Control System Studio 3.0

BOYのスクリプト

中本 崇志
takashi.nakamoto@cosylab.com

Your TRUSTED Control System Partner

COSYLAB

本日の予定

COSYLAB

- 13:30 – 13:50 BOYのスクリプトでできること
  - 話を聞きながら…
    - CSS 3.0.1 をダウンロード/インストール
      http://www-linac.kek.jp/cont/css/
    - 自分でIOCを作り、4〜5個 ai レコードを作る or
      制御用ネットワークへ接続しておく
- 13:50 – 14:10 Ruleを使ってみる
- 14:10 – 15:00 スクリプトを使ってみる1
- 15:00 – 15:15 休憩
- 15:15 – 15:30 PyDevのインストール
- 15:30 – 16:20 スクリプトを使ってみる2
- 16:20 – 16:30 まとめ・質疑応答

参考文献

- CSSのヘルプより **CSS Applications → Best OPI Yet (BOY) → Script**を参照
- BOYのサンプル
  (最後に時間があれば皆さんと一緒に使ってみます)
  - 5_1_Rules_Script.opi
  - 5_2_Script.opi
  - 5_3_Rules_Script.opi
  - 5_4_Python_Script.opi
  - 5_5_FileIOWithScript.opi
  - 5_6_UseThreadInScript.opi

スライド中の一部のサンプルコードは、上記参考文献より拝借しています。

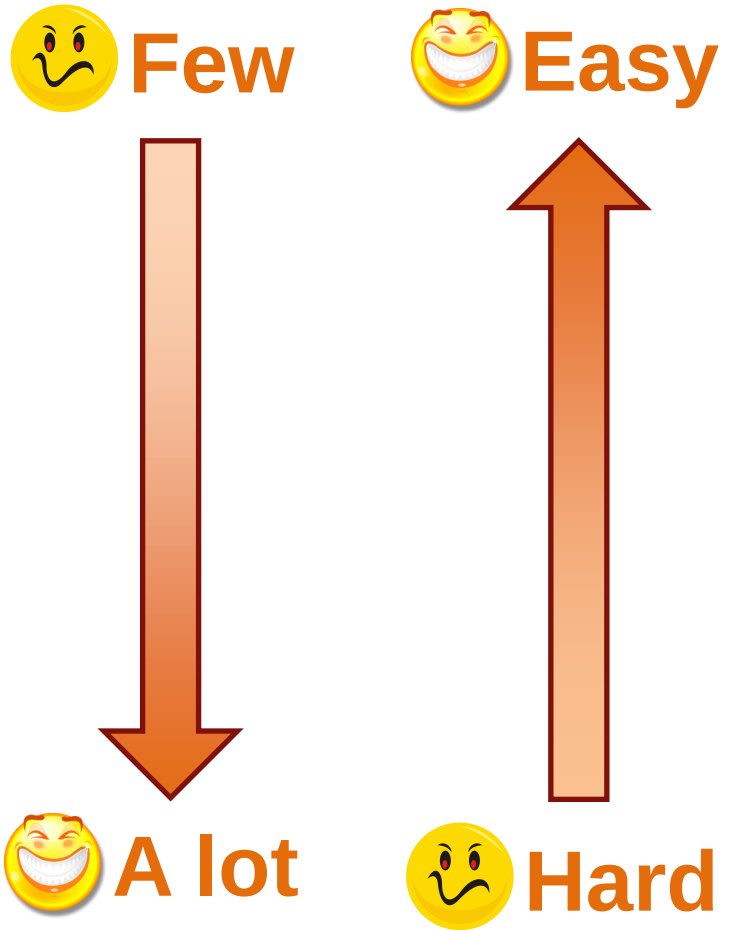
Control System Studio 3.0 BOYのスクリプト講座

BOYのスクリプトでできること

使いやすい画面を作るために

COSYLAB

できること　　難易度



- Macro
- Action
- Rule
- Script

本日の内容

- Javaでプラグインを書く

できるだけ簡単な方法で

COSYLAB

- できるだけ簡単な方法で
  - メンテナンス性の問題
- 例)
  - 同じ画面を別の機器に使いまわす
    → **Macro**
  - テキストの表示/非表示を切り替える
    → **Rule**
  - PVの値を元にアニメーションさせる
    → **Script (Python or JavaScript)**
  - Scriptでもできないことをやる
    → **Javaでプラグインを作って、BOYのウィジェットを作る**
  - それでもできないこと
    → **Javaでプラグインを作って、ビューなどを作る**

スクリプト言語と呼び出し方法

COSYLAB

- 言語
  - JavaScript
  - Python
- トリガ
  - PVの値の変化
    - Input PVsという、引数に相当するものがある
  - Actionの一つとして
    - Action Buttonを押下
    - Menu Buttonのアイテムをクリック
    - 各種ウィジェットを右クリック

