

開発者向けオンラインセミナー
**Visual Studio Codeを使用した
IRISプログラミング**

皆本 稔彦

インターシステムズジャパン株式会社

2024年2月29日



説明内容

- 1 **IRISプログラミングに必要な拡張機能**
- 2 **開発環境の作成手順**
- 3 **Interoperabilityのクラス作成方法**
- 4 **コードスニペットを使用する**



- Visual Studio Codeの世界へようこそ！



Visual Studio Code (VSCode) とは


- Microsoftが無償で提供している開発用エディタ
- ダウンロードサイト：<https://code.visualstudio.com/>
- Windows/MAC/Linux等で実行可能
- 日本語にも対応
- Gitとの連携が可能
- 拡張機能の仕様が公開されており、ベンダーやコミュニティ、個人が様々な拡張機能を提供
マーケットプレイスから自由にインストールが可能
- インストール方法は「VScode インストール 日本語」で検索



- IRISプログラミングに必要な拡張機能

InterSystems ObjectScript Extension Pack



- VSCodeにて拡張機能  をクリック、Extension Packをインストールします。



- 以下の拡張機能がインストールされます。



InterSystems ObjectScript

クラスやルーチンのコンパイル、デバッグコードの実行など



InterSystems Server Manager

サーバ設定の管理



InterSystems Language Server

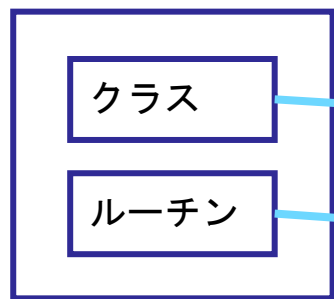
エディタに書かれたwordのハイライト化、使用されているメソッドやクラスへの移動

