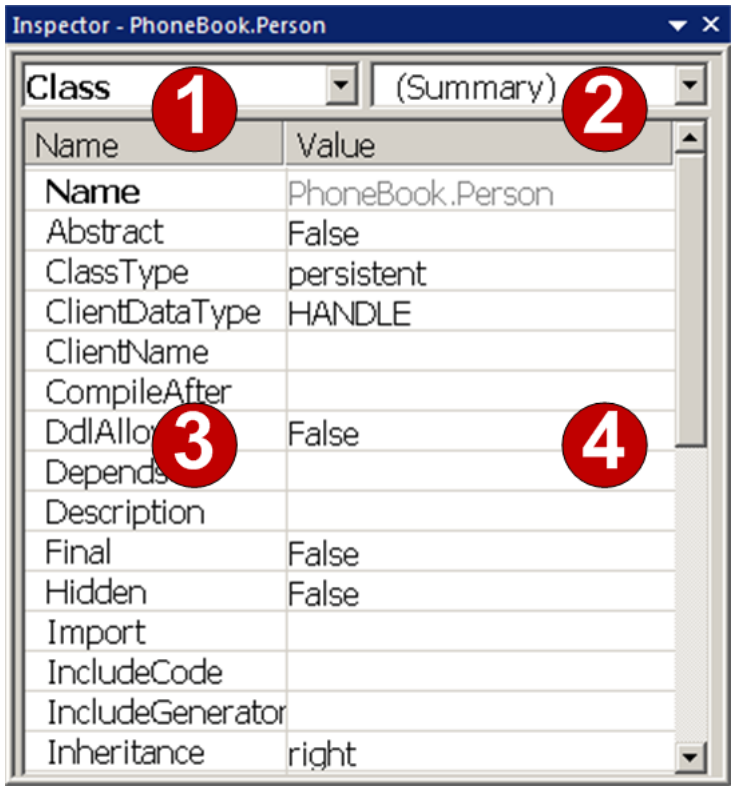
クラス定義を一度生成すると、その名前を変更することはできません。しかし、クラスのコピーを作成し、それに新しい名前を付けることはできます。手順は以下のとおりです。

1. **[ツール]→[クラスをコピー]** を選択します。
2. **[クラス定義のコピー元]** フィールドで、新しい名前を付けるクラスを選択します。
3. **[コピー先]** フィールドに新しいクラス名を入力します。
4. [ ]、[ ]、[ ] の 3 つのオプションのいずれかを選択します。
5. [OK] をクリックします。
6. 元のクラス定義のコピーが含まれた、新しいクラス・エディタ・ウィンドウが表示されます。クラス・エディタを使用して、希望の変更を行うことができます。この新規のクラス定義は、いつでも保存できます。
7. 必要な場合は、従来のクラス定義を削除することもできます。

2.7 クラス・インスペクタ

クラス・インスペクタには、現在のクラス定義が編集可能な表形式で表示されます。クラス・インスペクタの主なコンポーネントは以下のとおりです。

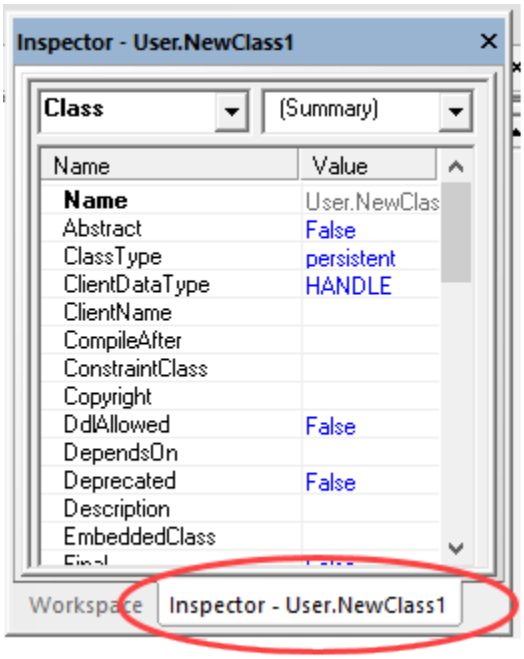
図 2-1: クラス・インスペクタ



1. メンバ・セレクタ：表示するキーワードのグループを制御します。クラス全体のキーワード、または特定のクラス・メンバ (プロパティやメソッド) のキーワードのいずれかを選択して表示できます。
2. 項目セレクタ：どのクラス・メンバ (特定のプロパティなど) を表示するのかを制御します。リストのコンテンツは、メンバ・セレクタの値によって異なります。(Summary) を選択すると、メンバ・セレクタで指定されたタイプの、すべてのメンバのリストが表示されます。
3. キーワード：現在のクラスのキーワードのリスト、またはメンバ・セレクタと項目セレクタによって選択されたクラス・メンバのキーワードのリストが表示されます。キーワードをハイライト表示すると説明が表示され、その値を編集できます (値の右の **[編集]** を選択するか、値を直接編集します)。値が明示的に設定された (継承されていない、または既定で設定されていない) キーワードは、bold で表示されます。
4. 値：キーワード・リストに表示されているキーワードの値が表示されます。クラス定義が最後に保存された以後に変更された値は、青で表示されます。

2.7.1 クラス・インスペクタの開始

クラス・インスペクタを開始するには、**[参照]→[インスペクタ]** を選択します。クラス・インスペクタでは、ワークスペースとペインを共有します。**[インスペクタ]** タブをクリックして、クラス・インスペクタにアクセスします。



2.7.2 クラス・インスペクタの起動

クラス・インスペクタを起動したときには、現在の情報が表示されます（起動していないときは、グレー表示されます）。クラス・インスペクタを起動する手順は以下のとおりです。

1. 現在のエディタ・ウィンドウには、クラス定義が表示されていることを確認してください（ルーチンの場合、クラス・インスペクタは機能しません）。
2. クラス・インスペクタを選択します。

クラス・インスペクタが起動すると、その背景は白色に変わり、コンテンツは更新されて現在のクラス定義が反映されます。インスペクタを使用してキーワード値を変更すると、対応するクラス・エディタ・ウィンドウが休止します（グレー表示されます）。インスペクタでの作業が終了したら、元のクラス・エディタ・ウィンドウを選択します。これで、クラス・エディタ・ウィンドウがアクティブになり、クラス・インスペクタで行った変更の結果が表示されます。

クラス・インスペクタで右クリックすると、ポップアップ・メニューが表示され、新規のクラス・メンバの追加などの操作を行えます。

2.8 クラス・ブラウザ

スタジオのクラス・ブラウザ・ユーティリティを使用すると、クラス階層に配置された利用可能なすべてのクラスを表示できます。これらのクラスごとに、スーパークラスから継承されたものも含めたプロパティやメソッドなどのクラス・メンバを表示することができます。クラス・ブラウザには、クラス・メンバが表形式で表示されます。列のタイトルを選択すると、その列でクラス・メンバを並べ替えることができます。

1. **[ツール]→[クラス・ブラウザ]** を使用してクラス・ブラウザを開きます。
2. クラス・ブラウザの項目を右クリックすると、その項目をプロジェクトに追加するか、クラス・エディタに表示するか、あるいはドキュメントを表示するかを選択できます。

2.9 スーパークラス・ブラウザと派生クラス・ブラウザ

スタジオには、すべてのスーパークラスを表示するブラウザと、現在のクラス定義の派生クラスを表示するブラウザがあります。

2.9.1 スーパークラス・ブラウザ

[クラス]→[スーパークラス] を選択してスーパークラス・ブラウザを開くと、現在のクラスのすべてのスーパークラスがアルファベット順に表示されます。

クラスを選択してからボタンを選択すると、そのクラスを現在のプロジェクトに追加したり、クラス・エディタに表示したり、あるいはドキュメントを表示できます。

2.9.2 派生クラス・ブラウザ

[クラス]→[派生クラス] を選択して派生クラス・ブラウザを表示すると、現在のクラス定義から派生したすべてのクラスがアルファベット順に表示されます。

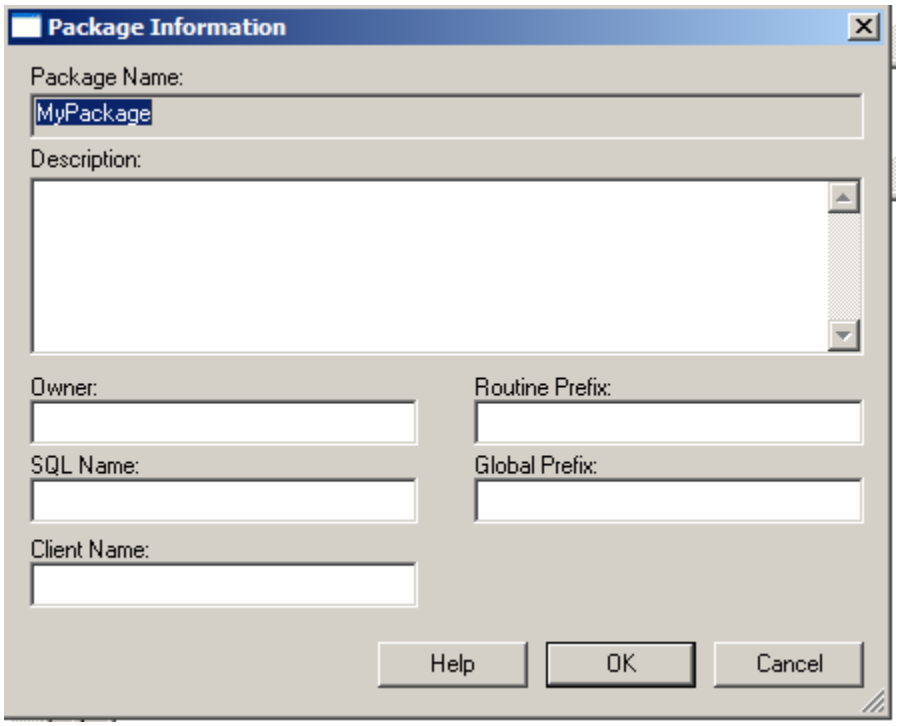
クラスを選択してからボタンを選択すると、そのクラスを現在のプロジェクトに追加したり、クラス・エディタに表示したり、あるいはドキュメントを表示できます。

2.10 パッケージ情報

スタジオでは、[パッケージ情報] ダイアログを使用して、特定のクラス・パッケージに関する情報を表示したり、編集したりできます。

パッケージ情報を表示するには、[ワークスペース] ウィンドウの [　　] タブで、パッケージ名を右クリックして **[パッケージ情報]** を選択します。

図 2-2: [パッケージ情報] ダイアログ



[パッケージ情報] ウィンドウには、以下の情報が表示されます。

パラメータ	説明
パッケージ名	パッケージの名前。
説明	パッケージの説明。
所有者	このパッケージの SQL の所有者名。これは、パッケージの SQL 表示に対するスキーマ・ワイドの権限を提供するために使用されます。
SQL 名	パッケージの関係を示すために使用される SQL スキーマの名前。
クライアント名	このパッケージのクラスに生成されたプロジェクションに使用されたパッケージ名。例えば、このパッケージに bank.account というクラスが含まれており、このパッケージに com.mycompany.bank というクライアント・パッケージ名を付けた場合、クラスがコンパイルされると、このクラスの Java プロジェクションは com.mycompany.bank.account になります。
ルーチン接頭語	このパッケージ内のクラスから生成されたルーチンの接頭辞として使用される文字列
グローバル接頭語	このパッケージ内の永続クラスによって使用される、既定のグローバル名の接頭辞として使用される文字列

クラス・パッケージの詳細は、"パッケージ" を参照してください。

3

クラスへのプロパティの追加

ここでは、クラス定義にプロパティを追加する方法について説明します。

オブジェクトのデータや状態は、プロパティに格納されます。クラス定義には、複数のプロパティ定義が含まれています (プロパティ定義が含まれていないこともあります)。

新規のプロパティをクラスに追加するには、以下の 2 つの方法があります。

- クラス・エディタでプロパティをクラス定義に追加する方法
- 新規プロパティ・ウィザードを使用する方法

クラス・エディタを使用してプロパティを追加するには、クラス・エディタの空の行にカーソルを置き、プロパティ宣言を入力します。

Class Definition

```
Class MyApp.Person Extends %Persistent
{
Property Name As %String;
Property Title As %String;
}
```

また、既存のプロパティ宣言をコピーして新しい場所に貼り付け、それを編集することもできます。

プロパティの定義の詳細は、"リテラル・プロパティの定義と使用" およびそれに続く章を参照してください。"クラス定義リファレンス" の "プロパティ定義" も参照してください。

3.1 新規プロパティ・ウィザード

新規プロパティ・ウィザードを開くには、**[クラス]→[追加]→[プロパティ]** を選択します。または、クラス・インスペクタで右クリックして **[追加]→[新規プロパティ]** を選択するか、プロパティのみが表示されている場合は (は左列の先頭にあります)、右クリックして **[新規プロパティ]** を選択するか、ツールバーから **[新規プロパティ]** アイコン  を選択することもできます。

新規プロパティ・ウィザードでは、情報の入力が促されます。**[完了]** ボタンはいつでも選択できます。指定していない情報に対しては既定値が設定されます。

3.1.1 名前および説明ページ

新規プロパティ・ウィザードでは、以下の情報の入力が促されます (これらの値は後で変更できます)。

プロパティ名

(必須) 新規プロパティの名前。これは、有効なプロパティ名である必要があります。また、定義済みの既存のプロパティと同じ名前を付けることはできません。プロパティ名に使用できる句読点は、ドット (.) と先頭のパーセント記号 (%) だけです。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規プロパティに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントに表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

3.1.2 プロパティ・タイプ・ページ

新規プロパティ・ウィザードでは、プロパティのタイプ (単一値、コレクション、ストリーム、またはリレーションシップ) を選択するように促されます。データ型などの属性を追加することで、これらの選択をさらに詳細に指定できます。

単一値

単一値プロパティは、単独の値が含まれます。単一値プロパティは、関連するタイプを持ちます。このタイプは、InterSystems IRIS® クラスの名前です。使用されているクラスのタイプがデータ型クラスの場合、そのプロパティは単純なリテラル・プロパティです。永続クラスの場合、そのプロパティはそのクラスのインスタンスに対する参照です。シリアル・クラスの場合、そのプロパティは埋め込みオブジェクトを表します。

クラス名は直接入力するか、**[参照]** ボタンを使用して、ストリームを含む利用できるクラスのリストから選択することもできます。

InterSystems IRIS で用意されている基本的なデータ型クラスの詳細は、"データ型" を参照してください。

コレクション

コレクション・プロパティには、複数の値が含まれます。コレクション・タイプには、リスト (単純に並べられたリスト) と配列 (キー値と関連する要素との単純なディクショナリ) の 2 つがあります。単一値プロパティと同様、コレクション・プロパティにもデータ型があります。この場合、データ型では、コレクション (集合) に含まれる要素のタイプを指定します。

"コレクションを使用した作業" を参照してください。

リレーションシップ

リレーションシップ・プロパティは、2 つのオブジェクト間の関係を定義します。リレーションシップの詳細は、"リレーションシップの定義と使用" を参照してください。

3.1.3 プロパティの特性ページ

新規のプロパティを、永続またはシリアル・オブジェクトのクラス定義に追加する場合、新規プロパティ・ウィザードでは、さらに特性を指定する必要があります。これには、以下のものがあります。

必須

(オプション) 永続クラス、またはシリアル・クラスに唯一関連します。このプロパティが必須であることを指定します (SQL 専門用語で NOT NULL)。永続またはシリアル・オブジェクトの場合、必須プロパティには必ず値を指定する必要があります。値を指定しないと、オブジェクトの保存に失敗します。

インデックス

(オプション) 永続クラスに唯一関連します。このプロパティに基づいてインデックスを作成することを指定します。これは、このフィールドに基づいてインデックスを作成するのと同じです。

ユニーク

(オプション) 永続クラスに唯一関連します。このプロパティの値が、このクラスのエクステント・オブジェクト (つまり、インスタンスとサブクラスを含むすべてのオブジェクト) 内で一意であることを指定します。これは、このフィールドに基づいて一意のインデックスを作成するのと同じです。

計算

(オプション) オブジェクト・インスタンスが作成されるとき、計算プロパティには、メモリ内ストレージが割り当てられません。その代わり、このプロパティにアクセサ (Get、または Set) メソッドを指定する必要があります。このオプションを選択した場合、新規プロパティ・ウィザードでは、空の Get アクセサ・メソッドを生成できます。

SQL フィールド名

(オプション) 永続クラスの場合、このプロパティに対応する SQL フィールドに使用する名前を指定します。既定では (このフィールドが空白の場合)、SQL フィールド名はプロパティ名と同じになります。プロパティ名と異なるフィールド名を使用する場合、またはプロパティ名が有効な SQL 識別子でない場合は、SQL フィールド名を指定します。

3.1.4 データ型パラメータ・ページ

すべてのプロパティには、プロパティのタイプによって決定されたパラメータ値のリストがあります。これらのパラメータの値は、プロパティの振る舞いを指定します。新規プロパティ・ウィザードの [プロパティ・パラメータ] のページに表示された表を使用すると、特定のパラメータの値を指定できます。

一般的なパラメータの詳細は、"リテラル・プロパティの定義と使用" を参照してください。

3.1.5 プロパティ・アクセサ・ページ

対応するオーバーライド・チェック・ボックスにチェックを付けると、プロパティに対する Set メソッド (プロパティ値を設定するメソッド) と Get メソッド (プロパティ値を取得するメソッド) をオーバーライドできます。このオプションのいずれかを選択すると、空の Set メソッド、または Get メソッドが作成されます。メソッドの内容は、後で入力します。"プロパティ・メソッドの使用とオーバーライド" を参照してください。

3.1.6 新規プロパティ・ウィザード実行の結果

新規プロパティ・ウィザードの実行後、クラス・エディタは更新され、新規のプロパティ定義が表示されます。例えば、以下のように表示されます。

Class Definition

```
/// This is a Person class
Class MyApp.Person extends %Persistent
{
Property Name As %String;
}
```

クラス・エディタまたはクラス・インスペクタを使用して、このプロパティをさらに変更することができます。

4

クラスへのメソッドの追加

ここでは、クラス定義にメソッド定義を追加する方法、および編集する方法を説明します。

新規のメソッドをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用してクラス定義にメソッドを追加する方法
- 新規メソッド・ウィザードを使用する方法

クラス・エディタを使用してメソッドを追加するには、クラス・エディタの空の行にカーソルを置き、次のようにメソッド宣言を入力します。

Class Definition

```
Class MyApp.Person Extends %Persistent
{
Method NewMethod() As %String
{
    Quit ""
}
}
```

また、既存のメソッド宣言をコピーして貼り付け、それを編集することもできます。

メソッドの定義の詳細は、"メソッドの定義と呼び出し" を参照してください。"クラス定義リファレンス" の "メソッド定義" も参照してください。

4.1 新規メソッド・ウィザード

[クラス]→[追加]→[メソッド] を選択して、新規メソッド・ウィザードを起動できます。または、クラス・インスペクタで右クリックして、**[追加]→[新規メソッド]** を選択します。ツールバーの **[新規メソッド]** ボタン  をクリックすることもできます。

新規メソッド・ウィザードでは、情報の入力が促されます。**[完了]** ボタンはいつでもクリックできます。指定していない情報に対しては既定値が設定されます。

4.1.1 名前および説明ページ

新規メソッド・ウィザードでは、以下の情報の入力が促されます (これらの値は後で変更できます)。

メソッド名

(必須) 新規メソッドの名前。これは、有効なメソッド名である必要があります。また、定義済みの既存のメソッドと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規メソッドに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントで表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

4.1.2 メソッド・シグニチャ・ページ

すべてのメソッドは、返りタイプ (存在する場合) や、その引数リスト (存在する場合) を示すシグニチャを持っています。メソッドのシグニチャには、以下を指定できます。

返りタイプ

(オプション) これは、メソッドから返される値のタイプを示します。このタイプは、InterSystems IRIS® クラスの名前です。クラス名は直接入力するか、**[参照]** ボタンを使用して、利用できるクラスのリストから選択することもできます。

例えば、True (1)、またはFalse (0) を返すメソッドの返りタイプは、**%Boolean** です。新規に作成するメソッドの返りタイプを指定しない場合は、このフィールドを空欄にしておきます。

引数

(オプション) メソッドの仮引数の名前、型、既定値、およびデータを渡す方法 (参照渡し、または値渡し) を示します。引数は、表形式で表示されます。テーブルにある **[追加]** ボタン  をクリックして、引数リストに新規の項目を追加することができます。このボタンをクリックすると、ポップアップ・ダイアログが表示され、引数の名前、その型、オプションの既定値、および値渡しか参照渡しかを指定できます。その下にあるボタンを使用すると、リスト内の項目の順序を並べ替えたり、項目を削除したりできます。

4.1.3 メソッド特性ページ

メソッドには、属性を追加して指定できます。これには、以下のものがあります。

プライベート

(オプション) このメソッドがパブリックであるか、プライベートであるかを示します。プライベート・メソッドは、同じクラスの他のメソッドからのみ呼び出すことができます。

最終

(オプション) このメソッドが最終であるかどうかを示します。最終メソッドは、サブクラスでオーバーライドすることはできません。

Class Method (クラス・メソッド)

(オプション) 新規のメソッドが (インスタンス・メソッドではなく) クラス・メソッドであることを示します。クラス・メソッドは、オブジェクト・インスタンスなしで実行できます。

SQL ストアド・プロシージャ

(オプション) このメソッドをストアドプロシージャとして ODBC クライアント、または JDBC クライアントからアクセスできることを示します。SQL ストアド・プロシージャとして呼び出せるのはクラス・メソッドのみです。

4.1.4 実装ページ

必要に応じて、クラス・エディタにソース・コード行を入力して、新規のメソッドに実装 (コード) を入力できます。ウィザードを実行した後、このソース・コードを入力することもできます。

4.1.5 新規メソッド・ウィザード実行の結果

新規メソッド・ウィザードの実行後、クラス・エディタが更新され、新規のメソッド定義が表示されます。このメソッド定義は、クラス・エディタとクラス・インスペクタのどちらを使用しても編集できます。以下に例を示します。

Class Definition

```
/// This is a Person class
class MyApp.Person extends %Persistent
{
Method Print() As %Boolean
{
    Write "Hello"
    Quit 1
}
}
```

4.2 メソッドのオーバーライド

注釈　**[リファクタ]** サブメニューは、スタジオが Windows サーバに接続されている場合にのみ利用できます。**[オーバーライド]** メニューは、他のプラットフォームで使用できます。

オブジェクト・ベース開発の優れた特徴の 1 つは、クラスがそのスーパークラスからメソッドを継承できる点にあります。また、スーパークラスから継承されたメソッドをオーバーライドする (つまり、新しい実装を作成する) こともできます。

[クラス]→[リファクタ]→[オーバーライド] を選択すると、現在のクラスでオーバーライド可能なスーパークラスで定義されているすべてのメソッドが一覧表示されるため、クラス・メソッドが簡単にオーバーライドされます。

例えば、永続クラスにおいて、クラスのインスタンスが保存されるときのカスタム検証を指定する場合、**%RegisteredObject** クラスの %OnValidateObject メソッドの既定の実装をオーバーライドすることがあります。