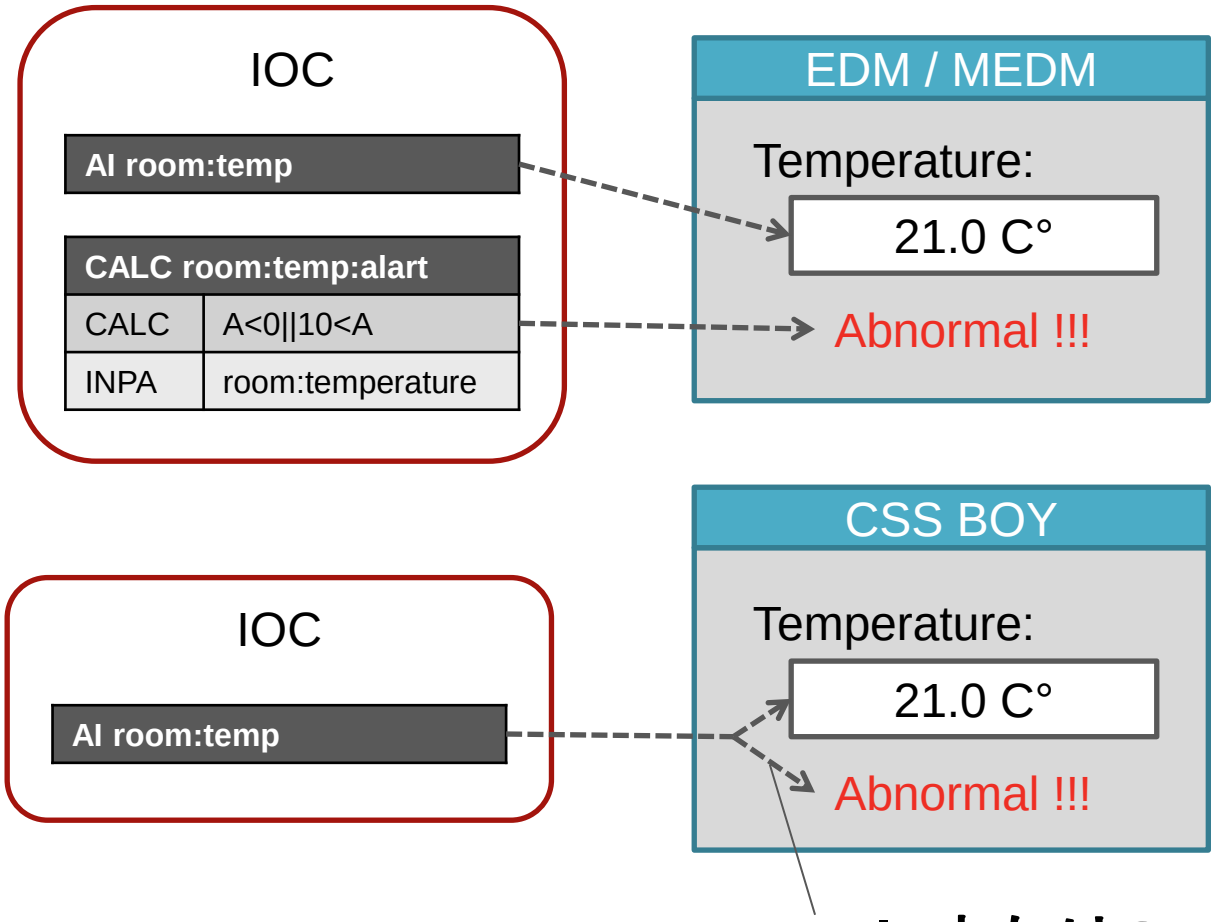
スクリプトでできること

- BOYのスクリプトでできること
  - BOYのウィジェットのプロパティの読み取り・書き換え
    - 表示・非表示の切り替え
    - 枠線の色の変更
    - アニメーション
  - PVへのアクセス (read & write)
  - それぞれの言語・標準ライブラリでできること全て
    - 豊富なライブラリを使ってファイルへのアクセスとか

IOC側での処理を減らす

COSYLAB

- 画面表示のロジックはBOY側で
  - IOCの負荷軽減、ネットワークトラフィックの軽減



どちらの言語が良い？

COSYLAB

- JavaScript
  - サンプルやヘルプではこちらがデフォルトになっている
  - 処理系はJavaで実装されたRhino
  - AJAXなどのWeb技術のおかげで、書籍も増えてきた
- Python
  - スクリプト言語として広く使われている
  - PyDevを使うことで色付きエディタ、コード補完の恩恵を受けられる (後で実際にインストールしてみます)
  - 標準的なライブラリも使える？
- 共通項
  - Javaのライブラリを使うことができる
    - CSSのプラグインの機能を利用することができる

どちらの言語が良い？

COSYLAB

- PVとウィジェットの取り扱いに関してはどちらの言語もほとんど同じです

JavaScript

```
importPackage(Packages.org.csstudio.opibuilder.scriptUtil);
var value = PVUtil.getDouble(pvs[0]);
var RED = ColorFontUtil.RED;
widget.setPropertyValue("start_angle", value);
widget.setPropertyValue("foreground_color", RED);
```

Python

```
from org.csstudio.opibuilder.scriptUtil import PVUtil
from org.csstudio.opibuilder.scriptUtil import ColorFontUtil
value = PVUtil.getDouble(pvs[0])
RED = ColorFontUtil.RED
widget.setPropertyValue("start_angle", value)
widget.setPropertyValue("foreground_color", RED)
```

