

JavaScript によるモデルの操作

目次

■ はじめに.....	1
■ モデル操作の手順.....	2
1. JavaScript の作成.....	2
2. 外部ファイルによるフィーチャの指定.....	3
3. JavaScript の実行.....	3
■ 外部ファイルに記述した選択指定の実行例.....	4
1. プロジェクトのインポート.....	4
2. JavaScript でフィーチャの選択を行う.....	6
3. pure::variants から JavaScript を実行する.....	7
コマンドラインからの JavaScript バッチ実行.....	9
■ 制約等を含むフィーチャモデルの外部定義とインポートについて.....	10

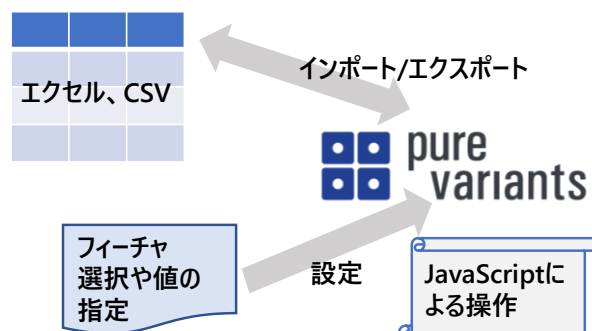
■ はじめに

pure::variants には API による JavaScript インターフェースが提供されており、JavaScript を使用して、外部に定義した一連のフィーチャを pure::variants のフィーチャモデル (*.xfrm) としてインポートすることや、バリエーション定義モデル (*.vdm) 上の選択内容を変更することができます。

この資料ではバリエーション定義モデルの設定を、外部のファイルに定義したフィーチャの選択情報や属性値で変更する例を通じて、pure::variants の API による JavaScript インターフェースについて紹介します。外部のファイルには、JavaScript との相性の良さから JSON 形式を想定しています。また JSON 形式であれば、様々なツールで定義された情報から変換することも容易です。

フィーチャモデルの外部定義とインポートについても、最後のページに紹介しています。

JavaScript インターフェース (JavaScript API) の詳細については、「pure::variants JavaScript Extensibility Guide」をご覧ください。このドキュメントは pure::variants SDK をインストールしている場合、メニューの「ヘルプ」から見るすることができます。



● pure::variants SDK のインストール

SDK は pure::variants インストール時に選択してインストールできますが、インストールされていない場合は、pure::variants のメニューから ヘルプ > 新規ソフトウェアのインストール で更新サイト (<http://www.pure-systems.com/pvce-update/>) を指定し、SDK を選択することでインストールできます¹。