以下の手順を実行します。

1. スタジオで、永続クラス定義をオープン (または生成) します。
2. **[クラス]→[リファクタ]→[オーバーライド]** を選択し、**[メソッド]** タブを選択します。ダイアログ・ウィンドウが開き、オーバーライドするメソッドのリストが表示されます。
3. リストから [%OnValidateObject] を選択し、**[OK]** ボタンを選択します。

クラス定義には、次のような %OnValidateObject メソッドの定義が追加されます。

Class Definition

```
class MyApp.Person extends %Persistent
{
// ...
Method %OnValidateObject() As %Status
{
}
}
```

これで、クラス・エディタを使用して、メソッドの本文にコードを追加することができます。

5

クラスへのクラス・パラメータの追加

クラス・パラメータは、特定のクラスのすべてのオブジェクトに対する定数値を定義します。このクラス・パラメータの値は、クラス定義を作成するときに(またはコンパイル前の任意の時点で)設定できます。既定では、各パラメータの値はNULL文字列です。パラメータの値を設定するには、それに対する値を明示的に提供する必要があります。コンパイル時に、パラメータの値はクラスのすべてのインスタンスに対して構築されます。この値は、実行時には変更できません。

クラス・パラメータをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用してクラス定義にクラス・パラメータを追加する方法
- 新規クラス・パラメータ・ウィザードを使用する方法

クラス・エディタを使用してパラメータを追加するには、クラス・エディタの空の行にカーソルを置き、次のようにクラス・パラメータを入力します。

Class Member

```
Parameter P1 = "x";
```

パラメータの定義の詳細は、"クラス・パラメータの定義と参照" を参照してください。"クラス定義リファレンス" の "パラメータ定義" も参照してください。

5.1 新規クラス・パラメータ・ウィザード

新規クラス・パラメータ・ウィザードを使用して、新規のクラス・パラメータを作成することができます。**[クラス]→[追加]→[クラス・パラメータ]** を選択して、新規クラス・パラメータ・ウィザードを開くことができます。または、クラス・インスペクタで右クリックして、**[追加]→[新規クラス・パラメータ]** を選択します。ツールバーの **[新規クラス・パラメータ]** ボタン  をクリックすることもできます。

新規クラス・パラメータ・ウィザードでは、情報の入力が促されます。**[完了]** ボタンはいつでも選択できます。指定していない情報に対しては既定値が設定されます。

新規クラス・パラメータ・ウィザードでは、以下の情報の入力が促されます (これらの値は後で変更できます)。

名前

(必須) クラス・パラメータの名前。これは、有効なパラメータ名である必要があります。また、定義済みの既存のパラメータと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規クラス・パラメータに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントで表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

既定値

クラス・パラメータの既定値。この値は、このクラスのすべてのインスタンスにおいて、このパラメータの既定値になります。

6

クラスへのリレーションシップの追加

リレーションシップは、2 つ以上のオブジェクト・インスタンスの相互の関連性を定義する、特別なプロパティのタイプです。例えば、企業を表す Company クラスが、社員を表す Employee クラスとの一対多のリレーションシップを持っているという場合です (つまり、1 つの企業には 1 人、または複数の社員が在籍し、各社員は特定の企業と関連しているということです)。

リレーションシップは、プロパティとは異なります。すべてのリレーションシップは 2 つで 1 組みです。リレーションシップのすべての定義には、対応する逆のリレーションシップがあり、それが反対側を定義します。

リレーションシップの詳細は、"リレーションシップ" を参照してください。

新規のリレーションシップをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用してクラス定義にリレーションシップを追加する方法
- 新規プロパティ・ウィザードを使用する方法

クラス・エディタを使用してリレーションシップを追加するには、クラス・エディタの空の行にカーソルを置き、リレーションシップ宣言を入力します。

Class Definition

```
Class MyApp.Company Extends %Persistent
{
Relationship TheEmployees As Employee [cardinality=many, inverse=TheCompany];
}
```

リレーションシップ定義では、cardinality と inverse の両方のキーワードに値を指定する必要があります。

このリレーションシップには 2 つの双方向の関係があるので、Employee のクラス定義には逆のリレーションシップも入力します。

Class Definition

```
Class MyApp.Employee Extends %Persistent
{
Relationship TheCompany As Company [cardinality=one, inverse=TheEmployees];
}
```

両方のリレーションシップが正しく相互に対応していない場合、コンパイル時にエラーが返されます。

6.1 リレーションシップ・プロパティを作成する新規プロパティ・ウィザード

新規プロパティ・ウィザードを使用して、新規のリレーションシップ・プロパティを作成できます。このウィザードを起動するには、**[クラス]→[追加]→[プロパティ]** を選択します。または、クラス・インスペクタで右クリックして、**[追加]→[新規プロパティ]** を選択します。ツールバーの **[新規プロパティ]** ボタンをクリックすることもできます。

新規プロパティ・ウィザードでは、情報の入力が促されます。この手順は、プロパティ・タイプで　　を選択することを除けば、新規の非リレーションシップ・プロパティを作成する手順と同じです。

6.1.1 名前および説明ページ

新規プロパティ・ウィザードで、以下の情報の入力が促されます (これらの値は後で変更できます)。

プロパティ名

(必須) リレーションシップの名前。これは、有効なリレーションシップ (プロパティ) 名である必要があります。また、定義済みの既存のリレーションシップやプロパティと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規リレーションシップに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントで表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

6.1.2 プロパティ・タイプ・ページ

新規プロパティ・ウィザードでは、各種のプロパティ・タイプが表示されます。　　を選択し、もう一方のリレーションシップのクラス名を入力します。

6.1.3 リレーションシップの特性ページ

新規プロパティ・ウィザードでは、追加のリレーションシップ属性の入力が促されます。これには、以下のものがあります。

`Cardinality`

一 : 他方の 1 つのオブジェクト

このリレーションシップ・プロパティは、1 つの関連するオブジェクトのインスタンスを参照します。結果のプロパティは、単純な参照フィールドとして動作します。

多 : 他方の複数のオブジェクト

このリレーションシップ・プロパティは、複数の関連するオブジェクトのインスタンスを参照します。結果のプロパティは、オブジェクトのコレクションとして動作します。

親 : このオブジェクトの親

のカーディナリティと同じですが、このプロパティは、依存したリレーションシップであり、このオブジェクトの親を参照する点が異なります。親子リレーションシップを作成すると、子は個別に格納されるのではなく、親内に格納されるため、インデックスの作成を選択できません。これは、グローバル構造を見ることで理解できます。子には、追加の添え字を作成することにより、自動的にインデックスが付けられます。

子 : このオブジェクトの子

のカーディナリティと同じですが、このプロパティは、依存したリレーションシップであり、子オブジェクトのコレクションを参照する点が異なります。

`Inverse`

このリレーションシップ・プロパティは以下のタイプのオブジェクトを参照します。

[参照] ボタンをクリックしてクラスを選択するか、逆リレーションシップの新規クラス名を入力します。

参照されたクラス内の対応するプロパティの名前は、以下のとおりです。

クラスからプロパティを選択するか、逆リレーションシップの新規プロパティ名を入力します。

6.1.4 その他の変更

実装するその他の変更を選択します。

新規クラス <inverse class> の作成

このフィールドは、ウィザードの前のページで指定したクラス名が存在しない場合にのみ有効です。新しいクラスをパッケージに追加するには、このオプションを選択します。リレーションシップが含まれたクラスをコンパイルするには、その前に新規クラスをコンパイルする必要があります。

クラス <inverse class> での新規プロパティ Parent の作成

このフィールドは、ウィザードの前のページで指定した参照されるクラスのプロパティが存在しない場合にのみ有効です。この新しいプロパティをクラスに追加するには、このオプションを選択します。リレーションシップが含まれたクラスをコンパイルするには、その前にこの新規プロパティが含まれたクラスをコンパイルする必要があります。

クラス <inverse class> のプロパティ Parent の変更

このフィールドは、参照されるクラスのプロパティが既に存在する場合にのみ有効です。このオプションを選択すると、参照されるプロパティが、定義しているプロパティとリレーションシップを持つように変更されます。

このリレーションシップのインデックスを定義します。

選択すると、このプロパティのインデックスを定義できます。これは、一対多リレーションシップにのみ適用されます。親子リレーションシップでは無効になります。

6.1.5 新規プロパティ・ウィザードで新規のリレーションシップを作成した結果

新規プロパティ・ウィザードで新規リレーションシップを作成した後、クラス・エディタは更新され、新規のリレーションシップ定義が表示されます。以下に例を示します。

Class Definition

```
/// This is an Employee class
class MyApp.Employee extends %Persistent
{

/// We have a one-to-many relationship with Company
Relationship Company As Company [cardinality=one, inverse=Employees];
}
```

このリレーションシップをさらに変更する場合は、クラス・エディタ、またはクラス・インスペクタを使用します。

また、リレーションシップ・ウィザードを使用すると、もう一方のリレーションシップに必要な変更を自動的に決定することができます。

リレーションシップ・ウィザードを開くには:

1. クラス・インスペクタで、プロパティのリストを表示します。
2. プロパティ・リスト内の目的のリレーションシップを右クリックし、ポップアップ・メニューから [**リレーションシップの追加/変更**] を選択します。

7

クラスへのクエリの追加

InterSystems IRIS® クラス定義には、クエリ定義が含まれる場合があります。

概要については、"クラス・クエリの定義と使用" を参照してください。詳細は、"クラス定義リファレンス" の "クエリ定義" を参照してください。

新規のクエリをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用して、クラス定義を編集する方法
- 新規クエリ・ウィザードを使用します。

クラス・エディタを使用してクエリを追加するには、クラス・エディタの空の行にカーソルを配置し、次のようにクエリ宣言を入力します。

Class Definition

```
Class MyApp.Person Extends %Persistent
{
Property Name As %String;

/// This query provides a list of persons ordered by Name.
Query ByName(ByVal name As %String) As %SQLQuery(CONTAINID = 1)
{
    SELECT ID,Name FROM Person
    WHERE (Name %STARTSWITH :name)
    ORDER BY Name
}

}
```

また、既存のクエリ宣言をコピーして貼り付け、それを編集することもできます。

7.1 新規クエリ・ウィザード

新規クエリ・ウィザードを使用して、新規クエリをクラス定義に追加できます。**[クラス]→[追加]→[クエリ]** を選択して、新規クエリ・ウィザードを開くことができます。または、クラス・インスペクタで右クリックして、**[追加]→[新規クエリ]** を選択します。ツールバーの **[新規クエリ]** ボタン  をクリックすることもできます。

[完了] ボタンはいつでも選択できます。指定していない情報に対しては既定値が設定されます。

7.1.1 名前、実装、および説明ページ

新規クエリ・ウィザードでは、以下の情報の入力が促されます (これらの値は後で変更できます)。

クエリ名

(必須) 新規クエリの名前。これは、有効なクエリ名である必要があります。また、定義済みの既存のクエリと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

実装

(必須) これが SQL 文を基にしたクエリ (ウィザードで生成されたクエリ) であるか、ユーザ記述のコードを基にしたクエリ (ユーザが自らクエリ実装のコードを記述したクエリ) であるかを指定します。

説明

(オプション) 新規クエリに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントで表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

7.1.2 入力パラメータ・ページ

クエリは、入力パラメータ (引数) を取ります (入力パラメータを必要としないこともあります)。

これらのパラメータの名前、型、および既定値を指定することができます。引数は、表形式で表示されます。テーブルにある **[追加]** アイコン  を使用して、引数リストに新規の項目を追加することができます。このアイコンをクリックすると、ポップアップ・ダイアログが表示され、引数の名前、そのタイプ、オプションの既定値を指定できます。上矢印 と下矢印  を使用すると、リスト内の項目の順序を並べ替えることができます。

7.1.3 列ページ

SQL ベースのクエリでは、結果セットに含める (または、生成された SQL クエリの SELECT 句によって) オブジェクト・プロパティ (列) を指定する必要があります。

クエリに列を追加するには、利用できるプロパティリストの左側から項目を選択し、[>] ボタンを使用してその項目を右側のリストに移動します。プロパティをダブルクリックして移動することもできます。

7.1.4 条件ページ

SQL ベースのクエリでは、結果セットを制限する (または、生成された SQL クエリの SQL WHERE 句によって) 条件を指定できます。

条件を設定するには、一連のコンボ・ボックスから値を選択します。[表現] ボックスには、表現 (リテラル値など) やクエリ引数 (: コロン文字で始まる SQL ホスト変数など) を指定できます。

7.1.5 照合順ページ

SQL ベースのクエリでは、結果セットの並べ替え (生成された SQL クエリの SQL ORDER BY 句) に使用する列を指定できます。

7.1.6 行指定形式ページ

ユーザ記述のクエリでは、クエリによって返される列の名前やタイプを指定する必要があります。

SQL ベースのクエリの場合、ウィザードではこの情報の入力は促されません。クラス・コンパイラが、SQL クエリを検証することで決定できるためです。

7.1.7 新規クエリ・ウィザード実行の結果

新規クエリ・ウィザードを実行した後、クラス・エディタ・ウィンドウは更新され、新規のクエリ定義が表示されます。以下に例を示します。

Class Definition

```
/// This is a Person class
class MyApp.Person extends %Persistent
{

Query ByName(ByVal name As %String) As %SQLQuery(CONTAINID = 1)
{
    SELECT ID,Name FROM Person
    WHERE (Name %STARTSWITH :name)
    ORDER BY Name
}

}
```

このクエリをさらに変更する場合は、クラス・エディタ、またはクラス・インスペクタを使用します。

ユーザ記述のクエリを指定した場合、クラス・エディタには新規のクエリ定義と、実装しようとしているクエリ・メソッドのスケルトンの両方が追加されます。以下に例を示します。

Class Definition

```
Class MyApp.Person Extends %Persistent
{
// ...

ClassMethod MyQueryClose(
        ByRef qHandle As %Binary
        ) As %Status [ PlaceAfter = MyQueryExecute ]
{
    Quit $$$OK
}

ClassMethod MyQueryExecute(
        ByRef qHandle As %Binary,
        ByVal aaa As %Library.String
        ) As %Status
{
     Quit $$$OK
}

ClassMethod MyQueryFetch(
        ByRef qHandle As %Binary,
        ByRef Row As %List,
        ByRef AtEnd As %Integer = 0
        ) As %Status [ PlaceAfter = MyQueryExecute ]
{
    Quit $$$OK
}

Query MyQuery(
        ByVal aaa As %Library.String
        ) As %Query(ROWSPEC = "C1,C2")
{
}

}
```

8

クラスへのインデックスの追加

ここでは、永続クラス定義にインデックス定義を追加する方法と編集する方法を説明します。

インデックス定義では、InterSystems IRIS® クラス・コンパイラに 1 つまたは複数のプロパティのインデックスを作成するように指定します。インデックスは通常、SQL クエリの効率を向上させるために使用されます。"SQL 最適化ガイド" の "インデックスの定義と構築" を参照してください。

インデックスをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用して、クラス定義を編集する方法
- 新規インデックス・ウィザードを使用する方法

クラス・エディタを使用してインデックスを追加するには、クラス・エディタの空の行にカーソルを置き、インデックス宣言を入力します。

Class Member

```
Index NameIndex On Name;
```

また、既存のインデックス宣言をコピーして貼り付け、それを編集することもできます。

詳細は、"クラス定義リファレンス" の "インデックス定義" を参照してください。

8.1 新規インデックス・ウィザード

[クラス]→[追加]→[インデックス] を使用して、新規インデックス・ウィザードを起動します。または、クラス・インスペクタで右クリックして、**[追加]→[新規インデックス]** を選択します。ツールバーの **[新規インデックス]** ボタン  をクリックすることもできます。

新規インデックス・ウィザードでは、情報の入力が促されます。すべてのウィザード・ページを完了する前に **[完了]** をクリックした場合、指定していない情報には既定値が設定されます。

8.1.1 名前および説明ページ

新規インデックス・ウィザードでは、以下の情報の入力が促されます (これらの値は後で変更できます)。

インデックス名

(必須) 新規インデックスの名前。これは、有効なインデックス名である必要があります。定義済みの既存のインデックスと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規インデックスに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントで表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

8.1.2 インデックス・タイプ・ページ

InterSystems IRIS は以下のインデックス・タイプをサポートしています。

通常インデックス

通常インデックスは、プロパティ値のインデックス作成に使用します。以下のオプションの 1 つを選択することで、通常インデックスを詳細に定義できます。

パラメータ	説明
ユニーク・インデックス	このインデックスに関連する一連のプロパティは、このクラスのオブジェクトのエクステント内で一意である、結合された値を持つ必要があります。
IDKEY	このインデックスに関連する一連のプロパティは、オブジェクト ID の作成に使用します。オブジェクト ID は、データベースにこのクラスのインスタンスを保存するのに使用されます。オブジェクトが一度保存された後は、IDKEY 定義の一部であるプロパティ値を変更することはできません。IDKEY は、プロパティが一意であることを意味しています (ユニーク・インデックスと同様)。
SQL 主キー	このインデックスに関連する一連のプロパティは、このクラスに投影された SQL テーブルの SQL 主キーとして報告されます。SQL 主キーは、プロパティが一意であることを意味しています (ユニーク・インデックスと同様)。

エクステント・インデックス

エクステント・インデックスは、オブジェクトの複数のクラス・エクステント内で、特定のクラスに属しているオブジェクトを記録するために使用します。これは、追加の属性を指定できない点で、通常インデックスとは異なります。

データベース内でのインデックスの物理的な実装方法を選択することもできます。

標準インデックス

このインデックスは、指定されたプロパティでの従来のクロス・インデックスです。

ビットマップ・インデックス

ビットマップ・インデックスは、指定したインデックス値に対応する一連のオブジェクト ID 値の圧縮表現を使用します。詳細は、"InterSystems SQL 最適化ガイド" の "ビットマップ・インデックス" を参照してください。

ビットスライス・インデックス

ビットスライス・インデックスは、合計や値域条件など、特定の式について非常に高速の評価を有効にする特殊な形式のインデックスです。詳細は、"InterSystems SQL 最適化ガイド" の "ビットスライス・インデックス" を参照してください。

8.1.3 インデックス・プロパティ・ページ

インデックス・プロパティのページでは、インデックスを基にした 1 つ、または複数のプロパティのリストを入力できます。各プロパティで、既定の照合関数をオーバーライドできます。既定の照合関数は、照合関数のすべてのパラメータと、インデックス内に保存された値を変換するために使用されます。

重要　IDKEY インデックスによって使用されるどのプロパティの値においても、そのプロパティが永続クラスのインスタンスへの有効な参照でない限り、連続する 2 つの垂直バー (||) は使用しないでください。この制限は、InterSystems IRIS SQL のメカニズムが動作するための方法に起因しています。IDKey プロパティ で || を使用すると、予測できない動作が起こる可能性があります。

8.1.4 インデックス・データ・ページ

インデックス・データ・ページでは、プロパティのデータのコピーをインデックスに保存するかどうかを指定します。

ビットマップ・インデックスでは、データ値のコピーを保存することはできません。

8.1.5 新規インデックス・ウィザード実行の結果

新規インデックス・ウィザードの実行後、クラス・エディタは更新され、新規のインデックス定義が表示されます。以下に例を示します。

Class Definition

```
/// This is a Person class
class MyApp.Person extends %Persistent
{

Property Name As %String;

Index NameIndex On Name;
}
```

インデックス定義を編集するには、クラス・エディタまたはクラス・インスペクタを使用できます。

8.2 インデックスの配置

インデックス定義をクラスに追加し、それをコンパイルした後、管理ポータルの **[インデックス再構築]** を使用して、インデックスを生成 (データを挿入) できます。詳細は、"InterSystems SQL 最適化ガイド" の "インデックスの構築" を参照してください。

スタジオでは、既存のデータのインデックスを自動的に生成しません。

9

クラスへのプロジェクションの追加

ここでは、クラス定義にプロジェクション定義を追加する方法を説明します。

プロジェクション定義では、クラス定義がコンパイルされたとき、または削除されたときに、InterSystems IRIS® クラス・コンパイラが指定された操作を実行するように指示します。プロジェクションは、(%Projection.AbstractProjection クラスから派生した) プロジェクション・クラスの名前を定義します。プロジェクション・クラスは、以下のいずれかが当てはまる場合に呼び出されるメソッドを実装します。

- クラスのコンパイルが終了した。
- クラス定義の削除か、クラスの再コンパイルのいずれかにより、クラス定義が削除される。

クラスには、任意の数のプロジェクション定義を指定できます。これらのすべてのアクションは、クラスがコンパイルされるときに呼び出されます (呼び出される順序は定義されていません)。

InterSystems IRIS には、Java や MV などからクラスにアクセスするためのクライアント・コードを生成する、事前定義されたプロジェクション・クラスが含まれています。

テーブル 9-1: プロジェクション・クラス

クラス	説明
%Projection.Java	Java からのクラスへのアクセスを可能にする、Java クライアント・クラスを生成します。
%Projection.Monitor	このクラスを、ログ・モニタで使用するルーチンとして登録します。メタデータが Monitor.Application、Monitor.Alert、Monitor.Item、および Monitor.ItemGroup に書き込まれます。Monitor.Sample という新規の永続クラスが作成されます。
%Projection.MV	MV からクラスへのアクセスを可能にする MV クラスを生成します。
%Projection.StudioDocument	このクラスをスタジオで使用するルーチンとして登録します。
%Studio.Extension.Projection	XData 'メニュー' ブロックをメニュー・テーブルに投影します。

また、独自のプロジェクション・クラスを生成し、組み込みのプロジェクション・クラスと同様に、スタジオで使用することもできます。

新規のプロジェクションをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用して、クラス定義を編集する方法
- 新規プロジェクション・ウィザードを使用する方法

クラス・エディタを使用してプロジェクションを追加するには、クラス・エディタの空の行にカーソルを置き、プロジェクション宣言を入力します。

また、既存のプロジェクション宣言をコピーして貼り付け、それを編集することもできます。

詳細は、"クラス定義リファレンス" の "プロジェクション定義" を参照してください。

9.1 新規プロジェクション・ウィザード

[クラス]→[追加]→[プロジェクション] を選択して、新規プロジェクション・ウィザードを起動できます。または、クラス・インスペクタで右クリックし、**[追加]→[新規プロジェクション]** を選択します。

[新規プロジェクション・ウィザード] では、複数のページが表示され、新規プロジェクションに関する情報の入力が促されます。すべてのウィザード・ページを完了する前に **[完了]** をクリックできます。この場合、指定していない情報には既定値が設定されます。

9.1.1 名前および説明ページ

新規プロジェクション・ウィザードで、以下の情報の入力が促されます (これらの値は後で変更できます)。

プロジェクション名

(必須) 新規プロジェクションの名前。これは、有効なプロジェクション名である必要があります。また、定義済みの既存のプロジェクションと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規プロジェクションに関する説明。

9.1.2 プロジェクション・タイプ・ページ

プロジェクトのタイプでは、クラス定義がコンパイルまたは削除されたときのアクションを決定します。定義するプロジェクションの種類を選択できます。

プロジェクション・タイプ

プロジェクション・クラスの名前。このクラスのメソッドは、クラス定義がコンパイル、または削除されるときに実行されます。

プロジェクション・パラメータ

プロジェクション・クラスの動作を制御する名前と値の組み合わせです。利用できるパラメータ名のリストは、選択されたプロジェクション・クラスによって決定されます。

9.1.3 新規プロジェクション・ウィザード実行の結果

新規プロジェクション・ウィザードの実行後、クラス・エディタ・ウィンドウは更新され、新規のプロジェクション定義が表示されます。以下に例を示します。

Class Definition

```
/// This is a Person class
class MyApp.Person extends %Persistent
{

Property Name As %String;

Projection JavaClient As %Projection.Java;
}
```