あとは好みの問題

本日は、少しだけJavaScriptを
あとは、Pythonを使います

Control System Studio 3.0 BOYのスクリプト講座

Ruleを使ってみる

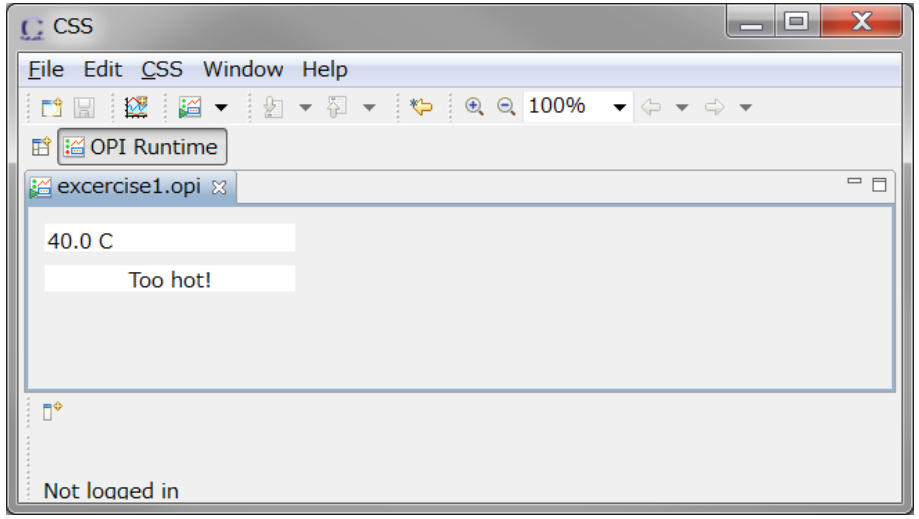
Rule

- 簡単なロジックであればRuleで十分
  - 条件分岐における条件をPVの値とJavaScriptの基本的な演算子のみで表せること
    例) 0 < pv0 && pv0 < 10
  - プロパティの書き換えのみであること
    例) Visibleプロパティ
- 仕組みはScriptに非常に似ている
  - Input PVsのどれかの変化がトリガ
  - 生成したRuleからJavaScriptを生成できる
    - これからScriptを使う人にとっては勉強になる
    - 複雑なことをやらないのであれば、JavaScriptもPythonもそれほど変わらない

演習1: 表示・非表示の切替

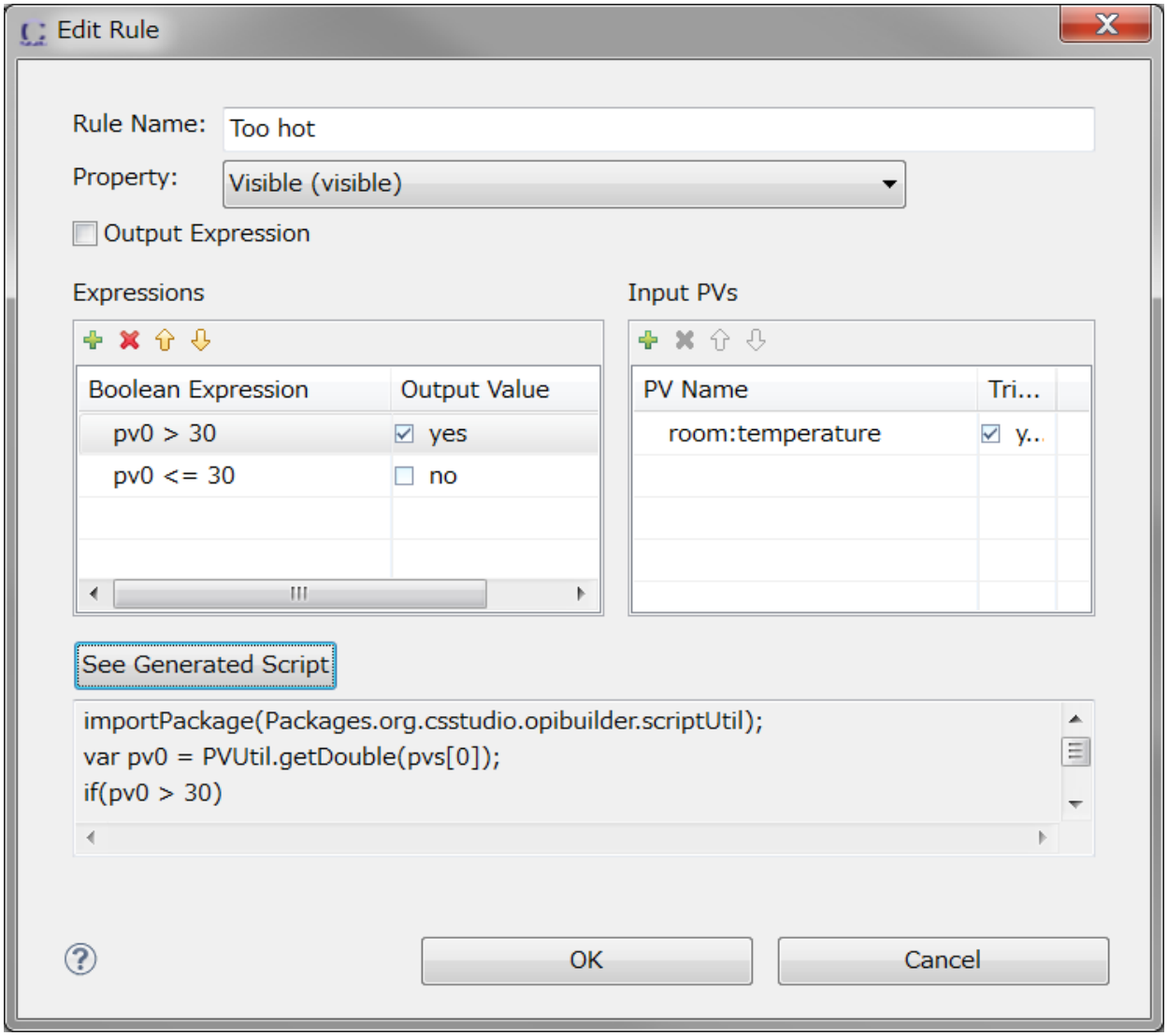
COSYLAB

- PVの値によってテキストの表示・非表示を切り替え
  - Update TextとLabelを配置
  - Labelの方にRuleを設定
    - Rule名は任意
    - Input PVsにPVを追加 (e.g. room:temperature)
    - Expressions
      - pv0 > 30 → yes
      - pv0 <= 30 → no
    - スクリプトの生成 (後で説明)