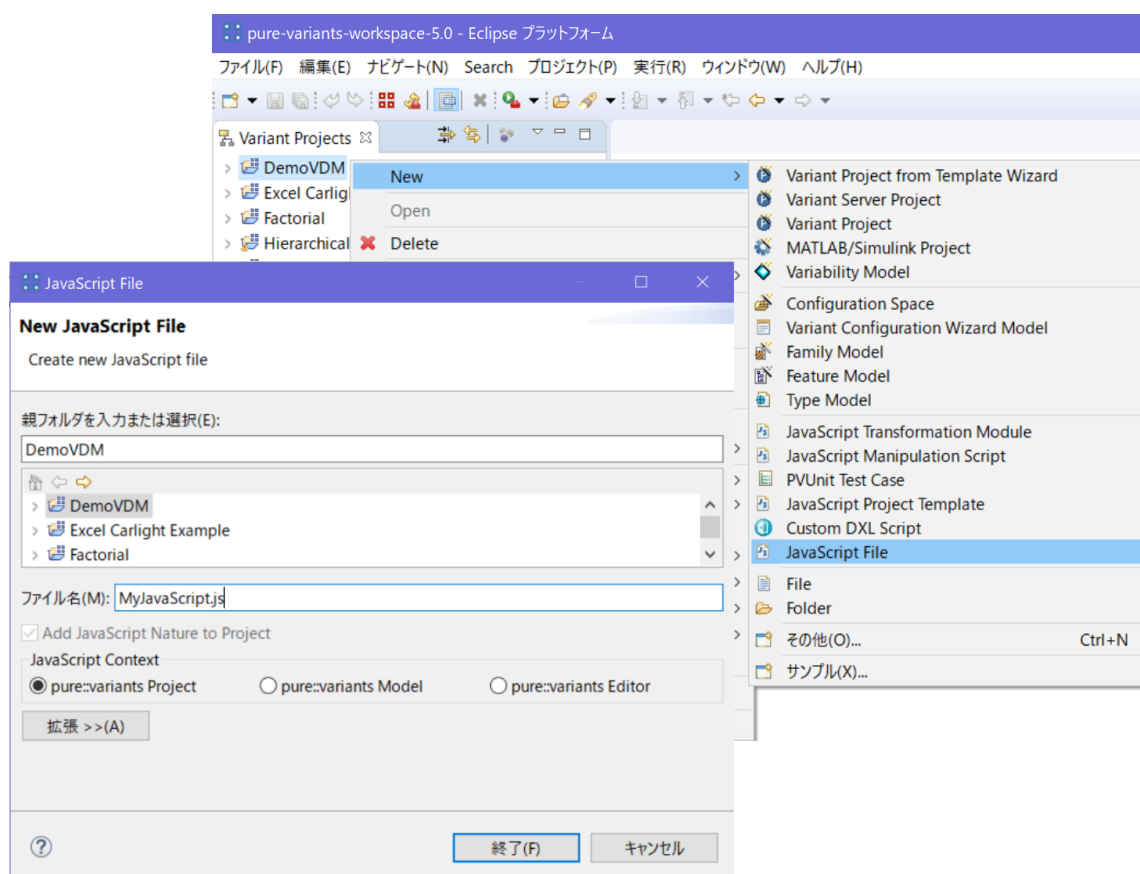
● サンプルプロジェクトのダウンロード

<https://www.fuji-setsu.co.jp/files/DemoVDM.zip>

■ モデル操作の手順

1. JavaScript の作成

pure::variants のプロジェクト内に JavaScript のスクリプトを作成するには、プロジェクトツリーから New > JavaScript File をクリックします。New JavaScript File ダイアログボックスの ファイル名 欄で JavaScript ファイルの名称を入れて終了し、ファイルの内容を入力できるようになります。



作成した MyJavaScript.js をエディタで開いて、スクリプトを入力できます。

¹ 別法として、Eclipse のメニューからファイル > 新規 > サンプル > Variant Management SDK を選択します。このカテゴリには「Extensibility Example Plugins」と「Extensibility Example Projects」がありますので、それぞれ選択してインストールできます。

スクリプトの例

```

/** Do some work on the model. E.g. get root element.
 * @param {IPVModel} model The pure::variants model.
 * @returns {IPElement} Any element
 */
function work(model) {
  // auto completion works for model.
  var element = model.getElementWithID(model.getElementsRootID());
  return element;
}

model = new PVModel();
var element = work(model);

// auto completion works for element.
var name = element.getName();

```

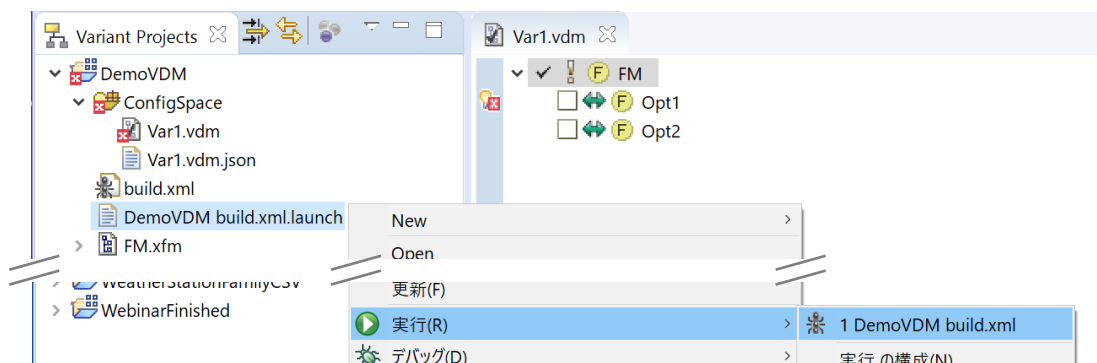
2. 外部ファイルによるフィーチャの指定

JavaScript API による操作対象のフィーチャを JSON 形式で外部のファイルに記述して指定します。非常に簡単な例として二つのフィーチャ「Opt1」と「Opt2」を指定する場合は以下のようなものになります。

