2012年2月15日 (水)

# 初心者向けCSS講習会2@KEK

Takashi Nakamoto – takashi.nakamoto@cosylab.com

COSYLAB

# 講習会の概要

COSYLAB

- 前回の講習会の参加者を対象に
  - Data Browser
  - BOYのより進んだ使い方
    - Macro
    - color.def, font.defなどBOYの設定
    - Rule
    - Scriptの第一歩

# 時間割

COSYLAB

- 10:00 – 11:00 「Data Browser」
- 11:00 – 11:30 「BOYのMacro」
- 11:30 – 12:00 「BOYの設定」
- 12:00 – 13:30 *昼休憩*
- 13:30 – 13:45 「BOYのRuleを使ってみる」
- 13:45 – 14:20 「BOYのScriptの初歩」
- 14:20 – 14:30 質疑応答

初心者向けCSS 講習会2@KEK

# DATA BROWSER

COSYLAB

# Data Browser

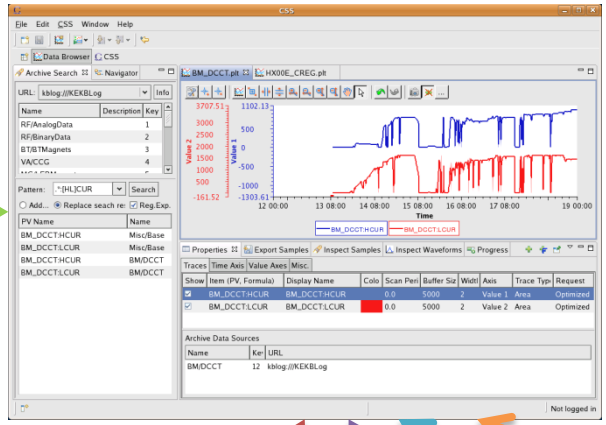
COSYLAB

- 時系列データを見るためのツール
  - Live Dataとアーカイバから取得したデータをシームレスにグラフに表示
    - Live Data: プロット画面を開いている間にChannel Accessから取得したデータ
  - 対応しているアーカイバ
    - RDB (MySQL, PostgreSQL, Oracle)
    - Channel Archiver
    - kblog (KEK版CSS 3.0.1以降)
      http://www-linac.kek.jp/cont/epics/css/css_databrowser_kblog_usage_public.pdf
- BOYの画面にData Browserのグラフを貼り付けることも可能

# Data Browserによるデータ取得

COSYLAB

EPICS Channel Access
(Live Data)