このプロジェクション定義を編集するには、クラス・エディタまたはクラス・インスペクタを使用します。

10

クラスへの XData ブロックの追加

XData ブロックは、クラス定義に追加できる XML コードのブロックです。

XData ブロックをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用して、クラス定義を編集する方法
- XData ウィザードを使用する方法

クラス・エディタを使用して XData ブロックを追加するには、クラス・エディタの空の行にカーソルを置き、次のようにXData 宣言を入力します。

```
XData ProductionDefinition
    {
    <Production>
    <ActorPoolSize2/ActorPoolSize>
    </Production>
    }
```

また、既存の XData ブロックをコピーして貼り付け、それを編集することもできます。

10.1 新規 XData ウィザード

[クラス]→[追加]→[XData] を使用して、新規 XData ウィザードを起動できます。

新規 XData ウィザードでは、1 つのページが表示され、XData ブロックの名前と説明を入力するよう促されます。終了するには、**[完了]** を選択します。XML コードをクラス・エディタ・ウィンドウに追加して、XData ブロックを完了します。

新規 XData ウィザードでは、以下の情報の入力が促されます (これらの値は後で変更できます)。

XData 名

(必須) 新規 XData の名前。これは、有効な名前でなければなりません。また、定義済みの既存の XData と同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規 XData に関する説明。

詳細は、"クラス定義リファレンス" の "XData 定義" を参照してください。

11

クラスへの SQL トリガと外部キーの追加

InterSystems IRIS® のすべての永続クラスは、SQL テーブルとして自動的に投影されます。ここでは、SQL の振る舞いを制御するクラス定義の部分を、スタジオで操作する方法について説明します。

"トリガの使用法" および "外部キーの使用法" も参照してください。詳細は、"クラス定義リファレンス" の "トリガ定義" および "外部キー定義" を参照してください。

11.1 SQL エイリアス

クラスや大部分のクラス・メンバに、SQL での使用に対して異なる名前を与えることができます。これは、以下のような理由により便利です。

- SQL には、識別子として使用できない多数の予約語があるため。
- InterSystems IRIS では、クラス名やクラス・メンバ名に下線文字がサポートされていないため。

クラスに対する SQL テーブル名を指定するには、クラス・インスペクタでクラス情報を表示し、`SqlTableName` キーワードの値を編集します。

クラス・メンバに対する SQL 名を指定するには、クラス・インスペクタで希望のプロパティを選択し、対応する SQL 名キーワードの値を編集します (プロパティには `SqlFieldName`、インデックスには `SqlName` など)。

11.2 SQL ストアド・プロシージャ

SQL ストアド・プロシージャは、InterSystems IRIS メソッドやクラス・クエリで、ストアド・プロシージャとして ODBC や JDBC クライアントによって呼び出すことができます。

InterSystems IRIS は、2 つのスタイルの SQL ストアド・プロシージャをサポートしています。

- クラス・クエリに基づくプロシージャ。
- クラス・メソッドに基づいて結果セットを返さないプロシージャ。

11.2.1 クエリ・ベースのストアド・プロシージャ

結果セットを返す SQL ストアド・プロシージャを作成するには、クラス定義にクエリ定義を追加し、クエリの SqlProc キーワードを True に設定します。以下の操作を行います。

1. [クラス]→[追加]→[クエリ] を選択して新規クエリ・ウィザードを開きます。
2. このウィザードを完了して、新規クエリをクラス定義に追加します。
3. クラス・インスペクタを使用して、クエリ定義キーワード SqlProc の値を `True` に設定します。

この結果は、以下のようになります。

Class Definition

```
Class Employee Extends %Persistent
{

/// A class query listing employees by name.
Query ListEmployees() As %SQLQuery(CONTAINID = "1") [SqlProc]
{
    SELECT ID,Name
    FROM Employee
    ORDER BY Name
}
}
```

CALL 文を使用して、このストアド・プロシージャを ODBC、または JDBC クライアントから呼び出すことができます。

```
CALL Employee_ListEmployees()
```

この呼び出しの後、ODBC アプリケーション、または JDBC アプリケーションは、クラス・クエリから返された結果セットのコンテンツをフェッチできます。

この方法は、カスタム記述コードを基にしたクエリ定義でも使用できます。SQL 文のみを基にしたストアド・プロシージャの定義に限定されません。

11.2.2 メソッド・ベースのストアド・プロシージャの作成

結果セットを返さない SQL ストアド・プロシージャを作成するには、クラス定義にクエリ・メソッドを追加し、メソッドの SqlProc キーワードを `True` に設定します。以下の操作を行います。

1. 新規メソッド・ウィザードを使用して、クラス定義内にクラス・メソッドを作成します。
2. クラス・インスペクタを使用して、メソッドのキーワード SqlProc の値を `True` に設定します。

この結果は、以下のようになります。

Class Definition

```
Class Employee Extends %Persistent
{

  ClassMethod Authenticate(
        ctx As %SQLProcContext,
        name As %String,
        ByRef approval As %Integer
        ) [SqlProc]
  {
    // ...
    Quit
  }

}
```

SQL ストアド・プロシージャとして使用されたメソッドの最初の引数は、%SQLProcContext オブジェクトのインスタンスであることに注意してください。詳細は、"ストアド・プロシージャの定義と使用" を参照してください。

CALL 文を使用して、このストアド・プロシージャを ODBC クライアントから呼び出すことができます。

```
CALL Employee_Authenticate('Elvis')
```

このストアド・プロシージャを JDBC クライアントから呼び出すには、以下のコードを使用します。

```
prepareCall("{? = call Employee_Authenticate(?)}")
```

11.3 クラスへの SQL トリガの追加

SQL トリガは、特定のイベントに応答して、SQL エンジンによって起動されるコードです。

SQL トリガは、オブジェクトの永続性が保持されている間は起動されません (**%Storage.SQL** ストレージ・クラスを使用していない場合)。

SQL トリガをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用して、クラス定義の値を編集する方法
- 新規 SQL トリガ・ウィザードを使用する方法

クラス・エディタを使用して SQL トリガを追加するには、クラス・エディタの空の行にカーソルを置き、次のようにトリガ宣言を入力します。

Class Definition

```
Class MyApp.Company Extends %Persistent
{

/// This trigger updates the Log table for every insert
Trigger LogEvent [ Event = INSERT ]
{
    // ...
}

}
```

11.3.1 新規 SQL トリガ・ウィザード

新規 SQL トリガ・ウィザード を使用して、新規の SQL トリガを作成できます。新規 SQL トリガ・ウィザードを開くには、**[クラス]→[追加]→[SQLトリガ]** を使用します。または、クラス・インスペクタで右クリックし、**[追加]→[新規 SQL トリガ]** を選択します。

新規 SQL トリガ・ウィザードで、情報の入力が促されます。**[完了]** ボタンはいつでも選択できます。指定していない情報に対しては既定値が設定されます。

11.3.1.1 名前および説明ページ

新規 SQL トリガ・ウィザードでは、以下の情報に関する入力が促されます (これらの値は後で変更できます)。

トリガ名

(必須)トリガの名前。これは、有効なトリガ名である必要があります。また、定義済みの既存のトリガと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規トリガに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントで表示されるときに使用されます。

11.3.1.2 トリガ・イベント・ページ

新規の SQL トリガ・ウィザードでは、イベントやトリガのタイミングを指定して、いつ新規のトリガを起動するかを示します。

イベント・タイプ

トリガが起動される SQL イベントを指定します。選択肢は、[ ] (新規の行が挿入されるとき)、[ ] (行が更新されるとき)、または [ ] (行が削除されるとき) です。

イベントのタイミング

いつトリガが起動されるかを指定します。選択肢は、イベントが発生する 、または です。

11.3.1.3 トリガ・コード

新規 SQL トリガ・ウィザードでは、トリガに対するソース・コードを入力できます (必要な場合)。

11.3.1.4 新規 SQL トリガ・ウィザード実行の結果

新規 SQL トリガ・ウィザードの実行後、クラス・エディタは更新されて、新規のトリガ定義が表示されます。

このトリガは、クラス・エディタとクラス・インスペクタのどちらを使用しても編集できます。

11.4 クラスへの新規 SQL 外部キーの追加

SQL 外部キーは、テーブル内の 1 つまたは複数のフィールドと、他のテーブルのキー (一意のインデックス) 間の整合性制約を定義します。

一般的に、オブジェクト・アプリケーションでは外部キーを使用しません。その代わり、オブジェクト・ベースでの操作に優れたリレーションシップを使用します。リレーションシップは、手動で定義する外部キー定義と同等の整合性制約を (SQL とオブジェクト・アクセスの両方に) 自動的に適用します。

アプリケーションでは、通常、本質的にリレーショナルなアプリケーションの外部キー定義を使用します。

SQL 外部キーをクラス定義に追加するには、以下の 2 つの方法があります。

- クラス・エディタを使用して、クラス定義の値を編集する方法
- 新規 SQL 外部キー・ウィザードを使用する方法

クラス・エディタを使用して SQL 外部キーを追加するには、クラス・エディタの空の行にカーソルを置き、次のように外部キー宣言を入力します。

Class Definition

```
Class MyApp.Company Extends %Persistent
{

Property State As %String;

ForeignKey StateFKey(State) References StateTable(StateKey);

}
```

11.4.1 新規 SQL 外部キー・ウィザード

新規 SQL 外部キー・ウィザードを開くには、**[クラス]→[追加]→[外部キー]** を使用します。または、クラス・インスペクタで右クリックして、**[追加]→[新規外部キー]** を選択することもできます。ツールバーの **[新規外部キー]** ボタン  をクリックすることもできます。

新規 SQL 外部キー・ウィザードでは、情報の入力が促されます。必要な情報をすべて入力したら、**[完了]** ボタンを選択します。指定していない情報には既定値が設定されます。

11.4.1.1 名前および説明ページ

新規 SQL 外部キー・ウィザードでは、以下の情報の入力が必要です (これらの値は後で変更できます)。

外部キー名

(必須) 外部キーの名前。これは、有効な外部キー名である必要があります。また、定義済みの既存の外部キーと同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

説明

(オプション) 新規外部キーに関する説明。この説明は、クラス・ドキュメントがオンライン・クラス・ライブラリ・ドキュメントで表示されるときに使用されます。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

11.4.1.2 属性ページ

ウィザードの 2 番目のページで、外部キーによって制約するクラスの 1 つ以上のプロパティを選択します。

11.4.1.3 キー構成ページ

ウィザードの 3 番目のページで、外部キー・プロパティの制約に使用する値を指定する、クラスとクラス内のキー (一意のインデックス) の両方を選択します。

11.4.1.4 新規 SQL 外部キー・ウィザード実行の結果

新規 SQL 外部キー・ウィザードの実行後、クラス・エディタは更新され、新規の外部キー定義が表示されます。

この外部キーの変更は、クラス・エディタとクラス・インスペクタのどちらを使用しても行えます。

12

クラスへのストレージ定義の追加

永続クラスまたはシリアル・クラスで使用する物理的ストレージは、ストレージ定義で指定します。スタジオを使用すると、このようなストレージ定義を表示し、編集することができます。

注釈　ストレージ定義は、InterSystems IRIS® オブジェクトのかなり高度な機能です。多くの場合は、ストレージ定義を操作する必要はありません。InterSystems IRIS クラス・コンパイラでは、永続オブジェクトのストレージ定義が自動的に作成され、管理されます。

ストレージ定義は、一般に以下のような場合に使用します。

- パフォーマンスの調整など、永続オブジェクトで使用するストレージの詳細な制御が必要な場合
- 既存のデータ構造の最上部に、オブジェクト定義をマッピングする場合

クラスには任意の数のストレージ定義を持つことができますが、一度に 1 つしか使用できません。新規のクラスは、クラスを保存しコンパイルするまで、または明示的にクラスにストレージ定義を追加するまでは、ストレージ定義を持つことはありません。**[クラス]→[追加]→[ストレージ]** を使用すると、クラスに新規のストレージ定義を追加できます。

注釈　クラスをコンパイルすると、そのストレージ定義が自動的に生成されます。永続クラスとシリアル・クラスだけに、ストレージ定義を作成できます。

スタジオでは、以下の 2 つの方法でクラスのストレージ定義を表示し、編集できます。

- 視覚的に行う方法。クラス・インスペクタ・ウィンドウのストレージ・インスペクタを使用します。クラス・インスペクタで [ ] を選択し、目的のストレージ定義を選択します。
- テキストを使用する方法。クラス・エディタを使用します。ストレージ定義はクラス定義の本文にあります。

これらの方法は、この後のセクションで詳細に説明します。

12.1 クラスへのストレージ定義の追加

クラス定義に新規のストレージ定義を追加するには、以下の 2 つの方法があります。

- クラス・エディタおよび **[クラス]→[追加]→[ストレージ]** を使用して、クラス定義にストレージ定義を追加する方法
- 新規ストレージ・ウィザードを使用する方法

12.1.1 新規ストレージ・ウィザードを使用する方法

新規ストレージ・ウィザードを使用して、新規ストレージをクラス定義に追加できます。新規ストレージ・ウィザードを起動するには、**[クラス]→[追加]→[ストレージ]** を使用します。または、クラス・インスペクタで右クリックし、**[追加]→[新規ストレージ]** を選択します。ツールバーの **[新規ストレージ]** ボタン  をクリックすることもできます。

新規ストレージ・ウィザードでは、情報の入力が促されます。**[完了]** ボタンはいつでも選択できます。指定していない情報には既定値が設定されます。

12.1.1.1 名前、タイプ、説明ページ

新規ストレージ・ウィザードでは、以下の情報の入力が促されます (これらの値は後で変更できます)。

ストレージ名

(必須) 新規ストレージ定義の名前。これは、有効なクラス・メンバ名である必要があります。また、定義済みの既存のストレージ定義と同じ名前を付けることはできません。

"識別子のルールとガイドライン" を参照してください。

ストレージ・タイプ

(必須) ストレージ定義によって使用するストレージのタイプ。タイプは、どのストレージ・クラスによって、このクラスのストレージ・インタフェースを実装するかを指定します。以下が、その選択肢です。

- ストレージ – このストレージ定義は、**%Storage.Persistent** クラスをベースにします。これは、新規の永続クラスに使用される既定のストレージ・タイプです。
- SQL ストレージ – このストレージ定義は、**%Storage.SQL** クラスをベースにします。このストレージ・タイプは、ストレージ・オペレーションを実行する SQL 文を使用します。このストレージ・タイプは、オブジェクトを既存のデータ構造にマッピングするか、または InterSystems SQL ゲートウェイを経由してリモートの RDBMS と通信するのに使用します。
- カスタム・ストレージ – このストレージ定義は、ユーザ定義のストレージ・クラスをベースにします。

説明

(オプション) 新規ストレージ定義に関する説明。

説明には、HTML フォーマット・タグを記述することもできます。"クラスの定義と使用" の "クラス・ドキュメントの作成" を参照してください。

12.1.1.2 %Storage.Persistent 定義のグローバル特性ページ

%Storage.Persistent ストレージ定義の場合、新規ストレージ・ウィザードで、永続クラスのデータやインデックスを保存するのに使用するグローバル (永続多次元配列) の特性を指定します。ウィザードで指定する属性には、以下のものがあります。

DataLocation

グローバルの名前であり、永続クラスのインスタンスの保存に使用する文頭の添え字です。例えば、グローバル ^data にデータを保存するように指定するには、このフィールドに `^data` と入力します。グローバル・サブノード ^data("main") にデータを保存するように指定するには、`^data("main")` と入力します。

IndexLocation

グローバルの名前であり、永続クラスのインデックス・エントリの保存に使用する文頭の添え字です。既定では、インデックスはインデックス名を基にした追加の添え字と共に、インデックス・リファレンス・グローバルに保存されます。これは、インデックスごとに上書きできます。

IdLocation

グローバルの名前であり、既定のオブジェクト ID カウンタを含むのに使用する文頭の添え字です。オブジェクト ID カウンタは、このタイプの次のオブジェクト・インスタンスの ID 番号を保持するのに使用されます。

12.2 クラス・インスペクタをストレージ定義で使用する

クラス・インスペクタを使用して、クラスのストレージ定義を視覚的に表示したり編集したりできます。クラス・インスペクタには、メソッドやプロパティの表示と同様にストレージ定義のリストが表示されます。

クラス・インスペクタで既存のストレージ定義を表示する方法は、以下のとおりです。

1. クラス・インスペクタを選択します。
2. インスペクタのメンバ・リスト・プルダウンで [Storage] を選択します。
3. ストレージ定義をダブルクリックします。

これで、クラス・インスペクタにストレージ・キーワードとその値が表示されます。

ストレージ・キーワードの一部には、特別な注意が必要なものもあります。

Data Nodes

%Storage.Persistent ストレージ・クラスによって使用される一連のデータ・マッピング全体を表します。Data Nodes キーワードをダブルクリックして起動するデータ・ノード・エディタによって、ストレージ定義に対する一連のデータ・ノード仕様を表示し、編集することができます。つまり、クラスのプロパティがグローバル・ノードに保存される方法を直接指定できます。

SQL Storage Map

%Storage.SQL ストレージ・クラスによって使用される一連のデータ・マッピング全体を表します。SQL Storage Map キーワードをダブルクリックして起動する SQL ストレージ・マップ・エディタによって、オブジェクト・プロパティを既存のデータ構造にマップするのに使用される、一連のマッピングを表示し、編集することができます。

12.3 クラス・エディタをストレージ定義で使用する

クラス・エディタを使用して、クラスのストレージ定義を表示したり編集することができます。[ビュー]→[コードの展開] を使用すると、クラスのストレージ定義の表示を切り替えることができます。

ストレージ定義は、クラス定義の外部 XML 表示で使用されるのと同じ XML 要素を使用し、クラス定義の本文にインライン XML アイランドとして表示されます。

例えば、シンプルな MyApp.Person クラスを考慮してみましょう。

Class Definition

```
/// A simple persistent class
Class User.Person Extends %Persistent
{
Property Name As %String;
Property City As %String;
}
```

このクラスをコンパイルしたら、(クラス・コンパイラが、そのクラスに対するストレージ定義を生成したことを確認するため) **[ビュー]→[コードの展開]** を使用 (または、ブロックの最上行の隣にあるプラス・アイコンを選択) して、そのストレージ定義を表示します。これは、クラス・エディタで以下のように表示されます。

CSP

```
/// A simple persistent class

Class User.Person Extends %Persistent
{

Property Name As %String;

Property City As %String;

Storage Default
{
<Data name="PersonDefaultData">
    <Value name="1">
        <Value>%%CLASSNAME</Value>
    </Value>
    <Value name="2">
        <Value>Name</Value>
    </Value>
    <Value name="3">
        <Value>City</Value>
    </Value>
</Data>
<DataLocation>^User.PersonD</DataLocation>
<DefaultData>PersonDefaultData</DefaultData>
<IdLocation>^User.PersonD</IdLocation>
<IndexLocation>^User.PersonI</IndexLocation>
<StreamLocation>^User.PersonS</StreamLocation>
<Type>%Storage.Persistent</Type>
}

}
```

XML ストレージ定義には、定義されたすべてのストレージ・キーワードと、XML 要素として表された対応する値が含まれています。

クラス定義と同様に、この定義はクラス・エディタで直接編集できます。無効な XML 構文を入力した場合、エラーとしてエディタにカラー表示でその構文が表示されます。

ストレージ定義は、単純な変更、または変更を繰り返す場合に便利です。

例えば、物理的なストレージ・レイアウトを変更することなく、プロパティ名を City から HomeCity に変更する場合 (つまり、新しい名前のプロパティで、古い名前のプロパティに保存されている値にアクセスする場合)を考慮してみましょう。これは、クラス・エディタを使用して以下のように行うことができます。

1. クラス定義をスタジオのクラス・エディタ・ウィンドウにロードし、そのストレージを表示します。
2. エディタの **[既定値に戻す]** コマンドを使用して、プロパティ City のすべてのオカレンスを HomeCity と置換します。(プロパティ定義、メソッド・コード、または説明、あるいはストレージ定義の本文の) プロパティ名である City のオカレンスだけを変更するように注意してください。他のクラス定義キーワードの値を置換しないでください。
3. クラス定義を保存し、再コンパイルします。

13

ルーチンおよびインクルード・ファイルを使用した作業

ルーチンは、InterSystems IRIS® サーバ内での実行の単位です。InterSystems IRIS サーバで動作するすべてのアプリケーション・ロジックは、ルーチンおよびルーチン内のエントリ・ポイントを呼び出すことによって実行されます。ルーチンは、InterSystems IRIS サーバ環境に組み込まれた仮想マシンで実行されます。ルーチンは、InterSystems IRIS でサポートされているすべてのプラットフォームへ移植可能であり、InterSystems IRIS 環境で自動的に共有できます。

インクルード・ファイル (.inc ファイル) は、マクロ定義 (またはその他のインクルード・ファイル) を含み、.mac ルーチンまたはクラス定義にインクルードできます。マクロの詳細は、"マクロの使用" を参照してください。

13.1 ルーチン・エディタ

ルーチン・エディタでは、ルーチンまたはインクルード・ファイルのソースを直接作成したり、編集することができます。ルーチン・エディタでは、構文のカラー表示がサポートされており、構文エラーは赤い波線で示されます。

クラス定義がコンパイルされるとき、クラス・コンパイラでは、そのクラスの実装が含まれた一連のルーチンが生成されます。この生成されたソース・コードを表示および編集する場合は、生成されたコードのコピーをコンパイラが保存するように指定する必要があります。生成されたコードのコピーを保存するには:

1. **[ツール]→[オプション]** を選択します。
2. 左側のペインで **[コンパイラ]→[フラグおよび最適化]** に移動します。
3. **[生成されたソースコードを保存]** を選択します。

13.2 ルーチン・ソース形式

InterSystems IRIS には、数種類のルーチン・ソース形式 (ファイル) があります。ルーチン・エディタでは、これらすべての形式に対して構文のカラー表示がサポートされています。形式には以下のものがあります。

- MAC — .mac 拡張子を持つマクロ・ソース・ファイル。このファイルは、マクロ、埋め込み SQL 文、および埋め込み HTML を解決するために InterSystems IRIS マクロ・プリプロセッサで処理されます。処理の結果、.int ファイルが生成されます。

- INT ― 中間ソース・ファイル。このファイルは実行可能な InterSystems IRIS オブジェクト (OBJ) コードに直接コンパイルされます。
- INC ― インクルード・ファイル。本質的にはルーチンではありません。.inc ファイルには、マクロ定義があり、その定義は .mac ルーチンによるインクルードが可能です。

既定では、新規の ObjectScript ルーチンを作成するとき、そのルーチンは .mac ルーチンとして保存されます。このルーチンを異なるタイプのルーチンとして保存するには、**[ファイル]→[名前を付けて保存]** を選択します (例えば、拡張子を .mac から .inc に変更します)。