演習1: 表示・非表示の切替

COSYLAB



生成されたスクリプトを見てみる

COSYLAB

```
importPackage(Packages.org.csstudio.opibuilder.scriptUtil);
var pv0 = PVUtil.getDouble(pvs[0]);
if(pv0 > 30)
	widget.setPropertyValue("visible",true);
else if(pv0 <= 30)
	widget.setPropertyValue("visible",false);
else
	widget.setPropertyValue("visible",true);
```

- ポイント
  - org.csstudio.opibuilder.scriptUtil というパッケージをインポート
  - pvs[i]がInput PVsで設定したPV
  - PVUtil.getDouble(pvs[i]) で浮動小数点値をPVから取得
  - widget.setPropertyValue(“プロパティID”, 値) でウィジェットのプロパティをセット

Control System Studio 3.0 BOYのスクリプト講座

スクリプトを使ってみる1

スクリプト

- 利用可能なスクリプト言語
  - Python
  - JavaScript
- スクリプトはファイルとしてワークスペースに保存
  - BOYのopiファイルなどと同じ
  - 機能単位で一つのファイルにしておく必要がある
    （実行するサブルーチンを指定できない）

演習2: Ruleをスクリプトで置換

COSYLAB

- さきのRuleと同じものをPythonで実装
  - LabelのRuleは削除
  - Pythonスクリプトファイルを作成

```
from org.csstudio.opibuilder.scriptUtil import PVUtil
pv0 = PVUtil.getDouble(pvs[0])
if ( pv0 > 30 ):
    widget.setPropertyValue("visible", True)
else:
    widget.setPropertyValue("visible", False)
```

  - Labelにスクリプトを追加
  - Input PVsにPVを追加

ウィジェットへのアクセス

- **widget**
  関連付けられているウィジェット
- **display.getWidget(“ウィジェット名”)**
  同じスクリーン上にあるウィジェット

演習3: 他のウィジェット

- 別のウィジェットを変更してみる
  - Rectangleウィジェットを追加
    - Nameプロパティを “Rect1” に設定
  - 前のPythonスクリプトで

```
display.getWidget(“Rect1”).setPropertyValue(“visible”, True)
```

と

```
display.getWidget(“Rect1”).setPropertyValue(“visible”, False)
```

をそれぞれ追加

演習3-1: Displayに関連付け

COSYLAB

- 複数のウィジェットを操作するのであれば、スクリプトはDisplayに関連づけるのが良い
  - Labelからスクリプトを削除
  - Displayに先ほどのスクリプトを追加
  - スクリプトの以下の部分をなおす

```
widget.setPropertyValue("visible", True)
 → display.getWidget("Label").setPropertyValue("visible", True)

widget.setPropertyValue("visible", False)
 → display.getWidget("Label").setPropertyValue("visible", False)
```

ウィジェットのプロパティ

COSYLAB

- **getPropertyValue(“プロパティID”)**
  - プロパティの取得
- **setPropertyValue(“プロパティID”, 値)**
  - プロパティの設定
  - プロパティの型は**getPropertyValue()**で返ってくるものと合致していなければなりません
- 型
  - 詳細はCSSのヘルプより **CSS Applications → Best OPI Yet (BOY) → Script → Property Type**を参照
  - 代表的な型
    - Boolean
    - Integer
    - String
    - …

代表的なプロパティ

COSYLAB

プロパティ名	プロパティID	型	Pythonでの型
Name	name	String	unicode
Enabled	enabled	Boolean	bool
Visible	visible	Boolean	bool
Border Color	border_color	Color	OPIColor
Border Style	border_style	Combo	int
Border Width	border_width	Integer	int
Background Color	background_color	Color	OPIColor
Foreground Color	foreground_color	Color	OPIColor
Height	height	Integer	int
Width	width	Integer	int
X	x	Integer	int
Y	y	Integer	int

演習4: さまざまなプロパティ

COSYLAB

- “Rect1”の大きさを変更してみる

```
from org.csstudio.opibuilder.scriptUtil import PVUtil
pv0 = PVUtil.getDouble(pvs[0])
display.getWidget("Rect1").setPropertyValue("width", pv0)
```

- 色とComboの扱い

```
if (pv0 > 30):
        display.getWidget("Rect1").setPropertyValue("background_color",¥
                                ColorFontUtil.getColorFromRGB(255, 128, 0))
        display.getWidget("Rect1").setPropertyValue("border_style", 8)
else:
        display.getWidget("Rect1").setPropertyValue("background_color",¥
                                                    ColorFontUtil.BLUE)
        display.getWidget("Rect1").setPropertyValue("border_style", 1)
```

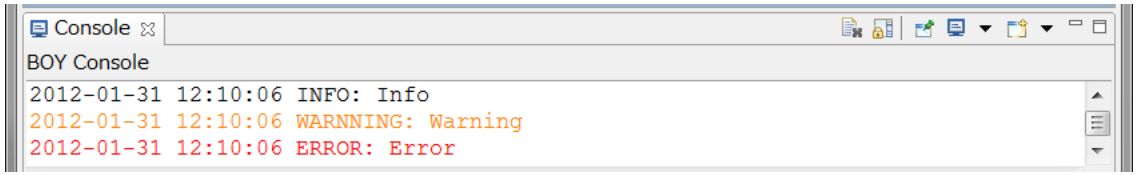
演習4-1: 型のチェック

COSYLAB

❑ Consoleへの出力方法

```
from org.csstudio.opibuilder.scriptUtil import ConsoleUtil

ConsoleUtil.writeInfo("…")
ConsoleUtil.writeWarning("…")
ConsoleUtil.writeError("…")
```