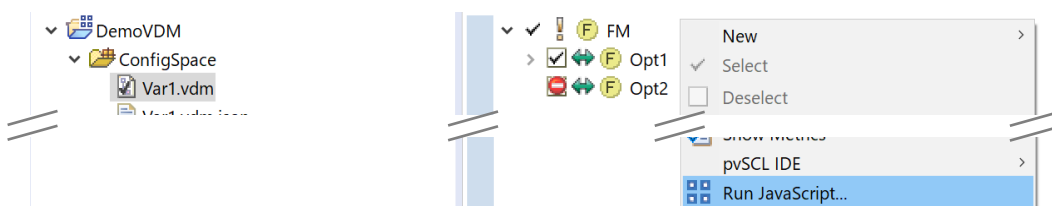
```
[ "Opt1", "Opt2" ]
```

3. JavaScript の実行

作成したJavaScriptスクリプトは、プロジェクトから ~.launch を右クリックし、実行 > ~ build.xml を選択することで実行できます。



また、プロジェクトからVDMエディタ内で右クリックし、Run JavaScript としても実行できます。



■ 外部ファイルに記述した選択指定の実行例

以下の例で、簡単なプロジェクトを用いて、JavaScript API と Ant によって外部のデータファイルを読み込んで、バリエーション定義モデルの設定変更を自動化する方法を紹介します。