[ビュー]→[他の表示] を使用すると、指定した .mac ファイルに対応する .int コードを表示でき、その逆も可能です。

13.3 新規のルーチンまたはインクルード・ファイルの作成

新規のルーチンまたはインクルード・ファイルを作成するには、**[ファイル]→[新規作成]** を選択します。ダイアログに、選択可能なテンプレートが表示されます。インクルード・ファイルの場合、[ObjectScript] を選択します。これにより、Untitled などの既定名が付いた新規のルーチン・エディタ・ウィンドウが表示されます。これは、**[ファイル]→[名前を付けて保存]** を使用すると、異なる名前を付けて保存できます。

13.4 既存のルーチンまたはインクルード・ファイルを開く

既存のルーチンを開くには、**[ファイル]→[開く]** を使用します。**[ファイルの種類]** ドロップダウン・リストで、必要なファイル拡張子を選択し (.mac、.int、または **全ファイル** など)、ルーチンを選択します。

以前に保存したルーチンまたはインクルード・ファイルを開くときは、**[開く]** ダイアログで、* や ? を使用したワイルドカード・マッチングによって、使用可能なルーチンまたはインクルード・ファイルのリストを表示できます。* (アスタリスク) は任意の文字列と一致し、? (疑問符) は任意の 1 文字と一致します。BAS、MAC、INT、INC というルーチン・タイプは、ワイルドカード・マッチングのためのファイル拡張子として使用されています。

13.5 ルーチン・テンプレート・ファイル

新規のルーチンをスタジオで作成するとき、スタジオでは新規のルーチン・エディタ・ウィンドウが開かれます。ルーチン・テンプレート・ファイルが存在する場合は、それが新しいファイルにコピーされます。ルーチン・テンプレート・ファイルを作成するには、テンプレートに含めるコンテンツを使用してファイルを作成します。ファイルをスタジオの実行可能ファイル (CStudio.exe) と同じディレクトリに Default.mac として保存します。

13.6 ルーチンの保存、コンパイル、および削除

ルーチンをデータベースに保存するには、**[ファイル]→[保存]** または **[ファイル]→[名前を付けて保存]** を選択できます。既定では、ルーチンを保存しても自動的にコンパイルは行われません。ルーチンが保存されるたびにコンパイルされるように、この動作を変更するには:

1. **[ツール]→[オプション]** を選択します。
2. 左側のペインで **[コンパイラ]→[動作]** に移動します。

3. [保存時にルーチンをコンパイル] を選択します。

ルーチンを直接コンパイルするには、[ビルド]→[コンパイル] を選択します (これにより、ルーチンは保存されます)。

ルーチンを削除するには、ワークスペース・ウィンドウで、ルーチンをハイライト表示して、[編集]→[削除] を選択します。ルーチンと生成されたすべてのファイルが削除されます。

13.7 ルーチンおよびインクルード・ファイルのバックアップの自動保存

既存のルーチン (またはインクルード・ファイル) を保存する際、スタジオでは自動的にバックアップ・ファイルを作成します。最大 5 つのバックアップ・ファイルを、;# (セミコロンと数字) 接尾語を使用した名前を付けて保存します。例えば、setup.MAC という名前のファイルを 6 回保存した場合は、次のように名前が付けられた 5 つのバックアップ・ファイルがあります。

```
setup.MAC;1
setup.MAC;2
setup.MAC;3
setup.MAC;4
setup.MAC;5
```

特に、次の拡張子を持つファイルは、自動的にバックアップされます。.BAS、.INC、.INT、.MAC、.OBJ、.MVB、.MVI、.CSP。

どのバックアップ・ファイルが存在するかを確認するには、[ファイル]→[開く] オプションの検索フィールドにセミコロンを使用します。使用できる構文例は以下のとおりです。

構文	結果
`*.*;*`	このフォルダ内のすべてのバックアップ・ファイルを表示します。
`*.mac;*`	.MAC 拡張子を持つすべてのバックアップ・ファイルを表示します。
`setup.*;*`	setup という名前を持つすべてのバックアップ・ファイルを表示します。

注釈　管理ポータルからルーチンやインクルード・ファイルを開いているときに、この構文を使用して、バックアップ・ファイルを検索することができます。管理ポータルからバックアップ・ファイルを開くには、以下の手順を実行します。

1. InterSystems IRIS ランチャーから管理ポータルを選択します。
2. [システムエクスプローラ]→[ルーチン] を選択します。
3. [進む] をクリックします。
4. 左側のペインの [ルーチンおよびインクルード・ファイル] 検索ボックスに、バックアップ・ファイルの構文を入力します。

14

スタジオ・デバッガの使用

スタジオのデバッガを使用すると、InterSystems IRIS® サーバで動作するプログラムをステップ実行できます。デバッグ可能なプログラムとしては、INT ファイル、BAS ファイル、MAC ファイル、CLS ファイル内のメソッド、Java から呼び出されるサーバ側のメソッド、サーバでホストされるアプリケーションなどがあります。クラスでステップ実行やブレークポイントの設定を行うには、対応する INT ファイルまたは BAS ファイルを開き、そのファイルでデバッグ・コマンドを使用します。INT ソース・コード・ファイルを表示する前に、以下を行う必要があります。

1. **[ツール]→[オプション]** を選択します。
2. 左側のペインで **[コンパイラ]→[フラグおよび最適化]** に移動します。
3. **[生成されたソースコードを保存]** を選択します。

以下のいずれかの方法で、デバッガをターゲット・プロセスに接続できます。

- **[デバッグ]→[アタッチ]** を選択して、InterSystems IRIS サーバで実行中のプロセスを選択します。実行中プロセスをアタッチするには、以下のいずれかが必要です。
  - **%ALL** ロール
  - **%System_Attach** リソースに対する **Use** 特権を持つロール
  - デバッグを試みるプロセスと同じ $USERNAME
- **[プロジェクト]→[設定]→[デバッグ]→[デバッグ対象]** (または **[デバッグ]→[デバッグ対象]**) を選択して、現在のプロジェクトのデバッグ対象 (デバッグするプログラムまたはルーチンの名前) を設定します。次に、**[デバッグ]→[実行]** を選択して対象のプログラムを起動し、そのサーバ・プロセスに接続します。

注釈　　または、コマンド行から zbreak コマンドを入力してデバッグを起動した方が、便利な場合もあります。zbreak の詳細は、"ObjectScript の使用法" の "コマンド行ルーチンのデバッグ" を参照してください。

14.1 デバッグ・セッションのサンプル : ルーチンのデバッグ

以下の例では、ルーチンをデバッグする方法を示します。

1. スタジオを起動し、**[ファイル]→[プロジェクトの新規作成]** を選択して、Project1 という新規プロジェクトを作成します。
2. **[ファイル]→[新規作成]→[一般] タブ→[ObjectScript ルーチン]** を選択して、新規のルーチンを作成します。

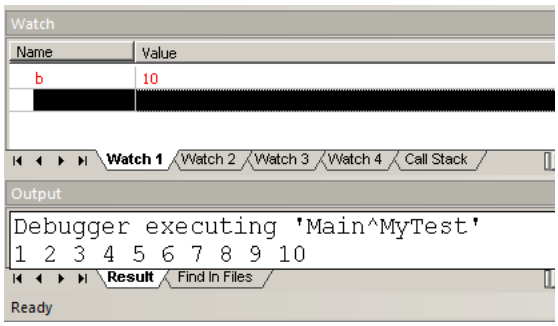
3. このルーチンにコードを入力します。

```
MyTest ; MyTest.MAC

Main() PUBLIC {
    Set a = 10
    For i = 1:1:10 {
        Set b = i
             Write b," "
    }
}
```

4. [ファイル]→[名前を付けて保存] を使用して、新規のルーチンを MyTest.MAC として保存し、コンパイルします。
5. [デバッグ]→[デバッグ対象] タブを選択し、[クラス・メソッドまたはルーチン] を選択します。次に、新規ルーチンのエントリ・ポイントの名前 `Main^MyTest` を入力して、プロジェクトのデバッグ対象を定義します。
6. ルーチン内のブレークポイントを設定します。`Set a = 10` という行にカーソルを置き、F9 (トグル・ブレークポイント) キーを押します。左の余白にブレークポイント・インジケータ  が表示されます。
7. [デバッグ]→[実行] を選択して、デバッグを開始します。デバッガがブレークポイントで停止すると、次に実行するコマンドの簡単な説明が、黄色のボックスに表示されます。INT ファイルが新しいウィンドウで開きます。INT ファイルが開かない場合は、[生成されたソースコードを保存] オプションが有効になっていることを確認してください。([ツール]→[オプション] を選択して、左側のペインで [コンパイラ]→[フラグおよび最適化] をクリックし、[生成されたソースコードを保存] を選択します。)
8. ウォッチ・ウィンドウ ([表示]→[ウォッチ]) に b および a を入力して、これらの値を参照できるようにします。
9. [デバッグ]→[ステップイン] (F11) を繰り返し選択してプログラムのステップ実行を行い、b の値が変化することに注意します。

[デバッグ]→[停止] を使用して、デバッグをプログラムの最後で停止できます。



14.2 現在のプロジェクトのデバッガ設定

デバッガ設定のいくつかは、現在のプロジェクトで指定したり、保存できるものがあります。これには、以下のものがあります。

- デバッグ対象
- ブレークポイント
- ウォッチポイント

14.2.1 デバッグ対象

デバッグ対象によって、デバッグするプロセスをスタジオに指定します。

プロジェクトのデバッグ対象を指定するには、**[プロジェクト]→[設定]→[デバッグ]→[デバッグ対象]** を選択するか、**[デバッグ]→[デバッグ対象]** を選択します。以下のいずれかを選択し、**[デバッグ]→[実行]** を選択して実行します。また、エディタ・ウィンドウで項目の横にカーソルを置いて、右クリックし、**[デバッグ対象として xxxx を設定]** を選択して、デバッグ対象を設定することもできます。

クラス・メソッドまたはルーチン

[デバッグ]→[実行] の選択時にデバッグするルーチン（およびタグ）、クラス、またはメソッド。例えば、ルーチン `MyRoutine` 内のタグ `Test` からデバッグを開始するには、`Test^MyRoutine()` と入力します。または、`##class(MyApp.Person).Test(1)` のように、実行するクラス・メソッドの名前を入力します。

CSP ページ（URL、CSP、またはクラス）

[デバッグ]→[実行] の起動時にアクセスする CSP ページ。デバッガは、CSP ページの HTTP 要求を処理するインターシステムズ・サーバ・プロセスに接続します。このオプションを使用して、CSP アプリケーションをデバッグします。例えば、**Test.csp** ページのコードを実行するには、デバッグ対象として **/csp/user/Test.csp** と入力します。InterSystems IRIS での CSP ファイルの使用はお勧めしません。

14.2.2 ブレークポイント

プロジェクトのデバッグ対象にデバッグを開始するとき（**[デバッグ]→[実行]**）、プロジェクトによって定義されたブレークポイントはすべて、対象プロセス内で設定されます。

ブレークポイントを設定および削除する最も簡単な方法は、コード行にカーソルを置き、**F9** キーを押して、ブレークポイントのオンとオフを切り替えることです。ブレークポイント位置にカーソルを置き、**[デバッグ]→[ブレークポイント]→[ブレークポイント切替]** を選択することもできます。ブレークポイントを表示し、条件付きブレークポイントを設定するには、**[デバッグ]→[ブレークポイント]→[ブレークポイントの表示]** を選択します。また、**[プロジェクト]→[設定]→[デバッグ]→[ブレークポイント]** を使用しても、ブレークポイントを追加、または削除できます。

注釈　1 つのルーチンに設定できるブレークポイントの最大数は 20 です。20 を超えるブレークポイントが設定されると、デバッガに `<ROUTINELOAD>^%Debugger.System.1` が表示され、デバッグが停止します。

14.2.3 ウォッチポイント

ウォッチポイントは、値が変化するたび、または強制終了されるたびに、実行を停止する変数です。ウォッチポイントを表示し、条件付きウォッチポイントを設定するには、**[デバッグ]→[ブレークポイント]→[ブレークポイントの表示]** を選択します。また、**[プロジェクト]→[設定]→[デバッグ]→[ウォッチポイント]** を使用しても、ウォッチポイントを追加、または削除できます。

14.3 [デバッグ] メニュー

[デバッグ] メニュー・オプションには、以下のものがあります。

メニュー・オプション	アクション
 アタッチ	インターシステムズ・サーバが現在実行しているプロセスのリストを表示し、デバッグするプロセスにアタッチします。 プロセスを選択し、[OK] を選択すると、スタジオは選択された対象プロセスに割り込み、デバッグを開始できるようになります。 対象プロセスで実行中の現在のルーチンのソースを生成している場合、そのソースはエディタ・ウィンドウに表示されます。 [デバッグ]→[停止] を使用してデバッグを終了した場合、後で対象プロセスの実行を再開できます。
 実行	現在デバッグ中でない場合は、[実行] をクリックすると、プロジェクトのデバッグ対象で指定されている対象プロセスが起動されます。 対象プロセスが設定されていない場合は、設定するように求められます。デバッグ対象は、実行するルーチンまたはメソッドの名前であり、[デバッグ対象] ダイアログで設定できます。 対象プロセスが起動されると、最初のブレークポイントまで実行されます。アプリケーション内でブレークポイントを設定していない場合、最後まで停止することなく実行を完了します。
 再開	対象プロセスの実行をいったん停止して再起動してから、デバッグを再開します ([実行] コマンドを使用した場合と同様)。
 停止	デバッグを停止し、対象プロセスを停止するか、アタッチを解除します。対象プロセスが実行されており、[アタッチ] を使用してアタッチされている場合、対象プロセスの実行は継続されます。対象プロセスが [実行] コマンドによって起動している場合は、終了します。
 ブレーク	対象プロセスの実行を一時停止します (現在停止中ではなく実行中の対象プロセスにデバッガがアタッチされている場合)。
 中断	現在のコマンドの実行を中断します。
ステップイン	現在のコマンドを対象プロセス内で実行して、次のコマンドで停止し、すべての関数呼び出しやループ本文にステップインします。
 ステップオーバ	対象プロセス内の現在のコマンドを実行し、次のコマンドで停止します。デバッガは、検出したすべての関数呼び出しやコード・ブロック (ループなど) をステップオーバして、関数呼び出しやコード・ブロックに続くコマンドで停止します。
 ステップアウト	現在のコード・ブロックや関数から離れるかステップ・アウトすることで対象プロセスの実行を進めて、この外部レベルの次のコマンドで停止します。
 カーソル位置まで実行	INT ルーチンを含むドキュメントに対してのみ利用できます。 対象プロセスの実行を開始し、現在のカーソル位置に到達したときに停止します。この操作は、エディタ・ウィンドウを使用して現在の行でブレークポイントを設定し、[実行] コマンドを実行して、プログラムが停止したときにブレークポイントを消去するのと同じです。
ブレークポイント	ブレークポイント切替 : 現在のドキュメント内の現在行のブレークポイントを設定またはクリアします。 ブレークポイントの表示 : [ブレークポイント] ダイアログが開いて、ブレークポイントのリスト表示、追加、および削除を実行できます。
デバッグ対象	デバッグ対象 (メソッドまたはルーチン) を入力します。"デバッグ対象" も参照してください。

14.4 ウォッチ・ウィンドウ

ウォッチ・ウィンドウには、変数の値と単純な式を示すテーブルが表示されます。ウォッチ・ウィンドウにリスト表示されているすべての変数と式は、各デバッガ処理（**ステップオーバ**など）の後に評価され、その結果として得られた値がウォッチ・ウィンドウの 2 列目に表示されます。デバッガ・オペレーション後に変数の値や式が変更された場合は、赤で表示されます。ウォッチ・リストの変数が評価されるときに未定義の場合、その値は <UNDEFINED> として表示されます。同様に、結果がエラーとなる式でも、その値に対するエラー・メッセージが表示されます。また、デバッガで変数の上にマウスを置くことにより、変数の値を確認することができます。

ウォッチ・ウィンドウに変数や式を追加するには、最初の列で空のセルをダブルクリックし、変数や式を入力します。または、エディタ・ウィンドウでマウスを使用してテキストをハイライト表示し、ウォッチ・ウィンドウの空のセルにドラッグ・アンド・ドロップすることもできます。ウォッチ・ウィンドウのコンテンツを編集する場合は、変数や式をダブルクリックして入力することができます。

以下は、ウォッチ・ウィンドウで入力できる変数や式の例です。

- a
- a + 10
- a(10,10)
- $L(a)
- person.Name

ウォッチ・ウィンドウの Value 列に新規の値を入力して、対象プロセスの変数の値を変更することもできます。

14.4.1 デバッガ・ウォッチ・ウィンドウ・コンテキスト・メニュー

デバッガ・ウォッチ・ウィンドウを右クリックすると、以下のコンテキスト・メニューが表示されます。

メニュー・オプション	アクション
削除	ウォッチ・リストからそのアクティブな変数を削除します。
表示形式	リストから表示形式を選択します。
オブジェクトのダンプ	選択された変数に対する %SYSTEM.OBJ.Dump() の結果を表示します。
[更新]	ウォッチ・リストを更新します。
すべて削除	ウォッチ・リストからすべてのアクティブな変数を削除します。
ウォッチに追加	選択された配列の要素またはオブジェクト・プロパティを、独立したエントリとしてウォッチ・メニューに追加します。

15
スタジオ・テンプレートの使用法

ここでは、スタジオ・テンプレートの使用方法について説明します。

テンプレートを使用すると、スタジオ編集ウィンドウに機能を繰り返し追加できます。以下に示す 2 つのタイプのテンプレートがあります。

- テキスト・テンプレート – (テンプレートという言葉で表現されるもの) シンプル・テキスト・テンプレートは、生成されたテキストをドキュメントに挿入します。インタラクティブ・テキスト・テンプレートには、ユーザ入力を含めることができます。テキスト・テンプレートを使用するには、**[ツール]→[テンプレート]→[テンプレート...]** を選択します。
- アドイン・テンプレート – スタジオに新規のツールを作成します。アドイン・テンプレートは、テキストをドキュメントに挿入せず、ドキュメントを開く必要がない点で、テキスト・テンプレートとは異なります。

  アドイン・テンプレートを使用するには、**[ツール]→[アドイン]** を選択します。SOAP Web クライアント・ウィザードの使用の詳細は、"Web サービスおよび Web クライアントの作成" を参照してください。XML スキーマ・ウィザードの使用の詳細は、"XML スキーマ・ウィザードの使用法" を参照してください。

注釈　スタジオ・テンプレートがすばやく開くように、Internet Explorer でプロキシ設定の自動検出を無効にします。

1. Internet Explorer を起動して、**[ツール]→[インターネット オプション]** から **[接続]** タブを選択します。
2. **[LAN の設定]** を選択して、すべてのチェック・ボックスのチェックを外します。このページでチェックが付いていないことを確認し、**[OK]** を 2 回選択して [インターネット オプション] ダイアログを閉じます。

15.1 スタジオ・テンプレートへのアクセス

テンプレートを開くには、**[ツール]→[テンプレート]** を使用するか、エディタ・ウィンドウの右クリック・メニューを使用します。各テンプレートは、1 つ以上のドキュメント・タイプに関連付けられています。現在のウィンドウのドキュメント・タイプに関連付けられているテンプレートのみが、テンプレート・リストに表示されます。

テキスト・テンプレートには、シンプルとインタラクティブの 2 つのスタイルがあります。シンプル・テンプレートでは、ユーザが操作しなくても、カーソル・ポイントにテキストが挿入されます。インタラクティブ・テンプレートでは、追加の情報の入力が促される 1 つ以上の画面 (ウィザードと同様) が表示されます。

テンプレートを開いたときにハイライト表示されているテキストは、テンプレートによって置換されます。多くのテンプレートは、現在ハイライト表示されているテキストを、テンプレート・プログラムに対する入力として使用します。

15.2 標準スタジオ・テンプレート

スタジオには、複数のテンプレートが用意されています。スタジオですべてのテンプレートのリストを、%SYS ネームスペースの [ワークスペース] ウィンドウ→[ネームスペース] タブ→[CSP ファイル] の下に表示できます。 現在のドキュメントで使用可能なテンプレートを表示するには、[ツール]→[テンプレート]→[テンプレート] を使用します。以下で、これらのテンプレートを説明します。

注釈　既定では、スタジオ・テンプレートはセッション・タイムアウトを 90 秒としています。スタジオ・テンプレートにデータを入力する場合、ユーザの入力が 90 秒間ないと、セッションは終了します。

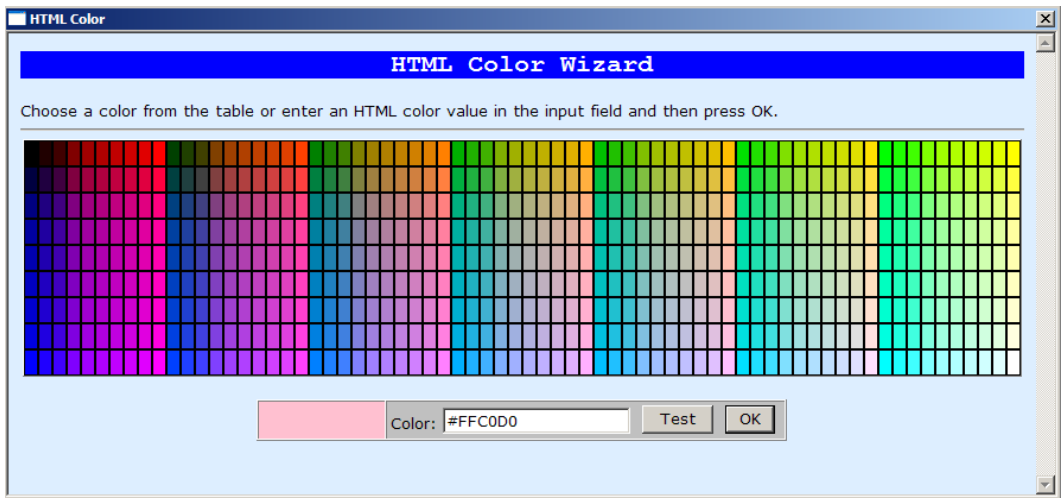
15.2.1 テンプレート

このセクション内の 3 つのテーブルでは、スタジオで使用できる以下のテンプレートが定義されています。

テーブル 15-1: テンプレート

テンプレート	説明
HTML Color	カーソル・ポイントに HTML カラー値文字列 (#F0F0F0 など) を挿入するために選択します。
HTML Input	HTML 入力制御をカーソル・ポイントに挿入するために選択します。
HTML Script	指定した言語とコンテンツが設定された <SCRIPT> タグをカーソル・ポイントに挿入するために選択します。
HTML Table	指定した特性が設定された HTML テーブルをカーソル・ポイントに挿入するために選択します。[Preview] を選択して、プレビュー・ウィンドウを表示します。
HTML Tag	指定した属性を持つリストから選択した HTML タグをカーソル・ポイントに挿入するために選択します。または、既存の HTML タグをハイライト表示してテンプレートを起動した場合は、表示された属性値を編集できます。