❑ 型のチェック

```
from org.csstudio.opibuilder.scriptUtil import ConsoleUtil

border_color = display.getWidget("Rect1"). ¥
                              getPropertyValue("border_color")
ConsoleUtil.writeInfo(str(type(border_color)))
```

PVへのアクセス

- **pvs[0], pvs[1], …**
  Input PVsに列挙されているPV
- **triggerPV**
  トリガの元になったPV
  （どのPVがトリガ元なのかのチェックに使う）
- **widget.getPV()**
  display.getWidget(“ウィジェット名”).getPV()
  ウィジェットに関連付けられているPV

スクリプト内で使いたいPVについては、
Input PVsに列挙しておくこと！

演習5:

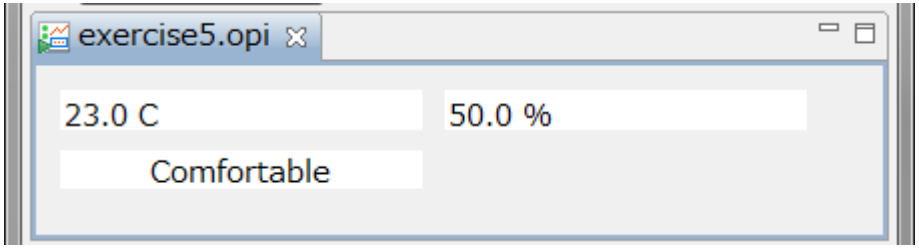
COSYLAB

- 2つのPVをトリガとするスクリプト
  - もう一つInput PVsにPVを追加

```
from org.csstudio.opibuilder.scriptUtil import PVUtil

t = PVUtil.getDouble(pvs[0])    ←―― 温度だと思ってください
h = PVUtil.getDouble(pvs[1])    ←―― 湿度だと思ってください

if (20 <= t and t <= 30 and 45 <= h and h <= 85):
        display.getWidget("Label").setPropertyValue("visible", True)
else:
        display.getWidget("Label").setPropertyValue("visible", False)
```



- 時間があれば、Triggerのところを切り替えて、どうなるのか試してみてください。

PVの操作

COSYLAB

- **PVUtil.getDouble(pv)**
  PVUtil.getLong(pv)
  PVUtil.getDoubleArray(pv)
  PVUtil.getLongArray(pv)
  PVUtil.getString(pv)
  値の取得
- **PVUtil.getSeverity(pv)**
  Severityの取得 (0: OK、-1: Invalid、1: Major、2: Minor)
- **PVUtil.getTimeInMilliseconds(pv)**
  タイムスタンプの取得
- **pv.setValue(val)**
  値の書き込み
- 詳細はCSSのヘルプより **CSS Applications → Best OPI Yet (BOY) → Script → Access PV**を参照

Control System Studio 3.0 BOYのスクリプト講座

PyDevのインストール

PyDev

COSYLAB

- Eclipse向けのPython開発環境
- CSSにもそのままインストールできます
  - Pythonスクリプトの高度な編集
    色づけ、インデント、自動補完など
  - ただし、BOY向けにかかれたスクリプトはPyDev上では実行できません
- インストール方法
  - **Edit → Preferences → Install/Update → Available Software Sites**から KEK/J-PARC 向けのものを削除する必要があります (とあるエラーの回避策)
  - CSSのヘルプより **CSS Applications → Best OPI Yet (BOY) → Script → Python Script** を参照

Jythonについて

COSYLAB

- CSSについてくるJythonはインタプリタだけのようです
- Jythonをインストールすることで標準ライブラリが使えるようになります
  - **Edit** → **Preferences** → **CSS Applications** → **Display** → **BOY** → **OPI Runtime** の PYTHONPATH に C:¥jython2.5.2¥Lib を追加
- すでにPythonがインストールされているのであれば、Jythonのインストールは必要ないかもしれません
  - その場合には、PyDevの方でCPythonインタプリタを指定してください
  - BOY上ではCSSについてくるJythonインタプリタが用いられます
    - 上記のPYTHONPATHは、CPythonについてくるものを指定してみてください (どうなるのか未確認)

Control System Studio 3.0 BOYのスクリプト講座

Scriptを使ってみる2

特殊なPVの活用

COSYLAB

- CSSのヘルプより**CSS Core → Process Variables**を参照
- これらの特殊なPVを活用することでいろいろなことが簡単に実現できるようになる
- sys://
  - システム情報を取得するのに活用
- sim://
  - アニメーションなんかに使える
- loc://
  - グローバル変数的な使い方
- const://
  - 定数