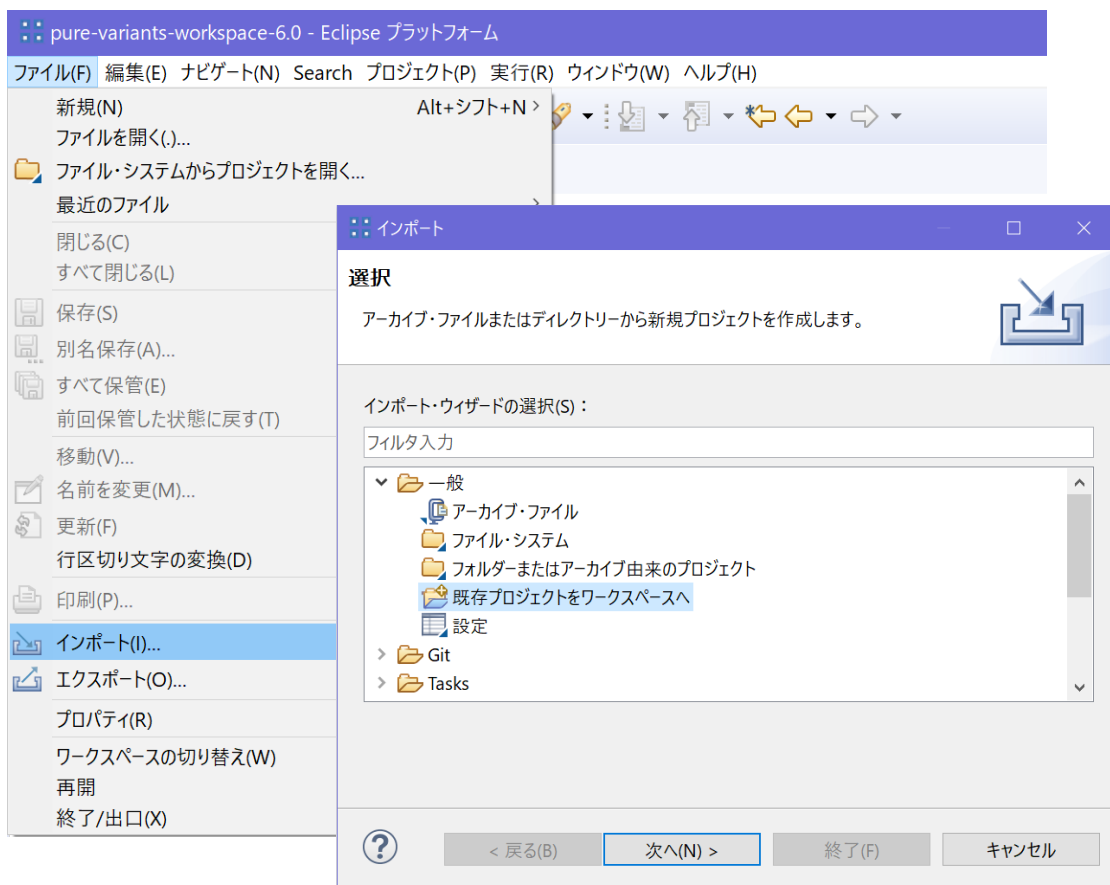
● プロジェクトの構成

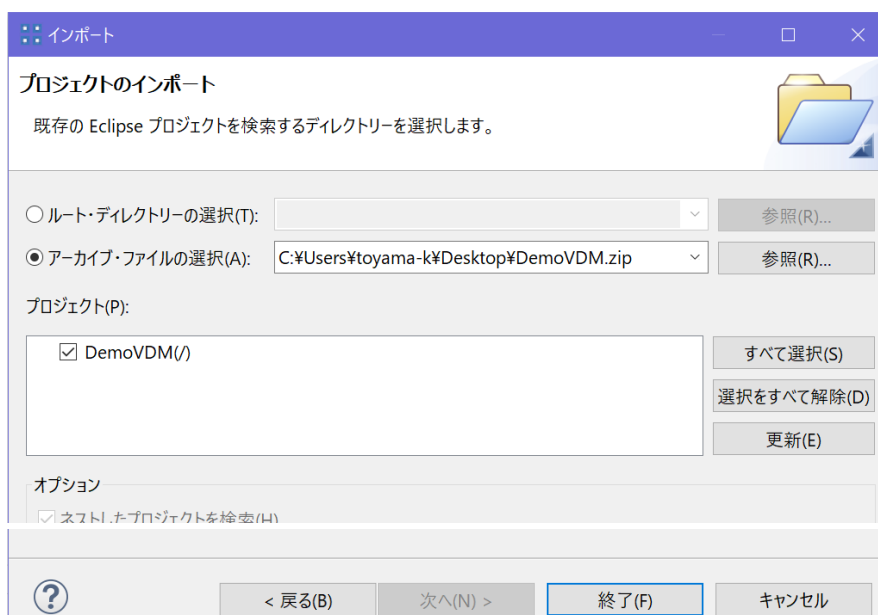
- FM.xfm … 簡単なフィーチャモデル
- ConfigSpace
 - └ Var1.vdm … 評価用に使用する VDM
 - └ Var1.vdm.json … 選択情報を定義した外部データファイル
- build.xml … Ant 自動化スクリプトであり、以下を行います
 - ・ まず、すべてのセットアップを行う (`pv.import`)
 - ・ 次に、Var1.vdm 上で JavaScript を実行
 - ・ 最後に、結果の VDM コンフィグレーションを評価
- SetSelection.js … 単純な JavaScript で、上記外部データによって選択操作を行う