図 15-1: インタラクティブ・テンプレート、HTML カラー・テーブルの例



15.2.2 クラス定義テンプレート

テンプレートの多くが、クラス定義内で利用できます (&html<> ブロックの場合に便利です)。また、以下のテンプレートが利用できます。

テーブル 15-2: クラス定義テンプレート

テンプレート	説明
SQL 文	指定した SQL 文のコードをカーソル・ポイントに挿入するために選択します。[Preview] を選択すると、ポップアップ・プレビュー・ウィンドウ内で (データベース内のデータを使用した) テーブルの結果を表示できます。テンプレートが SQL テキストのみを返すかまたは SQL テキストを基にした埋め込み SQL カーソルを返すかを指定できます。また、SQL テキストに基づいて %ResultSet オブジェクトを返すこともできますが、これは InterSystems IRIS® では推奨されません。
Web フォーム・ウィザード	CSP フォームを作成できるウィザードを開き、使用するフォームのクラス・メンバとテーブルのスタイルを指定します

15.2.3 アドイン・テンプレート

[ツール]→[アドイン] メニューには、プロジェクトに項目を追加できるウィザードの一覧が含まれています。メニューには、以下のアドインが含まれています。

テーブル 15-3: アドイン

アドイン	機能	詳細
アクティベート・ウィザード	このウィザード・オプションは、InterSystems IRIS では使用されません。	このウィザードのドキュメントについては、ここをクリックしてください。
SOAP ウィザード	WSDL (Web サービス記述言語) ドキュメントを読み取り、1 つ以上の Web クライアント・クラスまたは Web サービス・クラスを作成します。	Web サービスおよび Web クライアントの作成
XML スキーマ・ウィザード	XML スキーマを読み取り、対応するクラスのセットを作成します。	XML スキーマからのクラスの生成
XSL 変換ウィザード	指定された XSL スタイルシートを使用して XML ファイルを変換します。	XSLT 変換の実行

16

スタジオ・メニュー・リファレンス

ここでは、スタジオ・メニューで利用できるメニューとキーボード・オプションについて説明します。

16.1 [ファイル] メニュー

[ファイル] メニューには、ドキュメントやプロジェクトをオープンしたり、保存したりするオプションがあります。オプションは、ファイルを開いているかどうかによって変わります。"キーボード・アクセラレータ" も参照してください。

メニュー・オプション	アクション
新規スタジオ	別の InterSystems IRIS® サーバへの接続に使用します。
ネームスペース変更 ...	ネームスペースへの変更に使用します。
 新規作成 ...	エディタ・ウィンドウで新規ドキュメント (クラス定義またはルーチンなど) を作成するときに使用します。ファイルをスタジオにドラッグ・アンド・ドロップすることもできます。ドキュメント・タイプは以下の 4 つのタブに分類されています。 ・ 一般 – 新規に ObjectScript ルーチン、InterSystems IRIS クラス定義 (新規クラス・ウィザードを使用)、Web サービス/クライアント構成、または Web サービスを作成します。 ・ CSP ファイル – 新規に CSP ページ、XML ファイル、JavaScript ファイル、またはカスケーディング・スタイル・シート (CSS) ファイルを作成します。InterSystems IRIS での CSP ファイルの使用はお勧めしません。 ・ Zen – 新規に Zen アプリケーション、Zen コンポーネント、Zen ページ、Zen レポート、Zen フォーム、または Web サービスを作成します。InterSystems IRIS での Zen アプリケーションの使用はお勧めしません。 ・ カスタム – 新しい InterSystems Business Intelligence KPI (以前は DeepSee KPI と呼ばれていました) 用。詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

メニュー・オプション	アクション
 開く	現在の InterSystems IRIS ネームスペースおよびサーバから、既存の項目をオープンできます。 [開く] ダイアログは、現在のネームスペースに対して表示されます。必要に応じて、ファイル・タイプを選択します。Ctrl-A を押すと、すべてが選択されます。 項目が他のユーザによって使用されている場合、読み取り専用アクセスでオープンできます。 自動的に保存されたバックアップ・ファイルを開くには、"ルーチン、インクルード・ファイルのバックアップの自動保存" を参照してください。 スタジオでは、現在のサーバとネームスペースのみから、クラス定義やルーチンを編集することが可能です。別のサーバまたはネームスペースから項目をオープンするには、[ファイル]→[ネームスペース変更] を使用してネームスペース名を変更するか、別のサーバに切り替えます。[ファイル]→[新規スタジオ] を使用して、2 つ目のスタジオ・ウィンドウをオープンします。 項目を選択して [削除] を右クリックすると、項目とすべてのサブ項目は削除されます。例えば、.INT ファイルを選択すると、.INT ファイルと .OBJ ファイルが両方とも削除されます。.cls ファイルを選択すると、関連付けられているすべての .INT ファイルと .OBJ ファイルも削除されます。
 URL を開く	エディタに HTML ソースを表示します。これは、Web ベース・アプリケーションの開発時に、進捗状況を表示する場合に役立ちます。
閉じる	現在のエディタを閉じます。
 保存	現在のエディタのコンテンツを保存します。
名前を付けて保存 ...	現在のエディタのコンテンツを、指定した名前で保存します。
 全て保存	開いているすべてのウィンドウのコンテンツを保存します。
プロジェクトの新規作成	現在の InterSystems IRIS サーバとネームスペースで新規のプロジェクトを作成します。
プロジェクトを開く ...	現在の InterSystems IRIS サーバとネームスペースで既存のプロジェクトを開きます。プロジェクトをスタジオにドラッグ・アンド・ドロップすることもできます。 別のサーバまたはネームスペースからプロジェクトをオープンするには、[ファイル]→[ネームスペース変更] を使用してネームスペース名を変更するか、別のサーバに切り替えます。[ファイル]→[新規スタジオ] を使用して、2 つ目のスタジオ・ウィンドウにプロジェクトをオープンします。
プロジェクトを保存	現在のプロジェクトの設定を保存します。このコマンドでは、プロジェクトに属する変更されたドキュメントの保存を行うことはできません。
プロジェクトを名前を付けて保存	現在のプロジェクトを、指定した名前で保存します。
プロジェクトを閉じる	現在のプロジェクトを閉じます。
 印刷 ...	現在のドキュメントを印刷します。
印刷プレビュー	印刷したときのドキュメントの外観を表示します。
印刷設定 ...	[プリンタの設定] ダイアログが開きます。ここで、ドキュメントの印刷方法を設定できます。

メニュー・オプション	アクション
最近使用したファイル ...	最近使用したファイルが表示されます。[詳細] は、ファイル(ある場合)を [本日]、[昨日]、[最新 7 日間]、[最新 30 日間]、および [すべて] のカテゴリで表示します。[すべて] は 100 ドキュメントまでに制限されます。[履歴のクリア] を選択すると、すべてを削除できます。[ドキュメントを開く] を選択すると、選択したドキュメントが開きます。
最近使用したプロジェクト ...	最近使用したプロジェクトが表示されます。
終了	現在のスタジオ・セッションを終了します。

16.2 [編集] メニュー

[編集] メニューには、編集およびナビゲーション・オプションがあります。ほとんどのオプションは、キーボードによるショートカットからも利用できます。"キーボード・アクセラレータ" を参照してください。

16.2.1 基本的な編集

メニュー・オプション	アクション
元に戻す	最後に行われた操作を元に戻します。 ただし、クラス・インスペクタを使用してクラスに行った変更は、[元に戻す] で元に戻すことはできません。
やり直し	最後に行った [元に戻す] を無効にします。
切り取り	現在選択されているテキストを削除し、クリップボードにコピーします。
コピー	現在選択されているテキストを、クリップボードにコピーします。
貼り付け	クリップボードのコンテンツを、現在のカーソル位置に挿入します。
削除	現在選択されているテキストを、クリップボードにコピーせずに削除します。ワークスペース・ウィンドウで項目をハイライト表示すると、その項目とその生成されたすべてのファイルが削除されます。
全て選択	現在のドキュメントのすべてのコンテンツを選択します。

16.2.2 検索と置換

メニュー・オプション	アクション
検索	現在のドキュメント内でテキストを検索します。* と ? を使用したワイルドカード・マッチングを使用できます。*(アスタリスク) は任意の文字列と一致し、?(疑問符) は任意の 1 文字と一致します。*、?、または ¥(アスタリスク、疑問符、または円記号) 文字を検索するには、円記号(¥)を使用してエスケープします。タブを検索するには、\t を使用します。検索する要素タイプの指定とバックスラッシュ・エスケープの説明は、このテーブルの下にある "検索の詳細" を参照してください。

メニュー・オプション	アクション
ファイルから検索	InterSystems IRIS サーバ上の複数のファイルでテキストを検索します。検索する文字列を入力し、検索するファイルのタイプを選択して (クラス定義用の .cls など)、[検索] をクリックします。詳細は、このテーブルの下にある "ファイルから検索" を参照してください。
検索	現在のプロジェクト内でクラスを検索します。[検索] ウィンドウで、クラス名を入力します。リストには、一致したエントリが表示されます。クラスを開くには、目的のクラスをダブルクリックするか、行を選択して [移動] をクリックします。
 キャンセル	実行中の [ファイルから検索] 操作またはクラス・コンパイルをキャンセルします。
置換	現在のドキュメント内のあるテキスト文字列を別のテキスト文字列に置換します。
 移動	現在のドキュメント内の、行番号または (ルーチンやクラス定義の場合は) タグまたはクラス・メンバとして指定された位置に、カーソルを移動します。
戻る	[移動] アクションの後で、[移動] アクションの前の場所にカーソルを戻します。
ブックマーク	以下の "ブックマーク" を参照してください。
詳細	"詳細" を参照してください。
次のエラー	カーソルを、CSP ファイル内の次のエラーに移動します。InterSystems IRIS での CSP ファイルの使用はお勧めしません。
前のエラー	カーソルを、CSP ファイル内の前のエラーに移動します。InterSystems IRIS での CSP ファイルの使用はお勧めしません。
次の警告	カーソルをファイル内の次の警告に移動します。
前の警告	カーソルをファイル内の前の警告に移動します。

16.2.2.1 検索の詳細

バックスラッシュ・エスケープ

通常、検索エンジンでは、バックスラッシュ (¥) はメタキャラクタ (つまり、それ自体以外の何かを意味する文字) であると解釈されます。この場合、バックスラッシュとその後に続く文字は、2 文字のコードになります。 検索エンジンでは、バックスラッシュの検索時には常に 2 番目の文字が検索されるため、バックスラッシュ自体を検索する際には 2 文字のコードを作成する必要があります。2 番目のバックスラッシュがある 2 文字のコード (¥¥) を作成してください。検索エンジンは、これをバックスラッシュ文字自体と解釈します。

この規約は、UNIX の grep コマンドの開発時に実装されたものであり、それ以降、この規約 (または基になる C コード) は、何度も繰り返し複写されてきました。

一致する要素タイプ

特定の要素タイプ (コマンド、変数、演算子など) のテキストを検索するには、**[検索対象]** フィールドに目的のテキストを入力し、**[一致する要素タイプ]** チェック・ボックスにチェックを付けます。

注釈　**[検索]** では、選択した言語に関係なく、開いているファイルに、そのタイプのすべての要素の検索対象テキストが表示されます。**[言語]** フィールドで、**[要素]** リストに表示する要素タイプの数を制限するために、目的の言語を選択します。**[要素]** フィールドで、検索対象のテキストを含む要素タイプを選択して、**[次を検索]** を選択します。

例えば、**[クラス定義言語]** を選択した状態で、**[コメント]** で `Set` という単語を検索すると、ファイル内に任意の言語で存在するコメント内の `Set` という単語のすべてのインスタンスが一致します。

16.2.2.2 ファイルから検索

[ファイルから検索] ダイアログで **[検索]** を選択すると、スタジオは現在の InterSystems IRIS ネームスペースで、選択されたファイルを検索し、検索文字列を含むすべてのファイル (最大 5,000 個) のリストを返します。検索結果の項目をダブルクリックすると、ファイルが開かれてその項目がハイライト表示されます。選択した項目の行番号と列番号がステータス・バーの右隅に表示されます。

[ファイルから検索] は格納されたデータを検索しますが、変更された開いているドキュメントは検索されません。現在のプロジェクト内のみを検索し、現在のプロジェクトが新規のプロジェクトか変更したプロジェクトである場合、プロジェクトを保存するよう求められます。拒否した場合、[ファイルから検索] はキャンセルされます。

バックスラッシュ (¥) を [ファイルから検索] で見つけるには、もう 1 つのバックスラッシュ (¥¥) でバックスラッシュをエスケープする必要があります。

[フィルタ] フィールドには、以下の一覧の要素を含めることができます。SQL 論理演算子の AND および OR を使用すると、複数のフィルタを入力できます。例えば、`Type=5 AND Modified>01/01/08` などと指定できます。**[フィルタ]** フィールドの内容は、SQL WHERE 節の一部となります。これらのフィールドは、%Studio.OpenDialogItems クラスから得られます。

[ファイルタイプ] フィールドには、独自のカスタム・マスク (al*.mac など) を入力できます。任意のフィールドに複数のフィルタを入力するには、コンマ区切りリストを使用します。

[パターン・マッチングの有効化] チェック・ボックスを選択すると、**[検索対象:]** フィールドは、任意の 1 つの文字を表すものとして疑問符 (?) を、0 個以上の文字を表すものとしてアスタリスク (*) を受け入れます。

[フィルタ] フィールドでは、以下の項目を使用できます。

フィルタ要素	使用法
`IsTrue=1 または 0`	ドキュメントを検索するには 1 を指定します。ディレクトリを検索するには 0 を指定します。
`Name=file name`	選択された複数のファイル内で検索するファイル名を入力します。
`Characters=number of characters`	特定のサイズのドキュメントをフィルタします。SQL 関係演算子を使用できます。
`Type=#`	この後に、%Studio.OpenDialogItems.Type で示されるファイル・タイプ・リストに従ってフィルタするための整数値を入力します。これにより、ファイル・タイプ・フィールドよりきめ細かくフィルタできます。例えば、.mac ファイルのみを検索する場合、Type=5 と入力します。
`Modified=last modified timestamp`	SQL 関係演算子を使用できます。
`Generated=1 または 0`	生成されたファイルを検索するには 1 を指定します。ユーザが作成したファイルを検索するには 0 を指定します。
`Description=description`	ファイル内で検索する説明を入力します。

16.2.3 ブックマーク

ブックマークを使用すると、使用しているアプリケーション・ソース内の位置を記録できます。ブックマークは、スタジオが起動しているローカル・マシンに保存されます。これは、複数のユーザ間で共有することはできません。ブックマークのオプションは以下のとおりです。

メニュー・オプション	アクション
 ブックマーク切り替え	現在のドキュメント内の現在の行のブックマークを追加、削除します。
 ブックマークのクリア	現在のドキュメントで定義されたすべてのブックマークを削除します。
 次	カーソルを、現在のドキュメント内の次のブックマークに移動します。
 前	カーソルを、現在のドキュメント内の前のブックマークに移動します。

16.2.4 その他の編集

[その他の編集] メニューには、特定の状況でのみ表示されるいくつかのコマンドが含まれています。

メニュー・オプション	アクション

拡張コマンド	ObjectScript ルーチンを開いて、テキストをハイライト表示すると表示されます。現在選択されているテキスト内の略記されたすべての ObjectScript コマンドを、正式名称に置換します。以下はコードの例です。
	
	`S x = 10`
	
	これは、以下のように置換されます。
	
	`Set x = 10`

圧縮コマンド	ObjectScript ルーチンを開いて、テキストをハイライト表示すると表示されます。現在選択されているテキスト内のすべての ObjectScript コマンドを、略記バージョンに置換します。例えば、次のような場合、
	
	`Set x = 10`
	
	これは、以下のように置換されます。
	
	`S x = 10`

行インデントを増やす	選択した行のインデントを増やします。
行インデントを減らす	選択した行のインデントを減らします。
大文字にする	選択したテキストを大文字にします。
小文字にする	選択したテキストを小文字にします。
行のコメント化	行の最初にコメント区切り文字を追加することで、選択した行をコメント化します。選択は、先頭の空白を含め、行の最初から開始し、行の最後の文字を含めます。
行のコメント化解除	選択したコメント行の最初からコメント区切り文字を削除することで、この行を通常の行にします。
ブロックのコメント化	選択したテキスト・ブロックの最初と最後にコメント区切り文字を追加することで、このブロックをコメント化します。選択は、ブロックの最初の文字から最後の文字までに渡る必要があり、ブロック外の空の行を含めることはできません。選択がメソッドまたはルーチン内である場合、その選択には先頭の空白を含める必要があり、コメント行については、区切り文字は行の最初にのみ追加されます。

メニュー・オプション	アクション
ブロックのコメント化解除	選択したテキストのコメント・ブロックを、コメントの区切り文字を削除することで、通常のブロックにします。区切り文字でブロックを囲む場合、選択はブロックの先頭の最初の区切り文字から、末尾の最後の区切り文字までに渡る必要があります。区切り文字が行の先頭にある場合、これは複数行のコメント化解除のように機能します。
選択した行へのタブ付け	選択した行の各セットの最初にタブを追加します。
選択した行のタブ解除	選択した行の各セットの最初からタブを削除します。

16.3 [ビュー] メニュー

[ビュー] メニューには、表示を制御するさまざまなオプションがあります。[ビュー] メニューに表示されるオプションは、カーソルがある位置によって異なります。"キーボード・アクセラレータ" も参照してください。

メニュー・オプション	アクション
 ワークスペース	ワークスペース・ウィンドウの表示または非表示を指定します。ワークスペース・ウィンドウには、以下の 3 つのタブがあります。 · プロジェクト – 現在のプロジェクトのコンテンツを表示します。 · ウィンドウ – 現在の全ウィンドウのリストを表示します。 · ネームスペース – 現在のネームスペースのコンテンツを表示します。
 インスペクタ	インスペクタの表示または非表示を指定します。インスペクタは、クラス定義を編集可能な表形式で表示します。クラス定義には、ファイルでのみで変更可能なもの、ファイルまたはインスペクタで変更可能なもの、インスペクタのみで変更可能なものがあります。
 出力	出力ウィンドウの表示または非表示を指定します。出力ウィンドウは、サーバからの出力を表示します (コンパイルの結果であるエラー・メッセージなど)。このウィンドウには ObjectScript コマンドを入力でき、入力したコマンドはサーバ上で実行されます。選択した情報の行番号と列番号がステータス・バーの右隅に表示されます。
 ウォッチ	ウォッチ・ウィンドウの表示または非表示を指定します。ウォッチ・ウィンドウは、デバッグ時に変数および簡単な式の値を表示します。
ツールバー	スタジオのツールバーの表示または非表示を指定します。
 全画面	スタジオが全画面で表示されるように拡張します。
テキスト・サイズ	スタジオでのテキストのサイズを、拡大または縮小したり、通常に戻すことができます。
特殊文字を表示	エディタ・ウィンドウに、スペース、タブ、および行末文字を表示します。
行番号表示	行番号を表示します。

メニュー・オプション	アクション
再読み込み	ドキュメントがアクティブの場合は、現在のドキュメントの保存済みバージョンを再ロードします。ワークスペース・ウィンドウがアクティブの場合は、このウィンドウまたはプロジェクトを再ロードします。ワークスペース・ウィンドウの [ネームスペース] タブがアクティブの場合は、選択されたアイテムのサブツリーの親を再ロードします。ツリー全体を再ロードするには、最上位レベルをハイライト表示します。
他のコードの表示	カーソル位置に関連する、.INT や .MAC ファイルなど、コンパイラで生成されたソース・コードを表示します。このオプションは、現在のウィンドウに 1 つまたは複数のソース・ファイルが生成されている場合にのみ機能します。
他のドキュメントの表示	現在のドキュメントに関連する他のドキュメントを表示します。状況によって、[他のコードの表示] は、[他のドキュメントの表示] と同じ動作になることがあります。
ウェブページ	CSP ファイルで作業している場合にのみ使用できます。 既定の Web ブラウザを開いて現在の CSP ファイルを表示し、Web アプリケーションの開発時に CSP ページがどのように見えるかを確認できます。 必要に応じて、[プロジェクト]→[設定]→[一般]→[詳細] に移動して、HTTP アドレスのうちアプリケーション・パスの前に来る部分を変更できます (“InterSystems Web アプリケーション URL の構造” を参照してください)。 InterSystems IRIS での CSP ファイルの使用はお勧めしません。
コードの展開	一部のファイルでのみ使用できます。テキスト・エディタ・ウィンドウで完全なコードを表示します。
コードの縮小	一部のファイルでのみ使用できます。テキスト・エディタ・ウィンドウで一部のコード・セクションを縮小します。プラス・アイコンを選択すると、そのセクションが展開されます。
クラス・ドキュメントの表示	クラス定義ファイルで作業している場合、またはクラスが選択されている場合にのみ使用できます。(保存済みの) クラス・メンバの説明から導出された、現在のクラスのオンライン・ドキュメントを表示します。
言語モード	.INT や .MAC などソース・コード・ファイルで作業している場合にのみ使用できます。 InterSystems IRIS 言語バージョンのリストから、サイトに適したモードを選択します。

16.3.1 ツールバー

[ツールバー] メニューでは、ツールバーの表示/非表示を切り替えたり、ツールバーをカスタマイズしたりできます。

メニュー・オプション	アクション
標準	標準ツールバーの表示/非表示を切り替えます。
デバッグ	デバッグ・ツールバーの表示/非表示を切り替えます。
メンバ	クラス・メンバ・ツールバーの表示/非表示を切り替えます。
ステータス・バー	スタジオ・ウィンドウの下部にあるステータス・バーの表示/非表示を切り替えます。ステータス・バーの左側にはステータス・メッセージが表示され、右側にはカーソルの位置を示す行と列が表示されます。右側の 4 つのボタンは、ハイライト表示されている場合、Caps Lock キーがオンであること、Num Lock キーがオンであること、Insert キーがオンであること(上書き)、および現在のファイルが読み取り専用ファイルであることを示します。ステータス・バーには、[ファイルから検索] と [出力] ウィンドウの行番号と列番号も表示されます (該当する場合)。
ブックマーク	ブックマーク・ツールバーの表示/非表示を切り替えます。
カスタマイズ	[カスタマイズ] ダイアログを開きます。

以下では、テキスト・ラベルが表示された各種のツールバーを示しています。スタジオでテキスト・ラベルを表示するには、**[ビュー]→[ツールバー]→[カスタマイズ]** を選択して、[ツールバー] タブを開きます。任意のツールバーを選択して、**[テキスト・ラベルを表示]** を選択します。(いずれかのツールバーが既に選択されている場合は、そのツールバーのチェックを外してから再びチェックを付けて、**[テキスト・ラベルを表示]** にチェックを付けます)。

図 16-1: 標準ツールバー

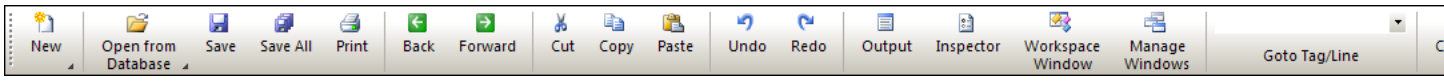
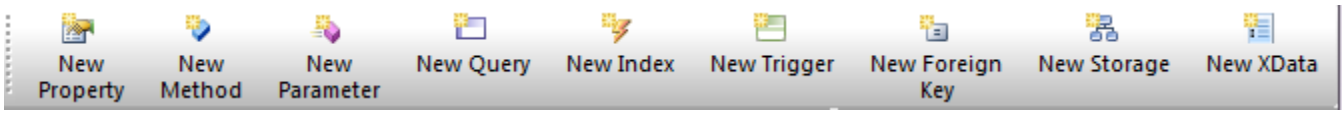