特殊なPV sys://

COSYLAB

- システム情報を使うのに活用

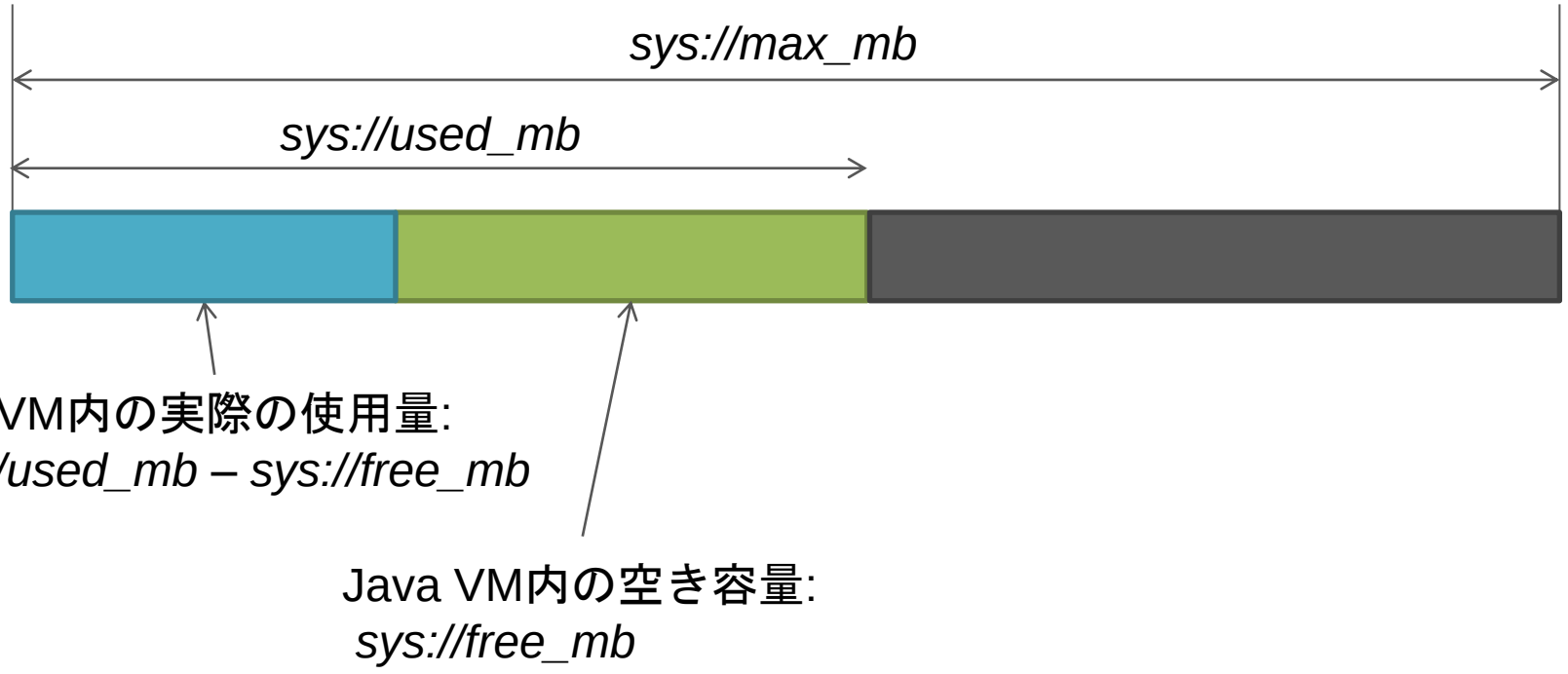
PV名	内容
time	現在時刻
free_mb	Java VM内の空きメモリ (MB)
used_mb	Java VMの使用済みメモリ (MB)
max_mb	Java VMに割り当て可能な最大のメモリ容量 (MB)
host_name	ホスト名
system.*	システムプロパティ ※ 例) system.os.name

※ http://java.sun.com/javase/ja/6/docs/ja/api/java/lang/System.html#getProperties()

演習6: Java VM内のメモリ使用量

COSYLAB

- 以下のような棒グラフを作ってください
  - Rectangleを3つ配置
  - それぞれのX, Widthをスクリプトで計算する
    - 3つのPVをInput PVsとする
  - Tooltipに “128.13 MB” というように表示させる



演習6: Java VM内のメモリ使用量

COSYLAB

```
from org.csstudio.opibuilder.scriptUtil import PVUtil

used_mb = PVUtil.getDouble(pvs[0])
free_mb = PVUtil.getDouble(pvs[1])
max_mb = PVUtil.getDouble(pvs[2])

total_width = 300
leftmost_x = 20
used_width = (used_mb - free_mb) / max_mb * total_width
free_width = free_mb / max_mb * total_width
rest_width = (max_mb - used_mb) / max_mb * total_width


display.getWidget("USED_IN_JVM").setPropertyValue("x", leftmost_x)
display.getWidget("USED_IN_JVM").setPropertyValue("width", used_width)
display.getWidget("USED_IN_JVM").setPropertyValue("tooltip", str(used_mb-free_mb) + "MB")
display.getWidget("FREE_IN_JVM").setPropertyValue("x", leftmost_x + used_width)
display.getWidget("FREE_IN_JVM").setPropertyValue("width", free_width)
display.getWidget("FREE_IN_JVM").setPropertyValue("tooltip", str(free_mb) + "MB")
display.getWidget("REST_FOR_JVM").setPropertyValue("x", leftmost_x + used_width + free_width)
display.getWidget("REST_FOR_JVM").setPropertyValue("width", rest_width)
display.getWidget("REST_FOR_JVM").setPropertyValue("tooltip", str(max_mb - used_mb) + "MB")
```