Extension Packを使ったプログラム編集方法

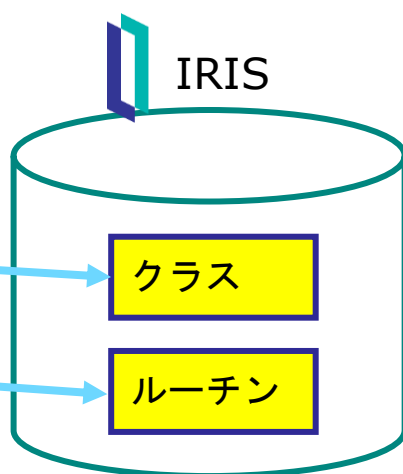


IRISにあるクラスやルーチンを直接編集する

- スタジオと同じ方法

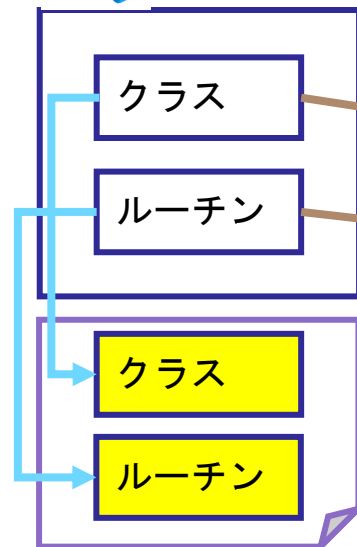


作成
保存



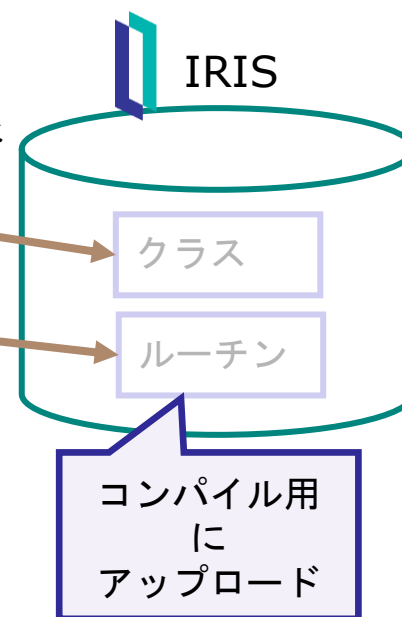
VSCodeのフォルダにあるファイルを編集する

- IRISにアップロードしてコンパイル
- VSCodeと同じ方法



windows フォルダ

アップロード
コンパイル



編集方法のメリット、デメリット



IRISにあるクラスやルーチンを直接編集する

- メリット
 - 複数の開発者で環境を共有するため、最新のIRISプログラムが反映されている
- デメリット
 - 複数の開発者で同じプログラムを編集するとデグレにつながり、戻せなくなる
 - GitHubとの連携が難しい

Windowsのフォルダにあるファイルを編集する

- メリット
 - GitHubとの連携は簡単
 - 他の言語のソースファイルも一緒に管理できる
- デメリット
 - IRISの開発環境が個別に用意が必要

おすすめ！



- 開発環境の作成



開発環境の構成

IRISをコンテナで実行するのがおすすめ

- メリット
 - IRISインストール作業が簡単
 - スタンドアロン環境で開発、コンパイル、テスト等が実施できる。
 - 毎回新しい環境で作業できる。余計なデータ、プログラムが残らない。
- デメリット
 - キューブが使えない ⇒ VSCodeの拡張機能から起動
 - ファイルパスはLinux形式

WSL (Windows Subsystem for Linux)

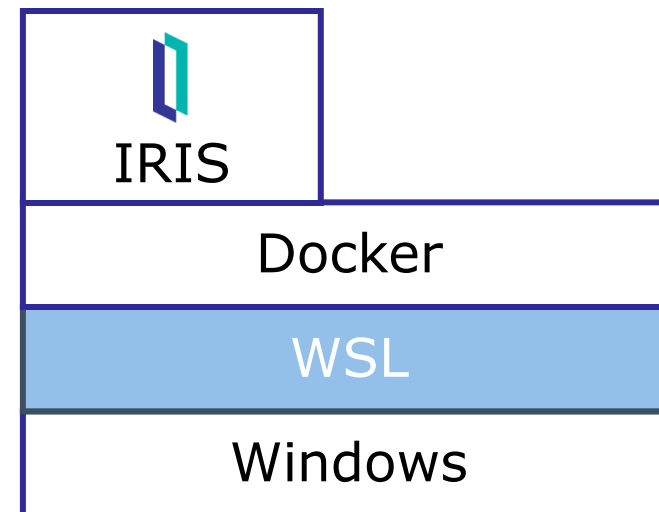


- LinuxのプログラムをWindows Server や Windows 10/11で動作させるためのソフトウェア
- WSL上でDockerコンテナが利用可能
- Homeエディションでも動作可能

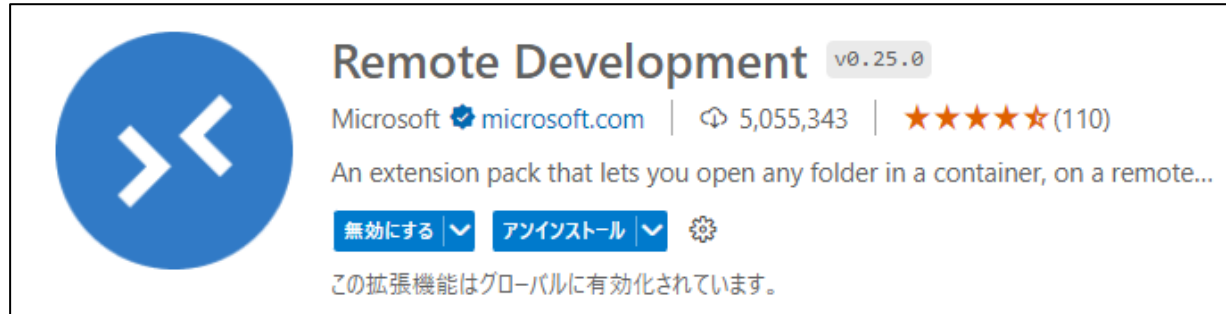
- コントロールパネルの「Windowsの機能の有効化または無効」にて
 - Linux用Windowsサブシステム
 - 仮想マシンプラットフォームを有効化する