図 16-2: デバッグ・ツールバー

図 16-3: クラス・メンバ・ツールバー



BPL ツールバーは InterSystems IRIS でのみ使用できます。

図 16-4: BPL ツールバー

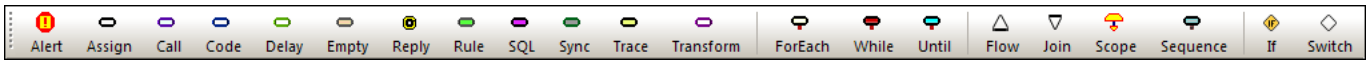
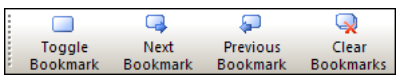


図 16-5: ブックマーク・ツールバー



16.3.2 カスタマイズ・ツールバー

[カスタマイズ] メニューは、スタジオ・インタフェースの構成要素をカスタマイズするための [コマンド]、[ツールバー]、[ツール]、および [オプション] という 4 つのタブで構成されています。

メニュー・オプション	アクション
コマンド	スタジオの各種メニューおよびこれらのメニュー上のすべてのコマンドをリスト表示します。このリスト内のコマンドは、表示されているツールバーにドラッグ・アンド・ドロップして追加できます。ツールバーからコマンドを削除するには、そのコマンドをツールバーの外にドラッグ・アンド・ドロップします。
ツールバー	ツールバーを表示するには、そのツールバーのチェック・ボックスにチェックを付けます。ツールバーにテキスト・ラベルを表示するには、[テキスト・ラベルを表示] を選択します。メニュー・バーのチェックを外すことはできません。新規ツールバーを作成するには、[新規作成] を選択します。
ツール	スタジオの [ツール] メニューにメニュー項目を追加します。項目名、そのコマンド、引数 (省略可)、および初期ディレクトリを指定します。
オプション	画面ヒント、ショートカット・キー付きの画面ヒント、および大きいアイコンを表示するには、該当するチェック・ボックスにチェックを付けます。

16.4 [プロジェクト] メニュー

[プロジェクト] メニューには、プロジェクトを操作するオプションがあります。

メニュー・オプション	アクション
アイテム追加	現在のエディタ・ウィンドウの項目を、現在のプロジェクトに追加します。
アイテム消去	現在のエディタ・ウィンドウの項目を、現在のプロジェクトから削除します。
設定	以下を使用して現在のプロジェクトの設定を編集します。 ・ 一般 : このプロジェクトの Web ページを設定するためにスタジオが使用する HTTP アドレス : HTTP アドレスのうち、このプロジェクトの Web ページのアプリケーション・パスの前に来る部分を設定します (″InterSystems Web アプリケーション URL の構造″ を参照してください)。定義 : プロジェクトのコンパイル時に適用されるマクロを定義します。他のマクロによってテストされるデバッグ・マクロを定義して、コード上で追加のチェックを有効にできます。例えば、debug=1 は、マクロ $$$debug が 1 となるよう定義します。 ・ デバッグ : [デバッグ対象] と [ブレークポイント] を設定します。詳細は、″スタジオ・デバッガの使用″ を参照してください。

16.4.1 一般的なプロジェクト・タスク

プロジェクトを削除するには、[ファイル]→[プロジェクトを開く] を選択して、削除するプロジェクトを右クリックします。

プロジェクトをインポートするには、[ツール]→[ローカルからインポート] を選択して、そのプロジェクトが含まれた .xml ファイルを選択します。

プロジェクトをエクスポートするには、プロジェクトを開き、[ツール]→[エクスポート] を選択して、[現在のプロジェクトをエクスポート] を選択して、出力ディレクトリに移動して、ファイル名を入力します。

プロジェクト名には、ピリオド (.)、コンマ (,)、セミコロン (;)、スラッシュ (/)、円記号 (¥) は使用できません。

16.5 [クラス] メニュー

[**クラス**] メニューには、クラス定義を編集するオプションがあり、クラス定義ファイルで作業している場合にのみ使用できます。

クラス・オプションには [**追加**] サブメニューと [**オーバーライド**] サブメニューがあります。以下のようなオプションがあります。

メニュー・オプション	アクション
追加	[プロパティ]、[メソッド]、[クラス・パラメータ]、[クエリ]、[インデックス]、[SQLトリガ]、[外部キー]、[ストレージ]、[プロジェクション]、または [XData] のいずれかを選択して項目のウィザードを開き、現在のクラス定義に項目定義を挿入します。各クラス・メンバのオプションの詳細は、該当する章を参照してください。
リファクタまたはオーバーライド	(リファクタリングは、スタジオが Windows サーバに接続されている場合にのみ利用できます(その他のプラットフォーム上ではオーバーライドされます)。いくつかの機能は部分的にWindows以外のプラットフォーム上で機能しますが、予測できない結果となることがあるため、これらの機能は使用しないでください。) オーバーライド : ウィンドウに、現在のクラスが継承する項目がリストされ、このリストから項目を選択します。選択したクラス定義にオーバーライド定義を挿入します。項目には [プロパティ]、[メソッド]、[パラメータ]、[クエリ]、[SQL トリガ]、[XData] ルーチンがあります。 名前変更 : 新しい名前を入力します。新しい名前により、ドキュメント内のすべての場所にある古い名前が置き換えられます(スタジオは埋め込み SQL 文や列名の文字列を受け取るメソッド内などのリテラル文字列内のコードはリファクタしません)。.[ストレージのリセット](クラスの場合)または [新規ストレージ・スロット](プロパティの場合)にチェックを付けると、名前が変更された項目がストレージなしで作成され、既定のストレージが生成されます。古いクラスを削除するとそのクラスのストレージ・エクステント定義も削除されることを確認するボックスが表示されます。 リファクタリングは、データベースに対する変更の適用を、すべてのドキュメントの変更の確認が終了するまで延期します。[変更を適用] を選択すると、変更が一時的な場所に保存されます。[完了] を選択すると、すべての変更が適用されます。[完了] を選択しても、一部のドキュメントの変更が受け入れなかった場合は、[?] というプロンプトが表示されます。ユーザは他のドキュメントの変更を適用することも、終了することもできます。 注意 : リファクタリングは運用環境では使用しないでください。 開発中にのみ使用してください。 スタジオは、一切のデータ操作を許可しません。
スーパークラス	現在のクラスのスーパークラスをアルファベット順にリスト表示します。このリストから選択して [プロジェクトに追加]、[ドキュメントの表示]、または [開く] を実行できます。
派生クラス	現在のクラスから派生したクラス(サブクラス)をリスト表示します。このリストから選択して [プロジェクトに追加]、[ドキュメントの表示]、または [開く] を実行できます。
サブクラスの作成	新規クラス・ウィザードが表示されます。このオプションを使用して作成されたクラスは、現在のウィンドウのクラスの派生クラスになり、そのメンバ(プロパティ、メソッド、クラス・パラメータ、使用可能なクラス・キーワード、および継承されたプロパティと継承されたメソッドのパラメータとキーワードなど)を継承します。

クラスをコンパイルすると、プロジェクションは自動的に他の言語の関連ファイルを生成します。例えば、タイプ %Projection.Java のプロジェクションを追加すると、コンパイル時に、使用しているクラスに対応する新規の Java クラスが生成されます (これにより、Java アプリケーションからクラスを使用できるようになります)。

16.6 [ビルド] メニュー

[ビルド] メニューには、アプリケーションのコンパイルやビルドを行うオプションがあります。[ビルド] オプションの動作は、スタジオの [オプション] ダイアログ ([ツール]→[オプション]) のコンパイル設定で指定します。

ビルド・オプションには次のものがあります。

メニュー・オプション	アクション
 コンパイル	現在のウィンドウのコンテンツをコンパイルします。[ツール]→[オプション] の [コンパイル] タブの設定を使用します。コンパイラからのすべてのメッセージすべては、出力ウィンドウに表示されます。 [関連最新アイテムをスキップ] にチェックが付いている場合 : · 現在のクラス定義が変更されている場合にだけ、そのクラス定義はコンパイルされます。 · 可能な場合、スタジオは増分コンパイルを実行します。クラス定義への変更が 1 つまたは複数のメソッドの実装のみに限られる場合、これらのメソッドのみコンパイルされます。
コンパイル・オプション	このセッションのオプションのみを選択するか、既定のコンパイル・オプションを変更するときに使用します。[ツール]→[オプション]→[コンパイル] タブでも、既定オプションを設定できます。
 すべて再ビルド	前のコンパイルから変更されたかどうかにかかわらず、現在のプロジェクト内のすべてのコンポーネントをコンパイルします。

16.7 [デバッグ] メニュー

[デバッグ] メニューには、デバッグ・オプションがあります。[デバッグ] メニューのオプションについては、"[デバッグ] メニュー" を参照してください。"キーボード・アクセラレータ" も参照してください。

16.8 [ツール] メニュー

[デバッグ] メニューには、その他のオプションがあります。

オプションには以下のものがあります。

メニュー・オプション	アクション
クラス・ブラウザ	スタジオ・クラス・ブラウザをオープンします。クラス・ブラウザは、現在のネームスペース内のクラスと、(定義済み、および継承済みの) クラス・メンバのすべてのリストを表示します。

メニュー・オプション	アクション
SQL文に対するプラン表示	SQL 文のダイアログを開きます。このダイアログでは、アクティブなドキュメント内で選択された SQL 文が表示されて編集可能になります。[プラン表示] を選択すると、クエリ実行プランが Web ページとして表示されます。
テンプレート	スタジオ・テンプレートのリストを表示します。テンプレートは、現在のカーソル位置にテキストのストリームを挿入します。スタジオには複数のテンプレートが用意されています。また、独自のテンプレートを作成することもできます。詳細は "スタジオ・テンプレートの使用法" を参照してください。
アドイン	開いているスタジオ・ファイルに項目を追加したり、標準形式を使用して既存のファイルに接続したりするのに役立つウィザードがリスト表示されています。"アドイン・テンプレート" を参照してください。
タスク・リスト	タスクのリストを表示します。新規タスク・ウィンドウでは、タスクを追加、編集、または削除できます。各タスクには、サーバ、ネームスペース、ドキュメント名、行番号、およびオプションの説明が含まれます。 現在の行コードまたは選択されているテキストが、既定でタスクの説明に使用されます。[移動] をクリックすると、選択したドキュメントおよび行番号に移動できます。必要に応じて、スタジオはタスクで指定されるサーバ/ネームスペースに、現在のセキュリティ資格情報を使用して接続します。
エクスポート	1 つまたは複数の項目 (クラス定義、プロジェクト、ルーチン、インクルード・ファイル) をローカル・ファイル、またはサーバ・システム上のファイルにエクスポートします。
特殊エクスポート	%RO ユーティリティで作成された形式の RO をエクスポートします。
リモートからインポート	リモート・ファイル (スタジオが接続されているサーバと同じマシン上のファイル) から項目をインポートできます。インポートされたすべての項目は、新規ドキュメント・ウィンドウに配置されます。 このオプションを使用して、XML または .RTN (%RO ルーチン形式ファイル) からプロジェクト、クラス定義、ルーチン、またはインクルード・ファイルをインポートできます。 [リモートからインポート] オプションは、選択可能なファイルに含まれるすべての項目をリストしたダイアログを表示します。また、インポートした項目を現在のプロジェクトに追加するかどうか、およびそれらの項目をコンパイルするかどうかを指定することもできます。 手アイコン は、インポートするファイルがシステム上のファイルよりも古いことを示します。
ローカルからインポート	ローカル・ファイル (スタジオと同じマシン上にあるファイル) から項目をインポートできます。インポートされたすべての項目は、新規ドキュメント・ウィンドウに配置されます。 このオプションを使用して、XML または .RTN (%RO ルーチン形式ファイル) からプロジェクト、クラス定義、ルーチン、またはインクルード・ファイルをインポートできます。 [ローカルからインポート] オプションは、ファイルに含まれるすべての項目をリストにしたダイアログを表示します。このダイアログを使用すると、インポートする項目を選択できます。また、インポートした項目を現在のプロジェクトに追加するかどうか、およびそれらの項目をコンパイルするかどうかを指定することもできます。インポートする項目のリストは、32 K 未満にする必要があります。 手アイコン は、インポートするファイルがシステム上のファイルよりも古いことを示します。

メニュー・オプション	アクション
比較	開いているファイルと、[参照] で選択したファイルを比較します。[ツール]→[オプション]→[環境]→[一般] で [比較] 設定を使用して、外部比較ツールを指定済みである必要があります。正しく動作するには、比較ツールでコマンド行パラメータ (tool.exe file1 file2) を受け入れることができなければなりません。テスト済みの比較ツールは、Microsoft Windiff と Perforce p4Diff.exe です。
クラスをコピー	既存のクラスのコピーを作成し、それに新規の名前を付けることができます。
C++ プロジェクションの生成	このオプションは、Cache および Ensemble にのみ適用されます。
設定のインポートとエクスポート	スタジオ設定のインポートとエクスポート・ウィザードを開きます。このウィザードでは、インポートおよびエクスポートするファイルおよびディレクトリを設定し、それらの設定を regedit.exe と互換性のある形式のテキスト・ファイルに保存できます。 このファイルは、任意の Windows コンピュータで、ウィザードを使用してインポートすることも、コマンド・プロンプトまたはエクスプローラから直接実行することもできます。設定のインポートおよびリセットによって、スタジオが再起動します。Windows Vista へのインポートを実行するには、管理者特権が必要です。
オプション	スタジオのオプションを設定できます。オプションの詳細は、"スタジオのオプション" を参照してください。
カスタマイズ	[カスタマイズ] ウィンドウを開きます。ここでは、メニューやツールバーなど、スタジオの UI についてカスタマイズできます。

16.9 [ユーティリティ] メニュー

[ユーティリティ] メニューには、次のようなスタジオ外部のリソースへのリンクがあります。

- 管理ポータル
- Telnet

16.10 [ウィンドウ] メニュー

[ウィンドウ] メニューには、スタジオのウィンドウを操作する標準オプションがあります。

16.11 [ヘルプ] メニュー

[ヘルプ] メニューには、オンライン・ヘルプにアクセスするオプションがあります。

オプションには以下のものがあります。

メニュー・オプション	アクション
スタジオのドキュメント	スタジオ・ドキュメントの目次を表示します。
スタジオ・コマンド	スタジオ・オプションとその簡単な説明のリストを表示します。
オンライン・ドキュメント	InterSystems IRIS オンライン・ドキュメントのホーム・ページを表示します。
ObjectScript リファレンス	ObjectScript のオンライン・リファレンスを表示します。
SQL リファレンス	InterSystems IRIS SQL のオンライン・リファレンスを表示します。
クラス定義シンタックス	クラス定義構文のオンライン・リファレンスを表示します。
InterSystems Web ページ	インターシステムズの Web サイト上の便利なページへのリンクを提供します。
スタジオについて	スタジオのバージョン情報を表示します。

16.12 コンテキスト・メニュー

スタジオの各領域を右クリックすると、領域ごとに異なるコンテキスト・メニューが表示されます。以下のセクションでは、これらのコンテキスト・メニューについて説明しています。

16.12.1 エディタ・コンテキスト・メニュー

スタジオのエディタ・ウィンドウを右クリックすると、コンテキスト・メニューが表示されます。このコンテキスト・メニューには、[エディタ] メニューの [切り取り]、[コピー]、[貼り付け]、[検索]、[ブレークポイント切替]、[戻る] など、メイン・メニューから利用できる多くの項目が含まれています。メイン・メニューからは利用できない追加のオプションも含まれています。これには、以下のものがあります。

メニュー・オプション	アクション
ヘルプ	[ヘルプ] オプションを使用すると、選択された構文要素に関するコンテキスト・ヘルプを表示できます。このオプションを使用するには、任意のテキストを右クリックして [ヘルプ] を選択します。 例えば、ObjectScript コードで Do という単語を右クリックし、[ヘルプ] を選択した場合、Do コマンドのリファレンス・ページがブラウザで表示されます。
タスクの追加	タスクをタスク・リストに追加できるタスクの追加・ウィンドウを開きます。"[ツール] メニュー" も参照してください。
シンタックスの色を設定する	特定の構文要素を表示するときに使用される色を選択するダイアログを表示します。 このオプションを使用するには、任意のテキストを右クリックして [シンタックスの色を設定する] を選択します。
<TAG> へ移動	ObjectScript ルーチンの編集時に使用可能です。ObjectScript タグを定義するコードに移動できます。 このオプションを使用するには、ObjectScript ルーチンで任意のタグを右クリックし、[<TAG> へ移動] を選択します。右クリックしたタグが他のルーチンで定義されている場合、そのルーチンが自動的に開かれます。
デバッグ対象として current item を設定	現在の項目をデバッグ対象として設定します。
単語のハイライト表示の切り替え	ドキュメント内でカーソルが指している単語のすべてのインスタンスをハイライト表示します。

16.12.2 ワークスペース・コンテキスト・メニュー

ワークスペース・ウィンドウを右クリックすると、コンテキスト・メニューが表示されます。表示されるメニューはカーソルの位置によって異なります。カーソルがパッケージ、クラスなどのいずれに配置されているかに応じて、それぞれ異なるコンテキスト・メニューが表示されます。次のテーブルでは、ワークスペース・パッケージのコンテキスト・メニューの項目のうち、メニュー・バーから利用できないものを示しています。

メニュー・オプション	アクション
追加	表示されたリストから選択されたクラスを現在のパッケージに追加します。
パッケージ “名前“ の削除	現在のパッケージをこのプロジェクトから削除します。
パッケージ “名前“ のコンパイル	現在のパッケージをコンパイルします。
パッケージ “名前“ の削除	現在のパッケージを削除します。
エクスポート	現在のパッケージを XML ファイルにエクスポートします(パッケージをインポートするには、[ツール]→[インポート] を選択してから XML ファイルを選択します)。
プロジェクトに追加	ハイライト表示された項目を現在のプロジェクトに追加します。
ソース・コントロール	[ソースコントロール] オプションの [チェックアウト]、[チェックアウトをアンドゥ]、[チェックイン]、[最新を取得] から選択します。 詳細は、″InterSystems IRIS とソース・コントロール・システムの統合″ を参照してください。
閉じる	現在のドキュメントを閉じます。
自分以外すべて閉じる	ハイライト表示されているドキュメント以外のすべてのドキュメントを閉じます。
すべて閉じる	すべてのドキュメントを閉じます。

16.12.3 インスペクタ・コンテキスト・メニュー

インスペクタ・ウィンドウを右クリックすると、次の項目が含まれたコンテキスト・メニューが表示されます (エディタ・ウィンドウ内の現在のドキュメントに適用可能な項目のみが表示されます)。

メニュー・オプション	アクション
追加	表示されたリストから選択されたメンバを現在のクラス定義に追加します。
Override (オーバーライド)	ウィンドウが開いて、現在のクラスから継承された項目のリストが表示され、このリストから項目を選択できます。オーバーライド定義は、現在のクラス定義ウィンドウに挿入されます。
デフォルト値に戻す	選択された項目を既定値にリセットします。
削除	表示された項目を削除します。
検索	クラス・メンバが表示されている場合に使用できます。そのメンバをエディタ・ウィンドウに表示します。
継承されたメンバを表示する	継承されたメンバをウィンドウに表示するかどうかを切り替えます。
ヘッダ表示	列ヘッダ ([名前] と [値]) をインスペクタ・ウィンドウに表示するかどうかを切り替えます。

16.12.4 タブ・コンテキスト・メニュー

タブ・ヘッダを右クリックすると、以下の項目が含まれたコンテキスト・メニューが表示されます。

メニュー・オプション	アクション
閉じる	現在のタブを閉じます。
自分以外すべて閉じる	現在のタブ以外のすべてのタブを閉じます。
すべて閉じる	すべてのタブを閉じます。

16.12.5 ウィンドウ表示のコンテキスト・メニュー

他のコンテキスト・メニューが対応付けられていないウィンドウで右クリックすると、以下の項目が含まれた一般のウィンドウ・コンテキスト・メニューが表示されます。

メニュー・オプション	アクション
浮動	選択されたウィンドウを固定位置から分離して、希望の位置に自由にドラッグできるようにします。
固定	選択されたウィンドウを既定の位置に固定します。
隠す	選択されたウィンドウを非表示にします。

16.12.6 デバッガ・ウォッチ・コンテキスト・メニュー

デバッガ・ウォッチ・ウィンドウ・コンテキスト・メニューについては、"デバッガ・ウォッチ・コンテキスト・メニュー" を参照してください。

16.13 キーボード・アクセラレータ

以降のセクションでは、スタジオのキーボード・アクセラレータ (キーボード・ショートカット) (押したときに特定のスタジオ機能を実行するキーの組み合わせ) を示します。

テキストのブロック

以下の表では、テキストのブロックはいくつかの完全な行を表します。テキストのブロックを選択するには、キャレットを最初の行の先頭に配置し、Shift キーを押して、関連するすべての行がハイライト表示されるまで**下矢印**を選択します。キャレットは、最後の完全な行の下の行の先頭に表示されます。

16.13.1 一般

アクセラレータ	動作
F1	ヘルプ
F4	ネームスペースまたは接続の変更
F8	スタジオのメニューおよびエディタ・ウィンドウの全画面表示の切り替え
Ctrl-N	新規ドキュメント
Ctrl-O	ドキュメントを開く
Ctrl-Shift-O	プロジェクトを開く
Ctrl-P	印刷 [印刷] ダイアログが開きます。テキストが選択されていると、[Selection] にチェックが付いています。
Ctrl-S	保存
Ctrl-Shift-I	エクスポート
Ctrl-I	ローカルからインポート

16.13.2 表示

アクセラレータ	動作
Ctrl-+	すべて展開 ドキュメント内の展開可能なセクションをすべて展開します。 マイナス・アイコンを選択すると、1 つのセクションが折りたたまれ、Ctrl+- を押すと、すべてのセクションが折りたたまれます。
Ctrl を押しながらプラス・アイコンを選択	すべてのブロック・セクションを展開 このコード・ブロック内の展開可能なセクションをすべて展開します。 マイナス・アイコンを選択すると、1 つのブロックが折りたたまれ、Ctrl+- を押すと、すべてのブロックが折りたたまれます。
Ctrl--	すべて非表示 非表示にできるすべてのセクションを非表示にします。
Ctrl-W	クラス・ブラウザの表示
Ctrl-Shift-V	他の表示 MAC ルーチンまたは INT ルーチンなど、現在のドキュメントに関連するドキュメントを開きます。
Alt-1	インスペクタ・ウィンドウ表示の切り替え

アクセラレータ	動作
Alt-2	出力ウィンドウ表示の切り替え 出力ウィンドウには、[結果] と [ファイルから検索] に使用できるタブがあります。
Alt-3	ワークスペース・ウィンドウ表示の切り替え
Alt-4	ウォッチ・ウィンドウ表示の切り替え
Alt+5	コード・スニペット・ウィンドウ表示の切り替え
Alt+6	[ファイルから検索] ウィンドウ表示の切り替え
Alt+7	クラス・ビュー・ウィンドウ表示の切り替え
Ctrl-Alt-+	フォントの拡大 (Ctrl キーと Alt キーと + キーを同時に押します)
Ctrl-Alt--	フォントの縮小 (Ctrl キーと Alt キーと - キーを同時に押します)
Ctrl-Alt-Space	空白記号、スペース、改行文字、およびタブの表示切り替え
Ctrl-B	ブラケット・マッチングの切り替え 現在のウィンドウのブラケット・マッチングのオンとオフを切り替えます。
Ctrl-Shift-N	行番号表示の切り替え
Ctrl-Tab	新規ウィンドウ
Ctrl-Shift-Tab	前のウィンドウ