特殊なPV sim://

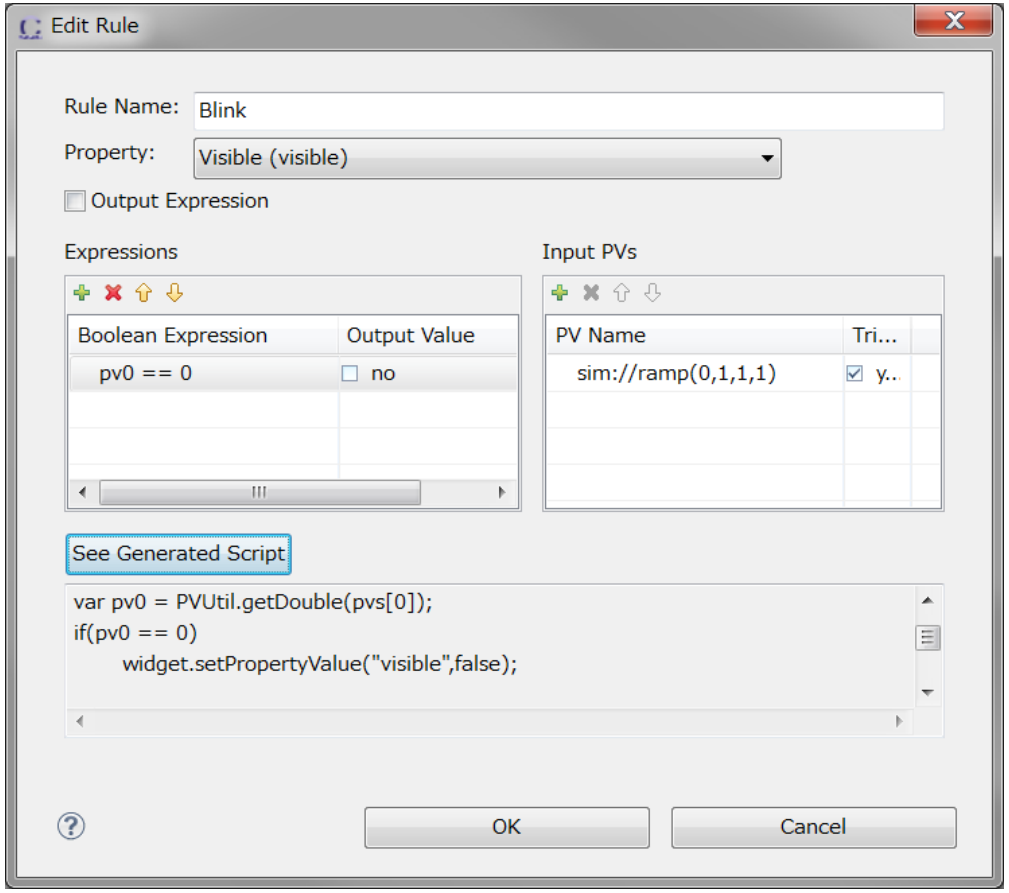
COSYLAB

- **sim://noise(min, max, rate)**
  - minとmaxの間のランダムな値
  - rate秒ごとに更新
- **sim://sine(min, max, step, rate)**
  - minとmaxの間で**サインカーブ**
  - 1周期をstep分割し、rate秒ごとに更新
- **sim://ramp(min, max, step, rate)**
  - minとmaxの間で**鋸波波**
  - rate秒ごとにstepだけ上昇 (stepは負でもOK)

演習7: 点滅させる

COSYLAB

- 1秒ごとに表示・非表示を切り替え
  - まず最初に sim://ramp(0,1,1,1) をData Browserで確認
  - ここではRuleを使いましょう



特殊なPV loc:// と const://

- loc://
  - CSS内で使える変数
  - loc://変数名(初期値)
    - すでに初期化されていれば、初期値は無視される
  - 配列もできる
    loc://foo(1.23, 2.2, 3, 4.5, 6.7)
- const://
  - スクリプトのInput PVsとして指定できる
  - const://(定数)
  - 配列もできる
    - const://(1.23, 2.2, 3, 4.5, 6.7)
  - 配列でなければ const://() は省略可能

演習8: loc://を使ってみる

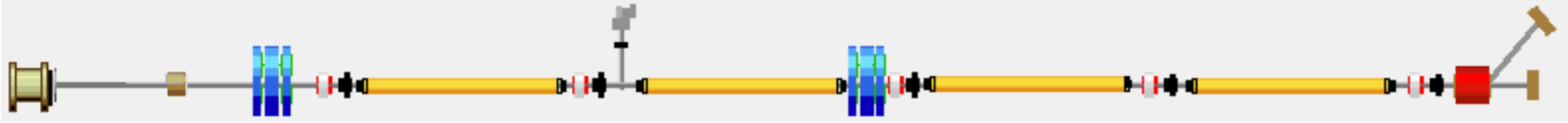
COSYLAB

- Probeで loc://FOO を定義
  - 初期値を指定したい場合には、 loc://FOO(9.2) などとする
- Data Browser で loc://FOO を表示させる
- Probeで loc://FOO を変更させてみる

演習9: ビーム電流をプロット

COSYLAB

- 複数のPVから配列を作成
  - DCCT1:CUR, DCCT2:CUR, DCCT3:CUR, …
  - これらの値を loc://CUR に配列として格納
  - XYGraphに loc://CUR を表示させる
    - X軸には const:// を活用
    - Concatenate Dataは no に



Example EPICS IOC application “Virtual LINAC”より拝借した絵

http://www.aps.anl.gov/epics/download/examples/index.php

演習ex1: 標準Libとアクション

- 標準ライブラリのパスを指定
  - **CSS Applications → Display → BOY → OPI Runtime** よりPYTHONPATHに標準ライブラリのルートディレクトリを指定
  - CSSを再起動
  - 標準のライブラリを使うスクリプトを書く
- アクションボタンを配置
  - ActionからPythonScriptを呼ぶ