- WSL、Dockerのインストールは以下を参照

https://developer.mamezou-tech.com/blogs/2023/09/09/docker_ubuntu_on_wsl2/



vscodeよりWSL、dockerコンテナを操作するための拡張機能

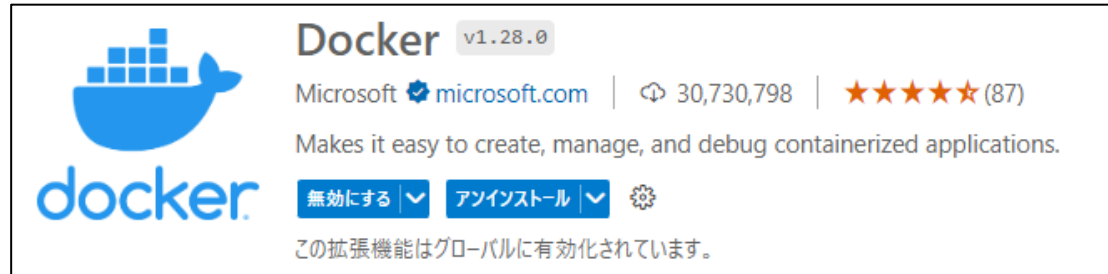


- 以下の拡張機能がインストールされます
 - Dev Containers
Docker desktop(有償)をインストールすることで、プロジェクトからコンテナの操作が可能
 - WSL
WSLのLinuxに接続し、vscodeからLinuxの操作が可能
 - Remote - SSH
 - Remote - Tunnels
- WSLに接続します



IRISコンテナ環境を作成、起動する拡張機能

- Docker



- GitHubからコンテナ作成に必要なファイルをダウンロード
<https://github.com/intersystems-jp/IRISContainer>
- docker-compose-community.yml を右クリック ⇒ 「Compose up」
- コンテナのビルドと起動