16.13.3 移動

アクセラレータ	動作
Home	行頭に移動 連続して押すと、行の先頭と行のテキストの先頭の間でキャレットが移動します。
Ctrl-Home	ドキュメントの先頭に移動
End	行末に移動
Ctrl-End	ドキュメントの末尾に移動
Ctrl--	戻る
Ctrl-Shift--	進む
Page Up	前ページ
Page Down	次ページ
Ctrl-PageUp	表示ページの先頭に移動
Ctrl-PageDown	表示ページの末尾に移動
Ctrl-↓	下方スクロール
Ctrl-↑	上方スクロール

アクセラレータ	動作
Ctrl-G	Goto
Ctrl-Shift-G または F12	タグのドキュメントに移動
Ctrl-F3	次のエラーに移動
Ctrl-Shift-F3	前のエラーに移動
Alt-F3	次の警告に移動
Alt-Shift-F3	前の警告に移動
Ctrl-]	ブラケットへ移動 最も内側のブラケットのペア（小括弧または中括弧など）間でカーソルを移動します。入れ子になっている場合、3 種類すべてのペアを（1 つずつ）ハイライト表示できます。このアクセラレータは、ブラケット・マッチングがオンにされている場合のみ機能します（Ctrl-B を参照してください）。

16.13.4 編集

アクセラレータ	動作
Insert	挿入/上書きモードの切り替え 挿入モード（文字の入力時に新規の文字が挿入される）と上書きモード（文字の入力時に新規の文字が既存の文字と置換される）を切り替えます。
Ctrl-Delete	次の語の削除または語尾までの削除 キャレットが語頭にある場合、その語を削除します。キャレットが語中にある場合、キャレットから語尾までを削除します。
Ctrl-Backspace または Ctrl-Shift-Delete	前の語の削除または語頭までの削除 キャレットが語尾にある場合、その語を削除します。キャレットが語中にある場合、キャレットから語頭までを削除します。
Ctrl-Shift-L	行の削除
Ctrl-C または Ctrl-Insert	コピー
Shift-Delete または Ctrl-X	切り取り
Ctrl-L	行の切り取り
Ctrl-V または Shift-Insert	貼り付け
Ctrl-A	全て選択
Ctrl-Y または Ctrl-Shift-Z	やり直し
Ctrl-Z	元に戻す
Ctrl-Space	ポップアップの表示 カーソルが適切な位置にある場合、スタジオ・アシスト・ポップアップが表示されます。このポップアップには、その位置で利用可能なオプションが表示されます（必要に応じて、クラス、メソッド、プロパティなど）。

アクセラレータ	動作
Ctrl-~	タブ展開の切り替え Tab を押したときのタブの入力とスペースの入力を切り替えます。
Ctrl-U	大文字選択
Ctrl-Shift-U	小文字選択
Ctrl-Alt-O	[解析の遅延] オプションを切り替えます。
Ctrl-Alt-U	語頭の大文字選択
Ctrl-(	開始および終了括弧の挿入 (ドイツとスイスのキーボードでは機能しません*)
Ctrl-{	開始および終了中括弧の挿入
Ctrl-[	開始および終了角括弧の挿入
Ctrl-<	開始および終了山括弧の挿入
Ctrl-=	インデントのクリーンアップ。選択されたテキストの行全体のブロックに対するインデントをクリーンアップします。
Ctrl-/	行のコメント化 選択した行の最初に # (シャープ記号) を追加して、コメント化します。
Ctrl-Shift-/	行のコメント化解除 選択したコメント行の最初から # (シャープ記号) を削除して、通常の行にします。
Ctrl-/	テキストのコメント・ブロック テキストのブロックが選択されているときに Ctrl+/ を押すと、テキストのブロックがコメント化されます。つまり、各行頭に #; (シャープとセミコロン) が追加される (ObjectScript ルーチンの場合) か、ドキュメントの言語に基づいて適切なマーカが追加されます(ドイツとスイスのキーボードでは機能しません*)。
Ctrl-Shift-/	テキストのアンコメント・ブロック テキストのブロックが選択されているときに Ctrl-Shift-/ を押すと、テキストのブロックがアンコメントされます。つまり、各行頭の #; が削除される (Objectscript ルーチンの場合) か、ドキュメントの言語に基づいて他のマーカが削除されます(ドイツとスイスのキーボードでは機能しません*)。
Ctrl+Alt+/	テキスト・ブロックへのコメント・マーカの追加 ブロック・コメントがサポートされている場合、特定の言語のブロック・タイプ・コメント (/*...*/ など) を挿入します。ブロック・タイプ・コメントが存在しない場合は、1 行のコメント・マーカが挿入されます(ドイツとスイスのキーボードでは機能しません*)。
Ctrl+Shift+Alt+/	テキスト・ブロックからのコメント・マーカの削除 ブロック・コメントがサポートされている場合、特定の言語のブロック・タイプ・コメント (/*...*/ など) を削除します。ブロック・タイプ・コメントが存在しない場合は、1 行のコメント・マーカが挿入されます (ドイツとスイスのキーボードでは機能しません*)。

アクセラレータ	動作
Ctrl-E	ObjectScript ドキュメントで、選択範囲内のコマンドが正式名に置き換えられます。
Ctrl-Shift-E	圧縮コマンド ObjectScript ドキュメントで、選択範囲内のコマンドが省略名に置き換えられます。
Ctrl-.	ドットの挿入 テキストのブロックを選択した場合、各行頭 (先頭の空白の後) にドットが挿入されます。行は、先頭の空白で開始する必要があります。引数なしの DO コマンドと先頭にピリオドを使用したブロック構造化で使用するためのものです。
Ctrl-Shift-.	ドットの削除 選択したテキストのブロックの先頭から先頭のドットを削除します (引数なしの DO コマンドと先頭にピリオドを使用したブロック構造化で使用)。
Ctrl-Shift-T	タスクの追加

16.13.5 検索と置換

アクセラレータ	動作
Ctrl-F	検索
F3	次を検索
Shift-F3	前を検索
Ctrl-Shift-F	ファイルから検索
Ctrl-, (コンマ)	検索
Ctrl-H	置換
Ctrl-Shift-G	移動
Ctrl-Alt-G	戻る

16.13.6 ブックマーク

アクセラレータ	動作
Ctrl-F2	現在の行のブックマークの切り替え
F2	次のブックマークに移動
Shift-F2	前のブックマークに移動
Ctrl-Shift-F2	すべてのブックマークのクリア

16.13.7 ビルドとコンパイル

アクセラレータ	動作
F7	プロジェクト内のすべてのドキュメントを再ビルド
Ctrl-F7	アクティブ・ドキュメントのコンパイル
Ctrl-Shift-F7	コンパイル・オプション
F5	ブラウザで表示

16.13.8 デバッグ

アクセラレータ	動作
Ctrl-Alt-L	スタジオ・デバッグ・ロギングの切り替え ロギングを有効または無効にします。有効な場合、スタジオ・デバッグを目的として、情報がファイル /irisinstall/bin/CD###.log に送信されます。このファイルは非常に大きくなる可能性があります。注意して使用してください。
Ctrl-Shift-A	デバッグのアタッチ デバッガをプロセスにアタッチします。
F9	現在の行のブレークポイント切替のデバッグ
Ctrl-F5	デバッグ開始
Ctrl-Shift-F5	デバッグ再開
Ctrl-F10	カーソル位置までデバッグ実行 プログラム実行を再開し、カーソルがある行またはブレークポイントで実行を停止します。
F11	デバッグのステップイン ブレークまたは中断から、次のループにステップインします。
Shift-F11	デバッグのステップアウト 現在のプロセスからステップアウトします。
F10	デバッグのステップオーバ 次のプロセスをスキップします。
Shift-F5	デバッグ停止

16.13.9 テンプレート

アクセラレータ	動作
Ctrl-Shift-1 ～ Ctrl-Shift-9	テンプレートを開く スタジオ・テンプレートのアクセラレータとして使用できます。アクセラレータを設定するには、`accelerator="#"` 形式の属性をテンプレートに追加します (`csp:StudioInteractiveTemplate` タグ、または `csp:StudioSimpleTemplate` タグに追加)。これにより、テンプレートにアクセラレータ Ctrl-Shift-# が設定されます。数字 (#) は 0 ～ 9 です。
Ctrl-T	テンプレートを開く

16.13.10 ウィザード : 新規メソッド・ウィザードの引数と新規クエリ・ウィザードのパラメータ

アクセラレータ	動作
Alt-A	追加
Alt-R	削除
Alt-U	上へ
Alt-D	下へ

16.14 スタジオ・メニューの追加

スタジオ・メニューにメニュー項目を追加するには、次の手順を実行します。

1. スタジオ・ツールバーでメニュー名を右クリックして、[**カスタマイズ**] を選択します。
2. [**ツール**] タブを選択して、.exe ファイルを追加します。

17

スタジオ・オプションの設定

[**ツール**] > [**オプション**] を選択して、スタジオの動作を変更できます。

[**オプション**] ダイアログのタブの詳細は、以下のセクションで説明します。

17.1 環境オプション

次のオプションはスタジオの環境を制御します。

一般

優先言語 : 新規クラスを作成するために使用される既定の言語バージョンを設定します。

最近使用したリストに表示される項目 : 最近使用されたファイルのリスト ([**ファイル**] メニュー内) に表示する項目数を指定します。

プロジェクトに追加されたファイルを開く : チェックが付いている場合、現在のプロジェクトにファイルを追加すると、スタジオでそのファイルが自動的に開かれます。

タブ化ドキュメント・セレクタ : 有効な場合、開いている各ドキュメントのタブが表示されます。行をスタジオ・ウィンドウの上部または下部のどちらに表示するかを選択できます。

プロジェクトを開く時に前回開いていたドキュメントを開く : チェックが付いている場合、スタジオを起動すると、現在のネームスペースに前回にいたときに開かれていたすべてのドキュメントが再び開かれます。このオプションを省略するには、Shift キーを押しながらスタジオを開きます。

操作完了後に検索と置換ウィンドウを隠す : チェックが付いている場合、[**検索と置換**] ダイアログでのタスクが完了すると、このダイアログが閉じられます。

比較 : [**ツール**]→[**比較**] を使用して、このドキュメントと比較するドキュメントを選択します。

フォント

以下のウィンドウおよび印刷で使用されるフォントのサイズや書体を指定します : **エディタ・ウィンドウ**、**出力ウィンドウ**、**印刷**、**ワークスペース**、**インスペクタ**。それぞれのウィンドウで任意の Windows フォントを使用できますが、書体とサイズは一種類に制限されています。

キーボード

含まれるコマンドを表示 : 検索するコマンドを指定します。

選択したコマンドのショートカット ： 選択したコマンドにショートカット・キーが割り当てられているかどうかを示します。

新しいショートカット・キーを押す ： 選択したコマンドに割り当てるショートカット・キーを入力します。

現在使用されているショートカット ： 入力したショートカット・キーが現在別のコマンドに割り当てられているかどうかを示します。

すべてリセット ： すべてのキーボード・ショートカットを元の設定にリセットします。

削除 ： 選択したコマンドの既存のショートカット・キーを削除します。

[開く] ダイアログ

最後のマスクを自動的に適用する :チェックを付けると、最後の検索マスクが **[ファイル]→[開く]** ダイアログで自動的に使用されます。

追加の専用サーバ・プロセスを使用する :ユーザがデータ収集を中止するオプションが必要な場合に、きわめて大規模なシステム上でのみ使用することをお勧めします。 きわめて大規模なシステムでは、まれに、**[ファイル]→[開く]** ダイアログでファイルを検索すると非常に時間がかかることがあります。これを解決するには、このオプションにチェックを付けて、別のプロセスで検索を実行します。検索が 0.2 秒より長くかかる場合、スタジオは **[キャンセル]** ボタンを備えた進捗バーを表示します。このボックスにチェックを付け、追加のプロセスが開始された場合、ライセンス・カウントに影響します。

サーバ定義の色

ソフトウェア・インスタンスのステータス・パネルの背景色またはドキュメントの背景色を選択できます。カラー・パレットを表示するには、インスタンスの右端にある四角を選択します。カラー・スウォッチおよび 16 進数のカラー・コードで表示されるパレットから色を選択します。

ドキュメントおよびプロキシ

オンライン・ドキュメントを表示する HTTP アドレス ： スタジオがオンライン・ドキュメントの取得に使用する場所を指定します。インスタンスに関連付けられている既定の場所を使用する場合は **[自動]**、別の HTTP アドレスを指定する場合は **[HTTP アドレス]** を指定します。アドレスのうちアプリケーション・パスの前に来る部分を入力します。"InterSystems Web アプリケーション URL の構造" を参照してください。

テンプレートおよびアドインは 'IRIS_instance' のプロキシ・サーバを使用する ： 現在のインスタンス用のテンプレートおよびアドインのロードに使用するプロキシ・サーバのサーバ・アドレスとポート番号を指定します。**[アドレス]** フィールドでは、`http://` と `https://` のアドレス形式、および `localhost` のようなアドレスがサポートされています。アドレス内に `://` が検出されない場合、スタジオでは `http://` が自動的に追加されます。

クラス

複数行のメソッド引数 ： このオプションを選択した場合、メソッド引数はクラス・エディタで 1 行に 1 つずつ表示されます。[複数行のメソッド引数] が有効な場合に [ファイルから検索] を使用し、複数行のメソッド引数のあるファイルの [ファイルから検索] の出力の行を選択すると、カーソルは間違った行番号に移動します。これが問題になる場合は、[複数行のメソッド引数] を無効にします。

Option explicit ： このオプションを選択した場合、#DIM を使用して宣言されていない変数を参照すると構文エラーが表示されます。このオプションと [スタジオアシスト] を選択すると、メソッド内に #DIM 文を入力したときに、未宣言のローカル変数が表示されます。

スタジオアシストで内部クラスメンバを表示 ： チェックが付いている場合、スタジオ・アシストで内部として指定されたクラス・メンバがリスト表示されます。

変数を追跡する ： このオプションを選択した場合、下線 (緑の波線) によって、変数の疑わしい使用が示されます。例えば、このオプションは、値が指定されていない、作成されていない、または既に削除されている変数をハイライト表示します。

縮小ビューでクラスを開く：このオプションを選択した場合、既定では、クラスは、すべての折りたたみ可能なセクションが縮小されて開きます (Ctrl-- を押した場合と同様)。選択していない場合、クラスは、折りたたみ可能なセクションが展開されて開きます (Ctrl++ を押した場合と同様)。

コード・スニペット

スニペットの表示 ： スタジオで表示するコード・スニペットのタイプにチェックを付けます。名前およびテキスト・ファイルを指定することによって、独自のコード・スニペットのセットを定義できます。ドキュメントのコンテキスト (右クリック) メニューから **[コード・スニペットの作成]** を使用します。現在アクティブなユーザ定義セット内にスニペットが作成されます。または現在アクティブなセットがない場合は、最初のセット内に作成されます。

詳細

自動保存は、ソフトウェアまたはシステム障害の場合に、スタジオ・ドキュメントに加えられた変更が消失するのを防ぎます。既定では、自動保存は有効で 5 分ごとに保存を実行します。ドキュメントのテキスト表示であるファイル X:¥documents and Settings¥<username>¥Local Settings¥Temp¥CST*.tmp への最後の保存後に変更された、任意の開いているドキュメントを保存します。その後ドキュメントを保存する (または閉じる) と、この .tmp ファイルは削除されます。スタジオがクラッシュすると、次回スタジオが開かれたときに、一時ファイルが存在するというメッセージが表示されます。この一時ドキュメントを開くと、必要な部分をスタジオ・ドキュメントに貼り付けることができます。

[サーバ・ステータス・チェックの有効化 (推奨)] は、開いているドキュメントやプロジェクトがこのスタジオ・プロセス外のサーバ上で変更されたかどうかをスタジオがチェックする頻度を決定します。スタジオがアクティブ・アプリケーションの場合は、**[スタジオはアクティブアプリケーション (2-60秒)]** という 1 つ目の設定が使用されます。スタジオがバックグラウンドで実行されている場合は、**[スタジオはバックグラウンドアプリケーション (30-600秒)]** という 2 つ目の設定が使用されます。低速のシステムを使用している場合は、このオプションのチェックを外してもかまいません。ただし、スタジオはサーバのステータスをチェックしなくなり、サーバ上のドキュメントやプロジェクトが変更されたり、スタジオの接続が切断されても、それらを適切な時点で検出できなくなります。使用には注意が必要です。

サーバ上で変更されて、スタジオ上では編集されていないドキュメントを自動的に再読み込みする：このオプションが選択されており、エディタでドキュメントを開いているがまだ編集していない場合、他のユーザがそのドキュメントの新規バージョンをサーバに保存すると、その開いているドキュメント・ファイルは自動的に更新されます。この設定は、`Set ^%SYS("Studio","Reload")=1` (無効にする場合は 0) というグローバルを使用して、ネームスペース単位で有効にできます。

生成されたドキュメントをネームスペース・ツリーで表示する：このオプションを選択した場合、ワークスペースのネームスペース・ウィンドウに生成されたファイルが表示されます。選択していない場合、生成されたファイルは表示されません。

ObjectScript の既定として INT を使用する：このオプションを選択した場合、**[ファイル]→[新規作成]→[ObjectScript ルーチン]** を選択すると、INT ファイルが開きます。選択していない場合、MAC ファイルが開きます。

資格情報を View Web Page に渡す：このオプションを選択した場合、`View Web Page` を選択すると、スタジオによってアクセス権が確認されます。

デフォルト言語を使用 (ツールバーをリセットして再起動します)： チェックを付けると、スタジオによって言語固有のリソースがロードされます。システムの既定の言語 (すべてのメニューを英語で表示) をオーバーライドするには、このチェック・ボックスのチェックを外します。変更内容を受け入れると、ツールバーがリセットされ、スタジオが再起動します。

エクスポート・フラグ ： ファイルのエクスポート時に使用するフラグを入力します。詳細は、"ObjectScript リファレンス" の $SYSTEM エントリの "フラグおよび修飾子" セクションを参照してください。

17.2 エディタ・オプション

エディタ・オプションにより、スタジオのテキストの動作を指定できます。オプションには以下のものがあります。

構文チェックおよびアシスト

シンタックスチェック有効：チェックが付いている場合、構文エラーはハイライト表示されます。構文チェックを実行するタイミングを指定できます。つまり、変更が加えられるたびに（文字を入力または削除するたびに）（**[変更時にシンタックスチェック]**）構文チェックを実行するのか、カーソルが現在の行から移動するたびに（**[行を離れるときにシンタックスチェック]**）構文チェックを実行するのかを指定できます。さらに、構文エラーに下線（赤の波線）（**[エラーに下線を付ける]**）を付けるかどうかも指定できます。

括弧の対応付けチェック有効：チェックが付いている場合、現在のカーソル位置を囲む括弧の組み合わせは、太字で表示されます。括弧は [ ] 角括弧、( ) 小括弧、< > 山括弧など、使用する言語によって異なります。**[括弧の対応付けチェック有効]** を有効にするには、**[シンタックス・チェック有効]** にチェックを付ける必要があります。**[括弧対応付け行数制限]** を選択すると、対応する括弧を検索する範囲がキャレット位置の上下の特定数の行に制限されます（大きいファイル内で無制限に検索するとエディタの速度が大幅に低下するため）。

スタジオアシスト：コード補完を有効にします。ObjectScript コードの入力時、ドロップダウン・メニューに、次に入力可能なオプションが表示されます。パッケージ名を入力する場合は、利用可能なクラスがリスト表示されます。クラスを入力する場合は、利用可能なメソッドがリスト表示されます。メソッドを入力する場合は、利用可能な引数がリスト表示されます。利用可能なオプションは、他の位置にも表示されます（#dim 宣言やトリガ・コードなど）。

#DIM 文を入力したときに未宣言のローカル変数を表示するには、[スタジオアシスト] と [Option Explicit] を選択済みであり、カーソルがメソッド内にある必要があります。この際にリスト表示されるための変数の条件は、% で始まっていないこと、パラメータでないこと、パブリック・リスト内にないこと、およびまだ宣言されていないことです。

$$$ と入力すると、使用可能なマクロが次のように一覧表示されます。ユーザ定義マクロがリスト表示されるのは、それらのマクロが現在のファイルで定義されており、インクルード・ファイルで定義されており、そのインクルード・ファイル内で、それらのマクロの前に 3 つのスラッシュ（///）で始まる行が記述されている場合です。現在のファイルがクラス・ファイルの場合、システム定義のマクロがリスト表示されます。

メンバ名の一部を入力すると、それに続いて部分的に一致するメンバ名のリストが自動的に表示されますが、この部分的な入力内容が二重引用符（または一重引用符）で始まっている場合は、ポップアップには、プログラム内で引用符で囲まれる必要のある名前（空白などの非英数字が含まれた名前）を持つメンバのみが表示されます。部分的な入力内容が二重引用符で始まっていない場合は、ポップアップには、引用符で囲む必要のない名前を持つメンバのみが表示されます。ピリオドの直後にスタジオ・アシストが呼び出された場合は、ポップアップにはすべてのメンバ名が表示されます。

解析の遅延：**解析が遅い場合はチェックしてください。行が点滅する場合はチェックをはずしてください。[変更時に構文チェック]** および **[解析の遅延]** の両方が有効な場合、テキストの入力がパーサのリパースより速いと、テキストは解析の色と黒色で交互に点滅します。テキストが点滅する場合、このオプションのチェックを外すか Ctrl-Alt-O を押して、**[解析の遅延]** を無効にします。すべてのキーストロークによってリパースが生じるので、応答が多少遅くなることがありますが、点滅は停止します。このスイッチは、各スタジオ・セッションの開始時に設定する必要があります。

スタジオ・アシストは、RESTSpec XData ブロック用のコード実行ポップアップや補助的なポップアップを提供します。

色

スタジオの構文チェックでは、サポートされている言語ごとに、異なるカラー表示方法を使用できます。スタジオで構文のカラー表示が有効の場合、このオプションにより、構文要素をハイライト表示するための色を指定できます。

特定の構文要素に対して、スタジオ・エディタで使用する色を変更する場合、**[オプション]** ダイアログの **[見栄え]** タブで以下の操作を行います。

1. 利用できるオプションから、言語を選択します (ObjectScript など)。
2. 構文要素を選択します (コメントなど)。
3. 希望のフォアグラウンドの色 (およびバックグラウンドの色) を選択します。
4. **[適用]** ボタンを選択して、新しい色を使用します。

[リセット] を使用すると、選択した構文要素を既定の色に戻すことができます。

[全てリセット] を使用すると、すべての色を既定値に戻すことができます。

注釈　エディタ・ウィンドウで特定の構文要素を右クリックし、**[シンタックスの色を設定する]** コマンドを選択することでも、構文の色を変更できます。

キーワード拡張ケース

この機能は、ObjectScript ルーチンにのみ適用されます。

[編集]→[詳細]→[拡張コマンド] を選択して、ObjectScript コマンドを拡張するときに使用する **[現在の大文字と小文字の使用]**、**[大文字]**、**[小文字]**、または**[大文字と小文字の混在]** を指定します。このオプションを設定し、拡張するコードをハイライト表示して、**[編集]→[詳細]→[拡張コマンド]** を選択します。このオプションは、コマンドの圧縮時にも使用します。