● プロジェクトの使用法

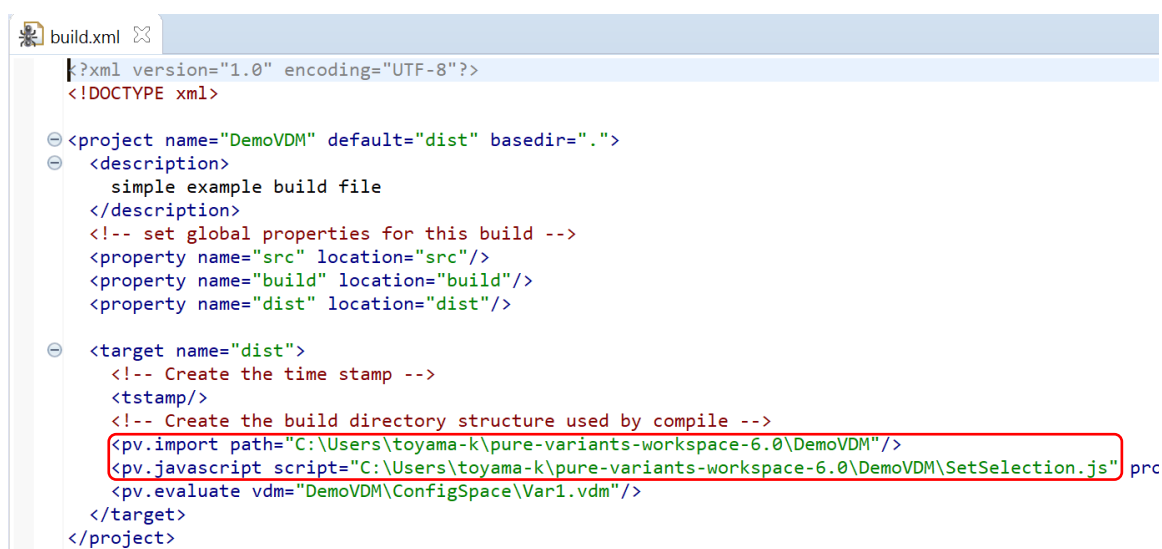
1. プロジェクトのインポート

例題プロジェクトファイルのアーカイブ (DemoVDM.zip) を適当な場所に置き、このプロジェクトをワークスペースにインポートします。



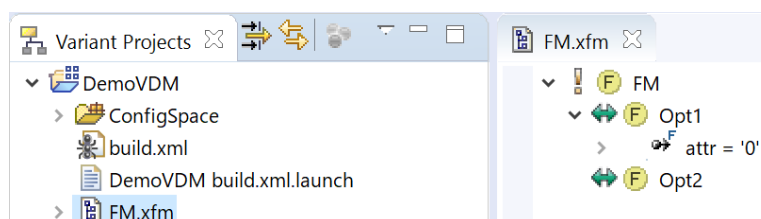


build.xml 内の "pv.import path" や "pv.javascript script" のパスをインポートしてきたワークスペースのものに変更します。

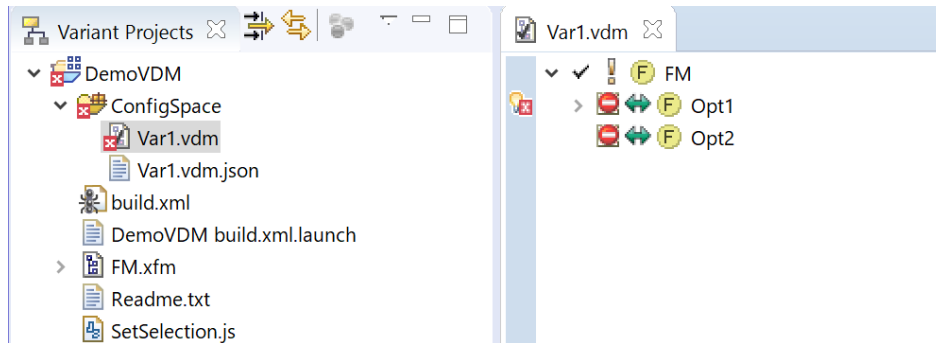


フィーチャの初期状態

プロジェクト起動時に、フィーチャモデルは下図のとおりであり、Op1 と Op2 の 2 フィーチャがあって、Op1 には Attributes 属性 attr があり、値は 0 です。



Var1.vdm は以下のとおりで、Opt1 と Opt2 は選択がなされていない状態です。これらは Alternative なフィーチャなので、いずれかを選択する必要がありますが、どちらも選択されていないため、先頭に警告（赤色）のアイコン  が出ています。