- Interoperabilityに必要なクラスを作成する

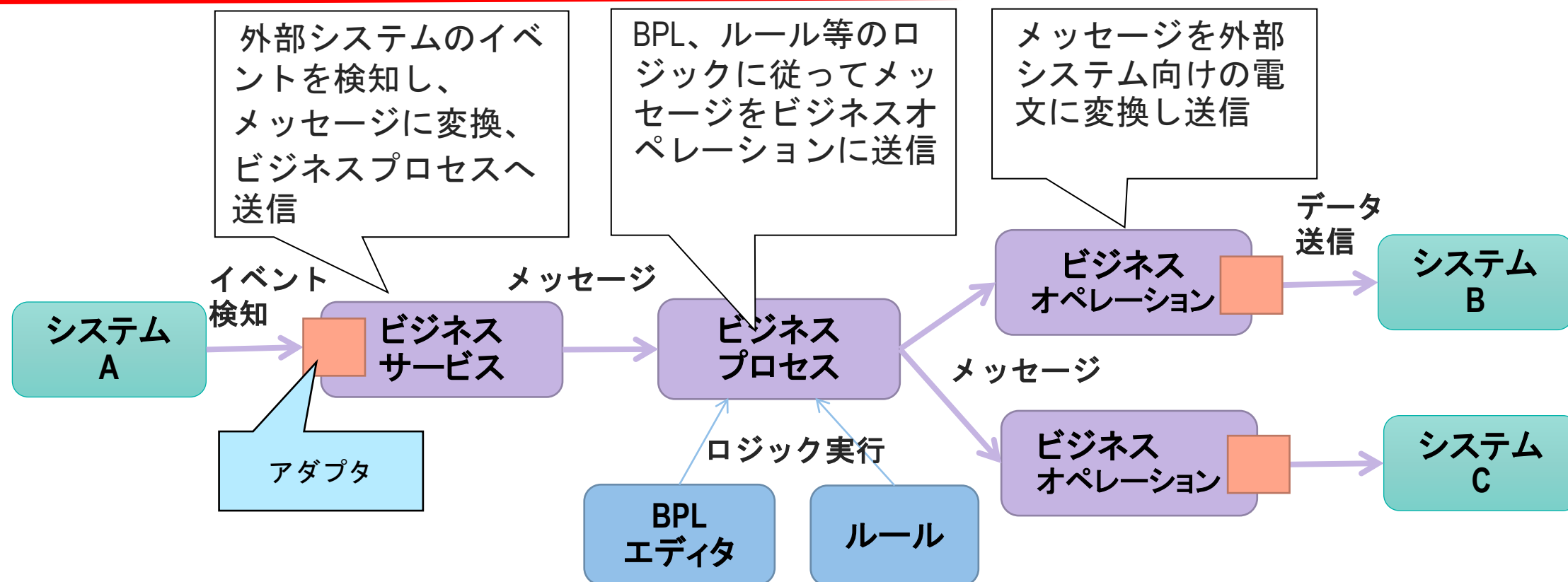
Interoperability とは



- 外部のシステムとの連携に必要な機能をまとめたライブラリ

- 開発者コミュニティを参照

<https://jp.community.intersystems.com/node/483021>



インターオペラビリティ関連クラスの作成



- 「ファイル」->「新しいファイル…」

- Business Service
ビジネスサービス
- Business Operation
ビジネスオペレーション
- Business Process
ビジネスプロセス
- Business Rule
ビジネスルール
- Data Transformation
データ変換





- コードスニペットの利用

スニペットの作成方法

- 「設定」->「ユーザースニペット」
- 「新しいグローバルスニペットファイル…」
- スニペットファイル名を入力
- スニペットファイルのフォルダに入力した「ファイル名.code-snippets」というファイルが作成される
- Json形式でスニペットを記述
- scopeに指定する言語
(複数の場合、カンマ区切り)
 - objectscript-class
クラス定義にて利用
 - objectscript
ルーチンにて利用

```
{
  "タイトル": {
    "scope": "objectscript-class",
    "prefix": "名称(英数字)",
    "body": [
      " 1 行目のテキスト",
      " 2 行目のテキスト",
      :
      " n 行目のテキスト"
    ],
    "description": "スニペットの内容"
  }
}
```



スニペットで可以使用の変数

- **タブ入力**

\$1,\$2, …\$n
\${n:デフォルト値}

- **クリップボード**

\${CLIPBOARD}

- **ファイル情報**

\${TM_DIRECTORY}	挿入するファイルのディレクトリ
\${TM_FILENAME}	挿入するファイルのファイル名
\${TM_FILENAME_BASE}	挿入するファイルのファイル名（拡張子を除く）
\${RELATIVE_FILEPATH}	プロジェクトフォルダから、指定されたファイルへの相対パス

- **年月日**

\${CURRENT_YEAR}	年
\${CURRENT_MONTH}	月
\${CURRENT_DATE}	日

参照 : <https://code.visualstudio.com/docs/editor/userdefinedsnippets>



変数の変換

- 正規表現を使った変換
- `${変数名 / 正規表現 / 変換先 /g}`
- バックスラッシュ(`\`)は2文字重ねて(`\\`)使用

- ファイル名の拡張子を取り除く変換例

`${TM_FILENAME/(.*)\\.[^.]+$/$1/}`

`(.*)` ... 0文字以上のすべての文字。変換先の`$1`はこの部分の文字列

`\\.` ... `.` (ドット文字)

`.+` ... 1文字以上のすべての文字

`$` ... 文字列の最終



まとめ

- VSCodeを使ったGitとの連携
- WSL/Dockerを使用した個人用開発環境の構築
- Interoperability に必要なクラスの作成方法
- ユーザースニペットの使用例