演習 ex2: BOYのサンプル

COSYLAB

- BOYのスクリプト関連のサンプルを探検
  - **CSS → Display → Install OPI Examples**
  - スクリプトが関連するサンプル
    - 5_1_Rules_Script.opi
    - 5_2_Script.opi
    - 5_3_Rules_Script.opi
    - **5_4_Python_Script.opi**
      - ダイアログの表示の仕方
    - **5_5_FileIOWithScript.opi**
      - アクションボタンでスクリプトを起動
      - ファイルの読み書き
      - ファイルダイアログの使い方
    - **5_6_UseThreadInScript.opi**
      - 別スレッドでの処理

本日の内容

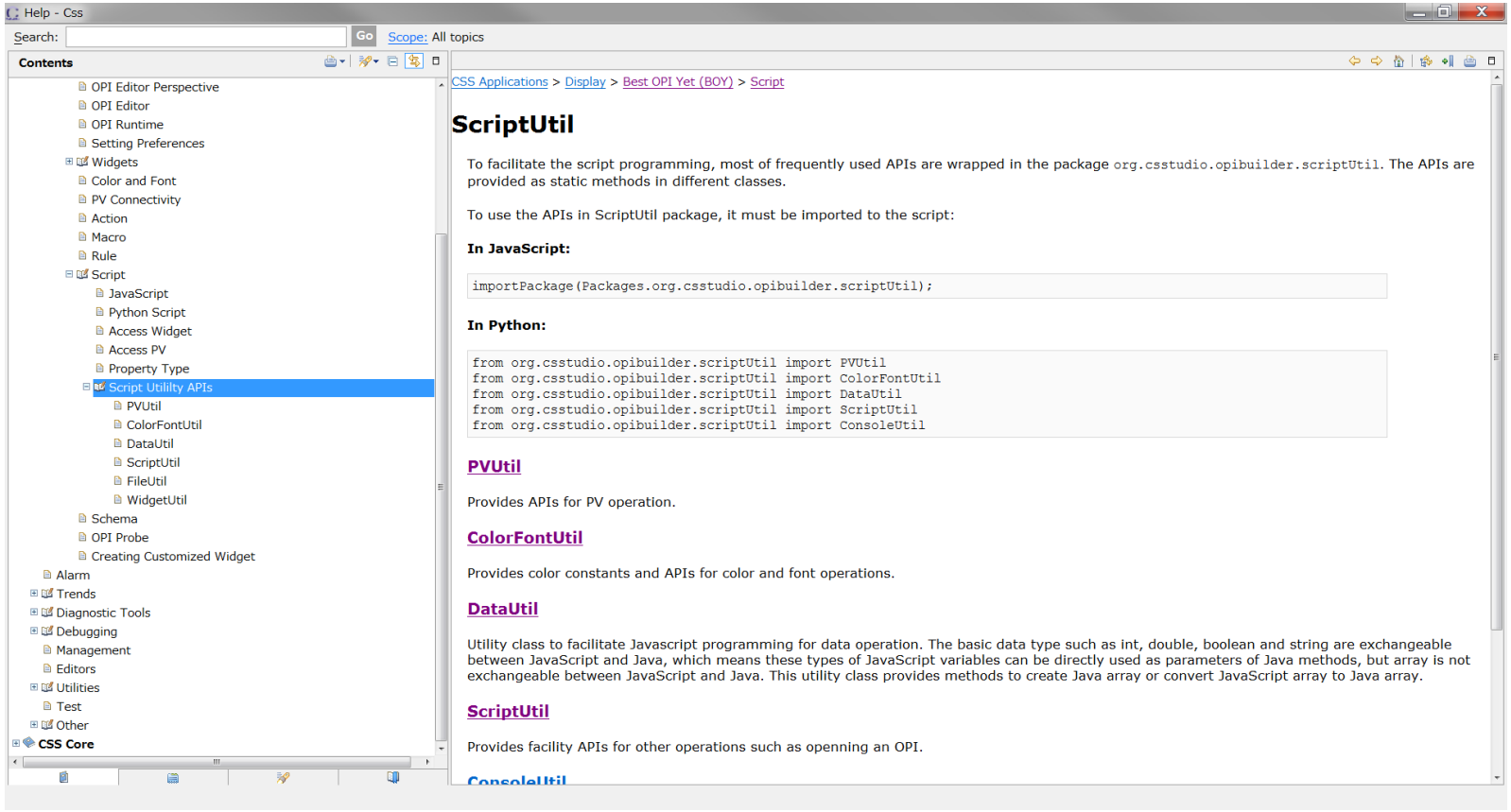
次に勉強するとしたら…

Control System Studio 3.0 BOYのスクリプト講座

最後に

リファレンス

- CSSのヘルプより **CSS Applications → Best OPI Yet (BOY) → Script → Script Utility APIsを参照**



スクリプトの管理

COSYLAB

- スクリプトの管理は意外と大変
  - どのopiファイルがどのスクリプトを使っているのかは一目では分からない
  - いろいろやっていくうちにファイルの数がどんどんと増えていく
- スクリプトの管理をするにあたって
  - スクリプトを置いておく場所(ディレクトリ構造)を決める
  - ファイル命名規則を作っておく

まとめ

- Ruleの使い方
- PVの変化を契機としてスクリプトを実行
  - PVの値の取得
  - ウィジェットのプロパティんの操作
- PyDevのインストール
- 特殊なPVの使い方