2. JavaScript でフィーチャの選択を行う

- 選択情報指定ファイル： Var1.vdm.json

```
[ "Opt1", "Opt2" ]
```

これは、フィーチャとして Opt1 と Opt2 があることを示しています。

- 設定用 JavaScript： SetSelection.js

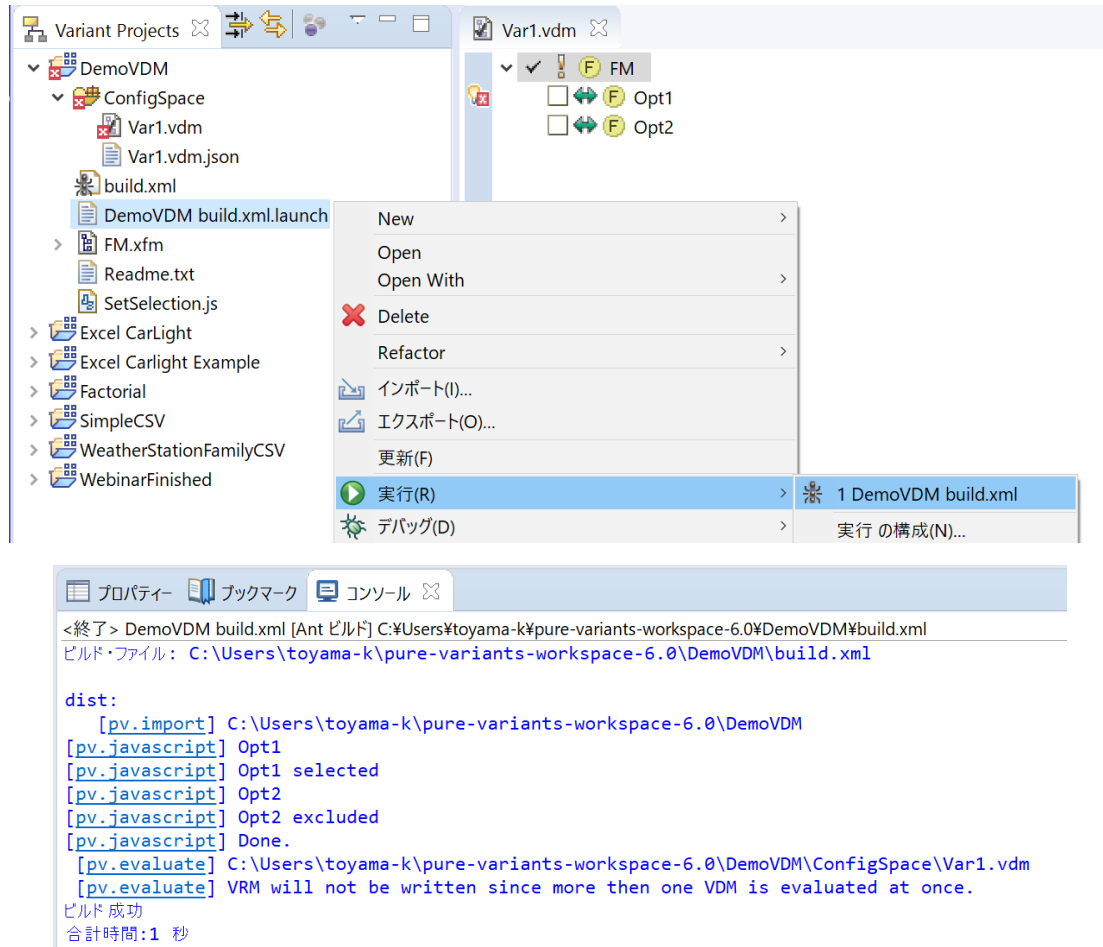
```
...
var config = JSON.parse(data);
...
e = fm.getElementWithName(config[0]);
...
    if (vdm.getState(e).getSelection() != VariantElementState().SELECTION)
    {
        op.select(e);
        console().println(e.getName() + " selected");
    }
    else
    {
        op.exclude(e);
        console().println(e.getName() + " excluded");
    }
}
```

このスクリプトでは、選択情報指定ファイルから読み込んだフィーチャ Opt1 に対して、それが選択されていない (`!= VariantElementState().SELECTION`) のであれば「選択」(`op.select(e)`) し、そうでない（選択済み）なら「非選択」(`op.exclude(e)`) と設定しています²。

² したがって、Opt1 が選択済みで、Opt2 が非選択の場合などに実行すると、両者が非選択となり、エラーとなります（「ビルド失敗」となります）。

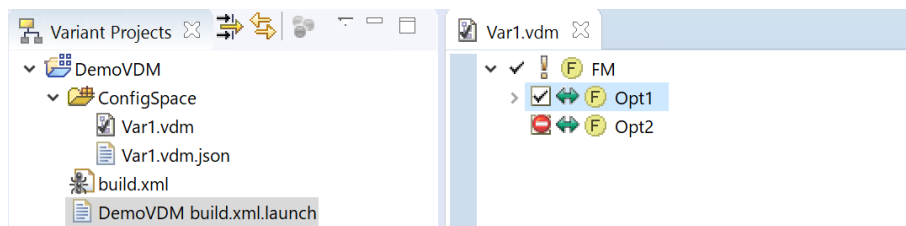
3. pure::variants から JavaScript を実行する

DemoVDM build.xml.launch を右クリックし「実行」を選択します。



選択設定されたフィーチャの状態

Var1.vdm は以下のとおり Opt1 が選択されている状態となります。



JavaScript スクリプトに以下の記述を行うことで、Opt1 に属性値を設定できます。

```
var prop = e.getPropertyWithName("attr");
...
var vel = vdm.getSelectionOfReference(e);
vel = op.change(vel);
VariantOperations().setPropertyValue(vel, prop, "123");
...
```

これは、名前 attr の属性 (Attributes) に 123 を設定するものです。プロジェクト内で提供する SetSelection.js ではコメントアウトされていますので、有効にして実行してください。