インデント

自動インデントの特性を定義します。

- **基本** : このオプションを選択した場合は、現在の行の先頭にタブ、スペース、またはスペースとタブの組み合わせが入力されている場合、Enter キーを押すと、自動的に次の行の先頭に同じスペースとタブの組み合わせが入力されます。
- **ユーザ定義 (タブは '/t')**

  有効の場合、後続の各行の先頭に自動的に入力する文字を指定できます。例えば、`\t.#/;` (タブ、ドット、ポンド記号、スラッシュ、セミコロン) の文字グループを入力すると、行がこれらの文字のいずれかまたはこれらの文字の任意の組み合わせで始まる場合に、Enter キーを押すと、自動的に次の行が同じ文字の組み合わせで始まります。
- **なし** : 自動インデントは行われません。

コメント

スタジオ・ドキュメント・タイプのコメント・デリミタのテーブルを表示します。セルを選択して、デリミタを入力します。スタジオ・ドキュメント内のテキストのブロックをハイライト表示してから、Ctrl-Alt-/ キーを押して Multi-Start および Multi-End 文字でブロックを区切ります。

表示

一部の項目の表示を制御します。

- **特殊文字を表示** : チェックが付いている場合、エディタは特殊シンボルを使用して、改行記号とタブ文字を表示します。
- **行番号の表示** : 有効な場合、エディタに行番号が表示されます。
- **タブをスペースに変換** : 有効な場合、エディタでタブはスペースに変換されます。
- **生成されたファイルの濁った色の背景色** : 有効な場合、ユーザ作成ファイルと区別するために、生成されたファイルが濁った色の背景色で表示されます。
- **タブ・サイズ** : スペースの数でタブのサイズを指定します。

17.3 コンパイル・オプション

コンパイル・オプションは、スタジオがコードをコンパイルする方法に影響します。[**フラグおよび最適化**]と[**動作**]という、2 つのページがあります。

フラグおよび最適化

このページには、[**コンパイルフラグ**]、[**最適化レベル**]、および [**フラグ**] の 3 つのセクションがあります。

[**コンパイルフラグ**] セクションには、以下の項目があります。

生成されたソースコードを保存 : チェックが付いている場合、コンパイラがコンパイルの結果として生成するすべての中間ソース・コード (ルーチン) を削除しないことを指定します。

依存したクラスをコンパイル : チェックが付いている場合、コンパイラはクラスの依存サブクラスもすべてコンパイルします。

関連する最新ドキュメントをスキップ : チェックが付いている場合、[最新ドキュメントをコンパイルしない] フラグが設定され、コンパイラは最後にコンパイルされてから変更されていない関連ドキュメントをコンパイルしません。ただし、エディタ内の現在のドキュメントは常に再コンパイルされます。

使用中のクラスをコンパイル : チェックが付いている場合、クラスのインスタンスが現在使用中であっても、コンパイラはそのクラスをコンパイルします。

[**最適化レベル**] セクションでは、実行速度を向上させるために、最適化のレベルを設定できます。最適化が有効になっている場合、コンパイラは、クラス間で式をコピーしてメソッド呼び出しを無くすなどのコードの再編成を行って最適化を図ります。レベルには以下のものがあります。

- **最適化なし** : 開発時に推奨されます。依存クラスをリコンパイルせず、ソースおよびオブジェクト・コード間の強い相関関係を維持し、読み取りとデバッグを容易にします。
- **プロパティの最適化** : ..property からインスタンス変数リファレンスへのリファレンスを最適化します (get/set メソッドがオーバーライドされない、データ型により記述された簡易プロパティの場合)。
- **クラスおよびライブラリ・クラス呼び出しの最適化** : クラスおよびシステム (%) クラスへの呼び出しを最適化します (プロセス時にコードが抽出および移動されるとき)。増分コンパイルは、最適化されたクラスには使用できなくなりました。

[**フラグ**] フィールドには、使用するコンパイラ・フラグを入力します。

このターミナルのフラグ・リストを確認するには、`d ##class(%SYSTEM.OBJ).ShowFlags()` を入力します。

修飾子のリストを確認するには、`d ##class(%SYSTEM.OBJ).ShowQualifiers()` を入力します。

フラグ	結果
a	アプリケーション・クラスを含めます。既定では、このフラグが設定されます。
b	サブ・クラスを含めます。
c	コンパイル。ロード後、クラス定義をコンパイルします。
d	表示。既定では、このフラグが設定されます。
e	エクステントを削除します。
h	ヘルプを生成します。
I	ロード時、XML エクスポート形式をスキーマに対して検証します。
k	ソースを保持します。このフラグを設定すると、生成されたルーチンのソース・コードが保持されます。
l	コンパイル時にクラスをロックします。既定では、このフラグが設定されます。
p	パーセント。%* 形式の名前でクラスを含めます。
r	処理を再帰的に実行します。先行依存するすべてのクラスをコンパイルします。
s	プロセス・システム・メッセージまたはアプリケーション・メッセージ
u	アップデートのみ実行します。最新状態であればクラスのコンパイルをスキップします。
v	SQL 使用下でのクラスの参照方式、または現在のクラスからの参照方式のいずれかにおいて、現在のクラスと関連するクラスを含めます。

注釈　　フラグの前にダッシュ (-) を記述すると、そのフラグをオフにすることができます。

動作

- **[コンパイル前]**：**[コンパイル]** を選択したときのスタジオの既定の動作を選択できます。コンパイル前に、スタジオの動作として、**[変更したドキュメントをすべて自動的に保存する。]**、**[変更されたドキュメントの保存にプロンプトを出す。]**、または **[変更されたドキュメントを保存しない。]** を選択できます。
- **保存時にルーチンをコンパイル**：このオプションを選択すると、**[保存]** を選択したときに、変更したルーチンがコンパイルされます。既定では、このオプションは選択されていません。

17.4 SQL オプション

このオプションは、主にスタジオを使用して、既存のレガシー・データにマップするクラスを生成する場合に使用します。

レガシー・モード：**クラスのレガシー SQL モード有効**

このオプションが有効の場合、このタブにある他の既定の設定が有効になります。このオプションは、スタジオ・ウィザードの操作にのみ影響し、アプリケーションの実行時の振る舞いには何の影響も与えません。

デフォルトストレージタイプ：新規クラス・ウィザードで新規のクラスを生成するときに使用されるストレージ・クラスを指定します。

$Pieceのデフォルト区切り文字:：レガシー・データ構造のマップを定義するときに使用される既定のデータ区切り文字を指定します。

デフォルトの照合 : レガシー・データ構造へのマッピングを定義するときに使用される既定のインデックス照合を指定します。

RowIDはプライベート : 新規クラスの SqlRowIdPrivate フラグを既定で設定するかどうかを指定します。

Row IDを自動生成 : データを既存のストレージ構造にマップするときに、行 ID フィールドを自動的に生成します。

17.5 スタジオの外観のオプション

このオプションを使用すると、リストからテーマを選択してスタジオのカラー・テーマを変更できます。

A

スタジオに関するよくある質問

スタジオに関する質問と回答のセットです。

プロジェクト

プロジェクトとは何ですか？

プロジェクトとは、便宜上まとめることのできる、クラス定義またはルーチンの集合です。

プロジェクトを使用すると、スタジオ・セッションを開始するときに、簡単に今までの作業に戻ることができます。例えば、アプリケーション、またはその一部に関連するすべてのクラスを、1 つのプロジェクトの中に置くことができます。スタジオを開始するときにこのプロジェクトを開くと、ワークスペース・ウィンドウの [プロジェクト] タブに、すべてのクラスが一覧表示されます。

1 つの外部ファイルからプロジェクト全体をエクスポートしたりインポートしたりできるので、簡単にアプリケーション・コードを保存したり、渡したりできます。

プロジェクトに項目を追加するには、どのようにすればいいでしょうか？

以下に、現在のプロジェクトに項目を追加する方法のいくつかを示します。

- **[ファイル]→[開く]** で 1 つまたは複数の項目を開く前に、**[開く]** ダイアログの **[プロジェクトに追加]** チェック・ボックスにチェックを付けます。
- 現在のエディタ・ウィンドウの項目を現在のプロジェクトに追加するには、**[プロジェクト]→[アイテム追加]** を選択します。
- ワークスペース・ウィンドウで、項目をハイライト表示して右クリックし、**[プロジェクトに追加]** を選択します。

異なるネームスペースのものを、プロジェクトに追加することはできますか？

いいえ。プロジェクトに含むことができるのは、現在の InterSystems IRIS® ネームスペースから見える項目のみです。

1 つの項目が、複数のプロジェクトに属することはできますか？

はい。プロジェクトは、一緒にグループ化するために選択して指定した項目 (クラス定義およびルーチン) のセットです。項目自体は、属するプロジェクトに対してリンクを持ちません。項目が属することのできるプロジェクトの数の制限はありません。

プロジェクトを使用したくない場合は、どのようにすればいいでしょうか？

スタジオでは、必ずしもプロジェクトを使用する必要はありません。完全に無視しても問題ありません。プロジェクトを無視するには、既定のプロジェクトに項目を追加せず、スタジオの終了時に、プロジェクトを保存するかどうかを聞かれたときに、これを無視してください。

プロジェクトをエクスポートできますか？

はい。**[ツール]→[エクスポート]→[プロジェクトのエクスポート]** を選択します。ファイル名を入力し、**[OK]** をクリックします。これにより、現在のプロジェクトのコンテンツがすべて (プロジェクト定義も含め) 1 つの XML ファイルにエクスポートされます。

プロジェクトを削除するには、どのようにすればいいでしょうか？

[ファイル]→[開く] を選択して、すべてのプロジェクトを表示します。プロジェクトを右クリックして、**[削除]** を選択します。

この方法では、**[ファイル]→[開く]** を使用して、サーバ上のすべてのタイプの項目を削除することができます。

ファイルを開く

クラス定義を開くには、どのようにすればいいでしょうか？

既存のクラス定義 (InterSystems IRIS サーバに保存されているもの) を開くには、以下を実行します。

1. クラス定義を含む InterSystems IRIS ネームスペースおよびサーバに接続していることを確認します。
2. **[ファイル]→[開く]** を選択します。
3. **[開く]** ダイアログの **[ファイルの種類]** コンボ・ボックスで、[`Class Definitions`] (.CLS) または [`All`] を選択して、クラス定義が表示されているかどうかを確認してください。
4. パッケージ名は、フォルダとしてファイルのリストに表示されます。パッケージ名を選択すると、パッケージ内のすべてのクラス (または、サブパッケージ) が表示されます。
5. クラス名をダブルクリックして開きます。

あるいは、開きたいクラスの名前に **.cls** 拡張子を付けて (**Sample.Person.cls** など) ファイル名編集ボックスに直接入力し、**[開く]** を選択します。

ルーチンを開くには、どのようにすればいいでしょうか？

既存のルーチン (InterSystems IRIS サーバに保存されているもの) を開くには、以下を実行します。

1. ルーチンを含む InterSystems IRIS ネームスペースおよびサーバに接続していることを確認します。
2. **[ファイル]→[開く]** を選択します。
3. **[開く]** ダイアログの [ファイルの種類] コンボ・ボックスで、[`MAC routines`] (.MAC)、[`INT routines`] (.INT)、または [`All`] を選択して、ルーチンが表示されているかどうかを確認してください。
4. ルーチン名をダブルクリックします。

あるいは、拡張子を付けたルーチン名 (**MyRoutine.MAC** など) をファイル名編集ボックスに直接入力し、**[開く]** を選択します。

CSP ファイルを開くには、どのようにすればいいでしょうか？

クラス定義やルーチンと同じ方法で CSP ファイルを開くことができます。主な違いは、**[開く]** ダイアログでは CSP アプリケーションがフォルダとして表示され（/csp/samples など）、アプリケーションの名前を選択すると、その中の CSP ページが表示されることです。InterSystems IRIS での CSP ファイルの使用はお勧めしません。

[開く] ダイアログの [システム・アイテムを含む] というチェック・ボックスは、何ですか？

[システム・アイテムを含む] チェック・ボックスにチェックが付いている場合は、**[ファイル]→[開く]** ダイアログに現在のネームスペースに含まれる項目に加え、システム項目（名前の最初に % 文字が付き、IRISLIB データベースに保存されている項目）も表示されます。

[ファイル]→[開く] ダイアログで、パターン・マッチングを使用できますか？

はい。標準の **[ファイル]→[開く]** ダイアログの場合と同様に、* 文字を任意の文字列と一致するワイルドカードとして使用できます。ファイル拡張子を使用すると、特定の項目をフィルタにかけることができます。例えば、*.cls は、選択されたパッケージ内のクラス定義のみを表示します。任意の文字の照合には、? 文字を使用できます。これらは Windows のパターン・マッチング規則であり、InterSystems IRIS のパターン・マッチングではありません。

異なるネームスペースからルーチンを開くには、どのようにすればいいでしょうか？

[ファイル]→[開く] ダイアログでは、現在のネームスペースとサーバにある項目のみが表示されます。異なるネームスペースやサーバからルーチンを開くには、以下の方法があります。

1. **[ファイル]→[ネームスペース変更]** を選択します。
2. 必要なルーチンを開きます。

% クラスを開くことはできますか？

はい。下部にある **[システム・アイテムを含む]** チェック・ボックスにチェックを付けることにより、**[ファイル]→[開く]** ダイアログで **%** クラス（パッケージ名が % 文字で始まり、IRISLIB データベース内に保存されているクラス）を一覧表示することができます。

%SYS 以外のネームスペースに接続したまま **%** クラスを開く場合、スタジオはそれらのクラスを読み取り専用で開きます。

[ファイル]→[接続] では、何が実行されますか？

スタジオは、特定の InterSystems IRIS ネームスペースとサーバへの接続を保持します。この接続を使用して、クラスのリストを表示します（プロパティ・タイプやスーパークラスの指定など）。また、デバッグにもこの接続を使用します。**[ファイル]→[接続]** によって別のサーバに接続できます。

デバッグ

デバッグを開始するには、どのようにすればいいでしょうか？

以下の方法で、デバッガをターゲット・プロセスに接続することができます。

- **[プロジェクト]→[設定]** を使用して、現在のプロジェクトのデバッグ対象 (デバッグを行うプログラム、またはルーチンの名前) を定義します。次に、**[デバッグ]→[実行]** を選択して対象のプログラムを起動し、そのサーバ・プロセスに接続します。
- **[デバッグ]→[アタッチ]** を選択して、InterSystems IRIS サーバで実行中のプロセスのリストから選択します。

詳細は、"スタジオ・デバッガの使用" を参照してください。

クラスをデバッグするには、どのようにすればいいでしょうか？

スタジオのデバッガを使用すると、InterSystems IRIS サーバで動作するプログラムをステップ実行できます。このようなプログラムとしては、INT ファイル、MAC ファイル、CLS ファイル内のメソッド、Java から呼び出されるサーバ側のメソッド、サーバでホストされるアプリケーションなどがあります。

1. デバッグ中に INT ファイルを表示し、後で見直すために INT を保存するには、クラスをコンパイルする前に、**[生成されたソース・コードを保存]** オプションを設定します。このオプションは、**[ツール]→[オプション]→[コンパイラ]→[フラグおよび最適化]** ページにあります。
2. 希望するソース行で **F9** (ブレークポイントの切り替え) を押すことにより、クラス・メソッド (または前述の任意の他のファイル) 内の希望する場所にブレークポイントを設定します。
3. **[デバッグ]→[デバッグ対象]** を使用して、デバッガがコードの実行を開始する位置を指定する、デバッグ対象を設定します。ステップ実行を行うクラス名とメソッド名を入力します。
4. **Ctrl-F5** キーを押すか、**[デバッグ]→[実行]** を選択してデバッガを起動します。

変数をウォッチできますか？

はい。デバッグ中に、スタジオのウォッチ・ウィンドウの左側の列に、変数名 (または式) を入力します。デバッガが一時停止するたびに、変数または式は再評価されます。

編集

エディタの色分けは何を意味するのですか？エディタ内の色を変更することはできますか？

スタジオでは、指定した言語の構文の要素を識別するために、色が使用されます。

以下の方法で、さまざまな構文要素の色を変更します。

1. **[ツール]→[オプション]→[エディタ]→[色]** を選択します。
2. 言語を選択します。
3. 要素 (comment、variable など) を選択します (利用できる要素は、選択した言語により異なります)。
4. 前景色と背景色を選択して、**[OK]** を選択します。

コードの下に赤い波線があるのですが、これは何ですか？

赤い波線は、そのコード (またはその前のコード) に構文エラーが含まれることを示しています。

スタジオで漢字および中国語文字を使用できますか？

はい。スタジオは、Unicode 文字および漢字を完全にサポートしています。

スタジオでヘブライ文字およびアラビア文字を使用できますか？

はい。スタジオ・エディタは、双方向編集と同様に、ヘブライ文字およびアラビア文字もサポートしています。

ファイルのインポート

外部ファイルから、クラス定義やルーチンをインポートできますか？

はい。**[ツール]→[インポート]** を選択します。

ローカル・ファイルとリモート・ファイルの違いは何ですか？

スタジオは、クライアント・サーバ・アプリケーションです。つまり、スタジオ自体はクライアントのシステム上で稼動し、サーバと会話します。サーバは、同じマシンまたはリモート・マシンのどちらにでも置くことができます。スタジオでローカルとリモートという用語を使用する場合、クライアント・システムおよびサーバ・システムに格納されているオペレーティング・システム・ファイル（インポートやエクスポートの際など）を意味します。

クライアントとサーバの両方が同じシステム上にある場合は、ローカルとリモートは同じです。

印刷

スタジオから印刷することはできますか？

はい。**[ファイル]→[印刷]** または **[ファイル]→[印刷プレビュー]** を選択します。

テンプレート

テンプレートとは何ですか？

テンプレートは、ユーザ定義のスタジオのアドインを生成するためのメカニズムです。テンプレートは、現在のドキュメントの現在のカーソル位置にコードの断片を挿入するプログラムです。必要に応じて、そのコードの断片をカスタマイズすることができます。詳細は、"スタジオ・テンプレートの使用法" を参照してください。

利用できるテンプレートのリストはありますか？

はい。**[ツール]→[テンプレート]→[テンプレート]** を選択します。

新規のテンプレートを作成できますか？

はい。"スタジオ・テンプレートの使用法" を参照してください。

マルチユーザ・サポート

スタジオは、複数のユーザによる開発をサポートしていますか？

はい。これには、以下のようにいくつかの方法があります。

- 共通の InterSystems IRIS サーバ・システムをセットアップし、すべての開発者がこれにコードを保存する方法。
- (開発者のシステム上の) ローカルの InterSystems IRIS サーバを使用して、エクスポートした XML ファイルとして、ソース・コードをソース管理システムに保存します。詳細は、"InterSystems IRIS とソース・コントロール・システムの統合" を参照してください。

他の人が編集中のクラス (またはルーチン) を開こうとすると、どうなりますか？

スタジオは、クラス (またはルーチン) は使用中であるというダイアログを表示し、読み取り専用モードで開くかどうかを尋ねます。

現在使用しているクラスのスーパークラスを、他の人が変更しようとするとどうなりますか？

スタジオは、ある開発者が使用中のクラスのスーパークラスに、他の開発者が変更を加えるのを阻止しません。

クラスを開いて編集するときに、常にすべてのサブクラスのロックを取り外すことはできますが、これは実際には非常に面倒で扱いにくいものです。その代わりに、開発チームで、スーパークラスの定義や変更についての規則や手順を調整する必要があります。これは、他の言語 (Java など) を使用している開発チームが、ソース管理システムでクラス定義を使用する場合の状況に似ています。

クラス

新規のクラスを生成するには、どのようにすればいいでしょうか？

[ファイル]→[新規作成] を使用して、[クラス定義] を選択します。[新規クラス] ウィザードが起動します。

詳細は、"クラス定義" を参照してください。

自分のクラスに生成されたソース・コードを見ることができますか？

はい。**[ビュー]→[他のコードの表示]** を使用して、クラス・コンパイラが生成したすべてのソース・コードを見ることができます (現在のウィンドウにクラス定義が含まれる場合にのみ利用できます)。

クラスをコンパイルする前に、**[生成されたソース・コードを保存]** オプションが設定されていることを確認してください。このオプションは、**[ツール]→[オプション]→[コンパイラ]→[フラグおよび最適化]** ページにあります。

クラスをコンパイルするときに、スタジオが "このクラスは最新なのでコンパイルする必要がない" というメッセージを表示します。強制的にコンパイルを実行することはできますか？

はい。**[関連する最新ドキュメントをスキップ]** オプションをオフにします。このオプションは、**[ツール]→[オプション]→[コンパイラ]→[フラグおよび最適化]** ページにあります。

ルーチン

INT ルーチンを生成するには、どのようにすればいいでしょうか？

[ファイル]→[新規作成] を選択して、新規の ObjectScript ルーチンを作成し、そのルーチンに **.INT** 拡張子を付けて保存します。インクルード・ファイル (**.INC**) も、同様の方法で生成できます。

SQL

SQL ビューを定義するには、どのようにすればいいでしょうか？

スタジオには、SQL ビューを定義するメカニズムがありません。このタスク、および SQL に関する他のタスクを実行するには、管理ポータルを使用してください。

ソース・コントロール

スタジオは、外部のソース管理システムと統合できますか？

はい。 詳細は、"InterSystems IRIS とソース・コントロール・システムの統合" を参照してください。

互換性

スタジオ・クライアントを InterSystems IRIS サーバに接続できますか？

スタジオ・クライアントでは、InterSystems IRIS と同じバージョンを実行しているか、その接続先の InterSystems IRIS サーバより新しいバージョンを実行している必要があります。

Linux でスタジオを使用できますか？

スタジオ・クライアントは、Windows プラットフォームでのみ稼動します。Windows クライアントを使用して、任意のサーバと対話できます。また、VMWARE などのパーティション・マネージャを使用して、Windows パーティションと Linux パーティションの両方を、ユーザの開発システム上で稼動させ、Linux パーティションで InterSystems IRIS を実行しながら Windows パーティションでスタジオを実行することもできます。唯一のコツは、Windows パーティションが TCP/IP 経由で Linux パーティションと通信できるようにネットワークを構成することです。また、スタジオは、Intel ベースの Macintosh 上の Windows でも実行できます。

スタジオ実装

スタジオは、なぜ Microsoft Visual Studio のライセンス・コンポーネントを使用していないのですか？

スタジオを、ライセンス供与や Visual Studio の拡張を利用しないで、新規に構築したのには、いくつかの理由があります。

- スタジオ・エディタは、Microsoft Studio フレームワークでは利用できない高性能の構文解析技術を使用しています。
- Microsoft 社は、Visual Studio の今後のバージョンでの互換性を保証しません。

スタジオのインタフェースは、なぜ Java で開発されなかったのですか？

現時点では、スタジオ・エディタで快適なパフォーマンスを得るためには、Windows API への直接呼び出しが唯一の方法です。Java を使用して開発された構文カラー表示エディタもありますが、これらのエディタにはスタジオが使用する高性能の多言語構文解析が提供されないので、相応のパフォーマンスを得るには非常に性能の高いコンピュータを必要とします。