※ 実行前に、この vdm で Opt1 と Opt2 の選択をどちらも非選択と、初期状態に戻してください。

```

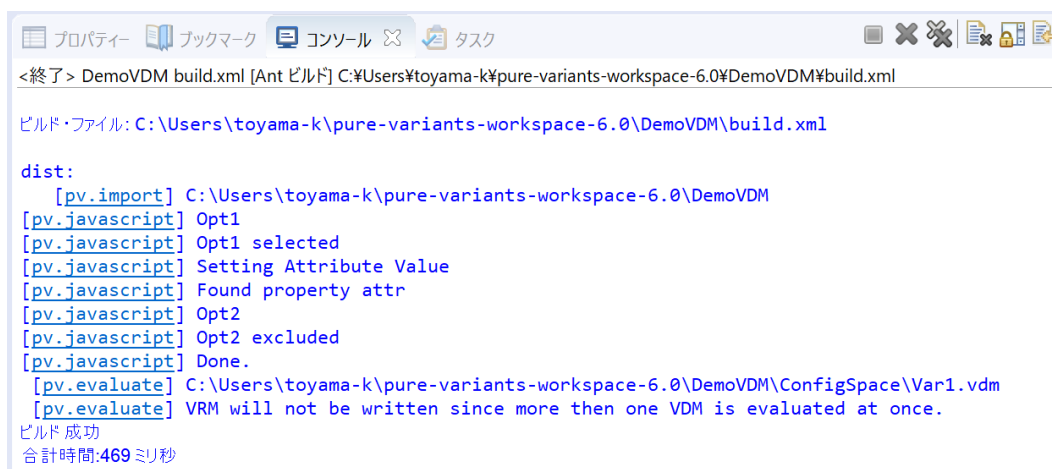
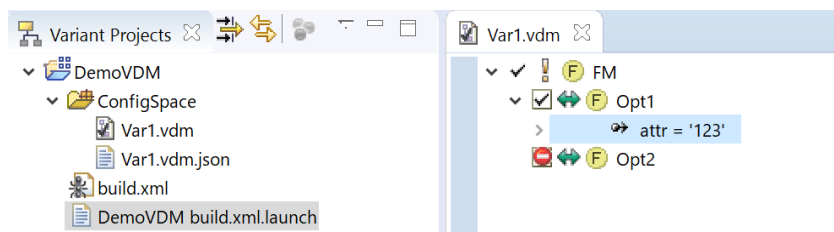
SetSelection.js
}

//console().println("Setting Attribute Value");
//var prop = e.getPropertyWithName("attr");
//if(prop != null)
//{
//  console().println("Found property " + prop.getName());
//  var vel = vdm.getSelectionOfReference(e);
//  vel = op.change(vel);
//  VariantOperations().setPropertyValue(vel, prop,"123");
//}

e = fm.getElementWithName(config[1]);
if (e != null)

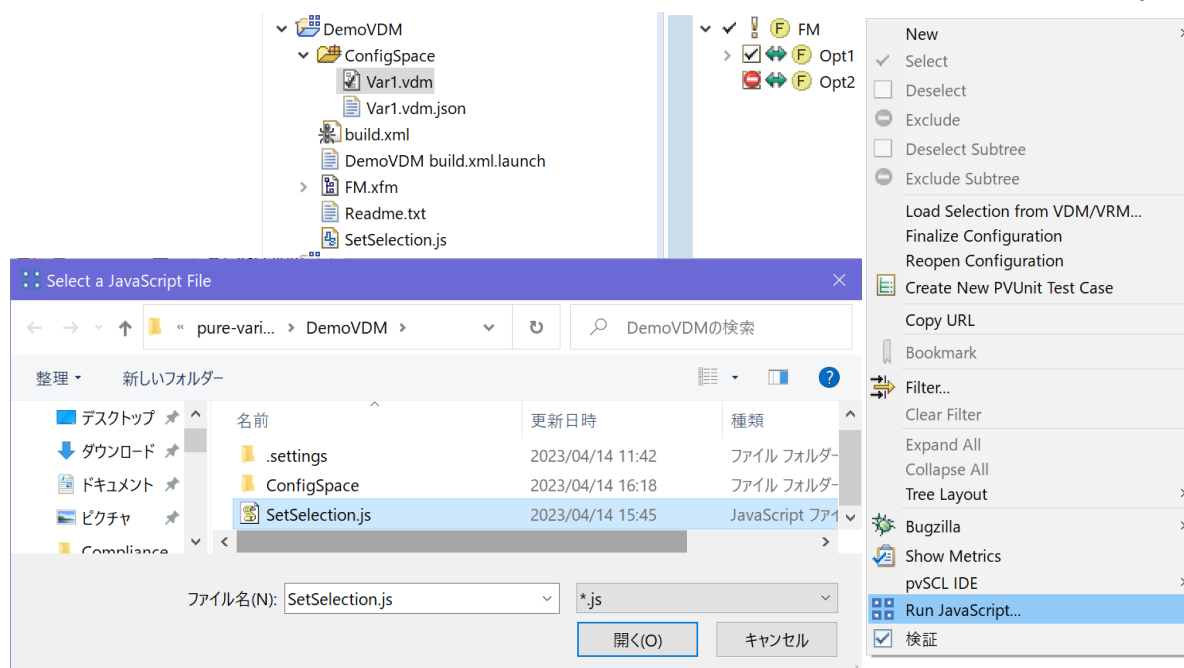
```

先と同様に DemoVDM build.xml.launch を右クリックして実行すると、Var1.vdm で Opt1 の attr が 123 に設定されます。

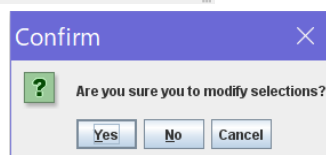


JavaScript 実行の別法

p.3 にありますように、SetSelections.js は VDM エディタの中、すなわちこの ConfigSpace のどの VDM からでも実行できます。Var1.vdmを開いてVDMエディタ内で右クリックし、Run JavaScript から Select a JavaScript File ダイアログで JavaScript ファイルを指定して実行します。



確認ダイアログが出ますので、Yes で進めます。



コマンドラインからの JavaScript バッチ実行

この JavaScript による実行は、コマンドラインからも行えます。pure::variants のインストールディレクトリ ¥cli¥ から runant.bat で、build.xml を引数に指定して実行してください。GUI での操作を行わず、コマンドラインから選択指定や値の設定ができます。

```

C:\Users\toyama-k>cd C:\Program Files\pure-systems\pv_Enterprise_6.0\cli
C:\Program Files\pure-systems\pv_Enterprise_6.0\cli>runant.bat C:\Users\toyama-k\pure-variants-workspace-6.0\DemoVDM\build.xml
Workspace: C:\Users\toyama-k\AppData\Local\Temp\runant-workspace-28442
ビルド・ファイル: C:\Users\toyama-k\pure-variants-workspace-6.0\DemoVDM\build.xml

dist:
[pv.import] C:\Users\toyama-k\pure-variants-workspace-6.0\DemoVDM
[pv.javascript] Opt1
[pv.javascript] Opt1 selected
[pv.javascript] Setting Attribute Value
[pv.javascript] Found property attr
[pv.javascript] Opt2
[pv.javascript] Opt2 excluded
[pv.javascript] Done.
[pv.evaluate] C:\Users\toyama-k\pure-variants-workspace-6.0\DemoVDM\ConfigSpace\Var1.vdm
[pv.evaluate] VRM will not be written since more then one VDM is evaluated at once.
ビルド成功

BUILD SUCCESSFUL
Total time: 4 seconds

```

■ 制約等を含むフィーチャモデルの外部定義とインポートについて

JavaScript API を利用することで、フィーチャモデルを外部で作成してインポートすることも可能です。pure::variants の JavaScript API には「モデル生成」や「フィーチャ生成」などに加えて「Relation 追加」、「Constraint 追加」、「Restriction 追加」、「Attribute 追加」などの API が用意されており、これらを使用することにより、外部ファイルでの記述から完全なフィーチャモデルを作成・インポートすることができます。（なお、User's Guide の 6.13 節に User-defined import manipulator with JavaScript の説明があります）

- * 外部データとして CSV や Excel、JSON などを用いて、Constraints の設定などを含めたフィーチャモデルを記述する定まった記法はありませんので、Constraints や Relations、Restrictions を記述して操作するコンバータープログラム（スクリプト）はすべてユーザーによって作成する必要があります。
- * たとえば、Constraints に Excel の評価機能（式）を利用しているような場合、ご自身でそのコンバーターを作成することは推奨されません。pure::variants 上の操作でそれら Constraints や Relations や Restrictions を設定する方が容易です。モデルやその表現と、Constraints などフィーチャの属性の評価エンジン+自動リゾルバー等は区別されなければならない、インポートにおいてこれらが適切に動作することが必要になるためです。

「pure::variants JavaScript Extensibility Guide」もご参考ください。

pure::variants にはフィーチャとバリエーション定義をインポートする、CSV インポーターと Excel インポーター機能もあります。これはフィーチャの Constraints や Relations、Restrictions の設定には対応しないシンプルな機能ですが、既に Excel 等にフィーチャの一覧等をまとめている場合などに活用いただけます。これについては、User's Guide の 6.13 節にある説明や、以下、CSV インポーターの資料を参考ください。

<https://www.fuji-setsu.co.jp/products/purevariants/tutorials.html#CSVimport>