Channel Archiver

kblog

PostgreSQL

MySQL

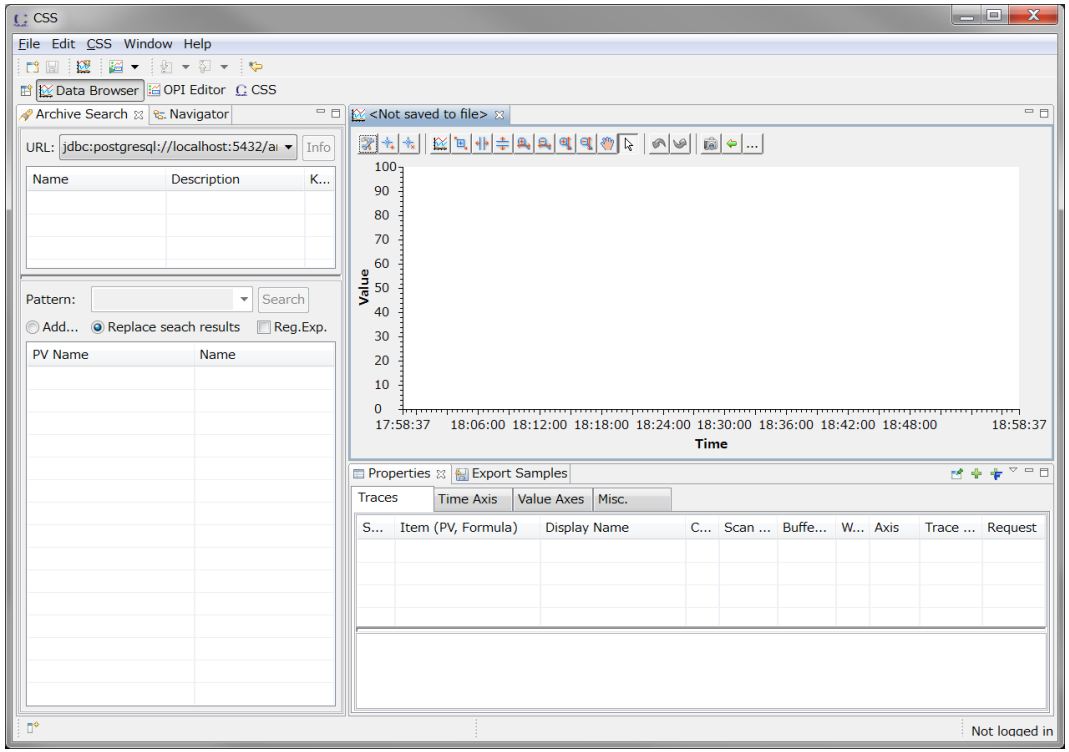
EPICS IOC

EPICS IOC

# Data Browserパースペクティブ

COSYLAB

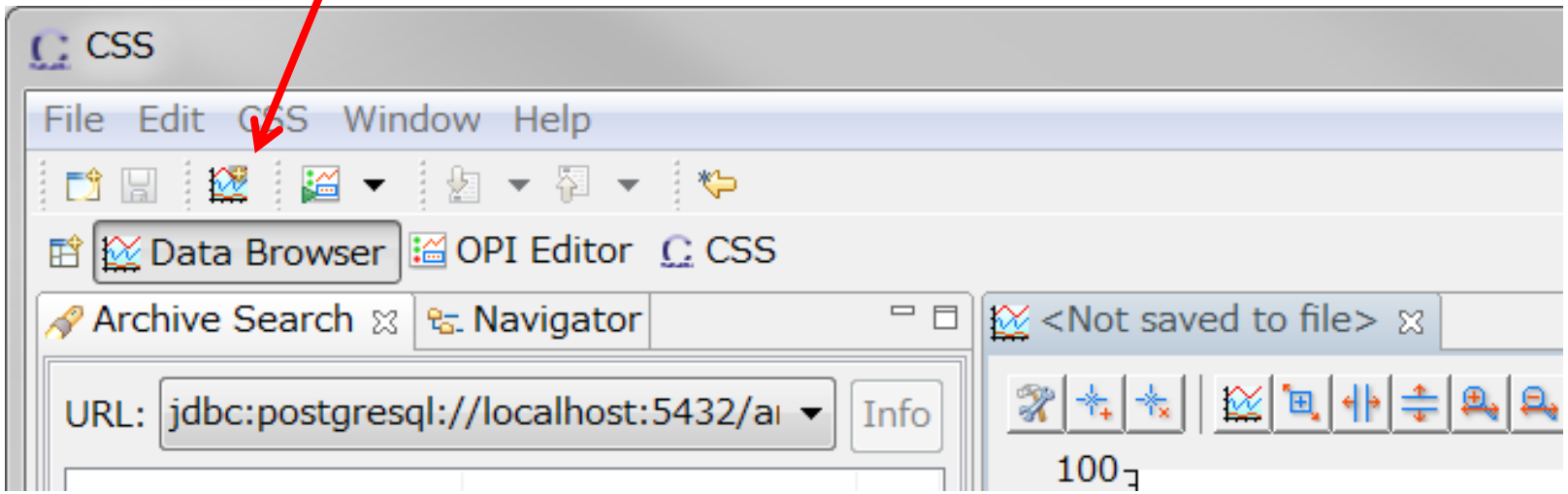
- ビューの切り替えは適宜
  - 左側: Archive Search、Navigator
  - 下側: Properties、Export Samples
    Inspect Samples、Inspect Waveform, etc.



# Data Browserの基本

COSYLAB

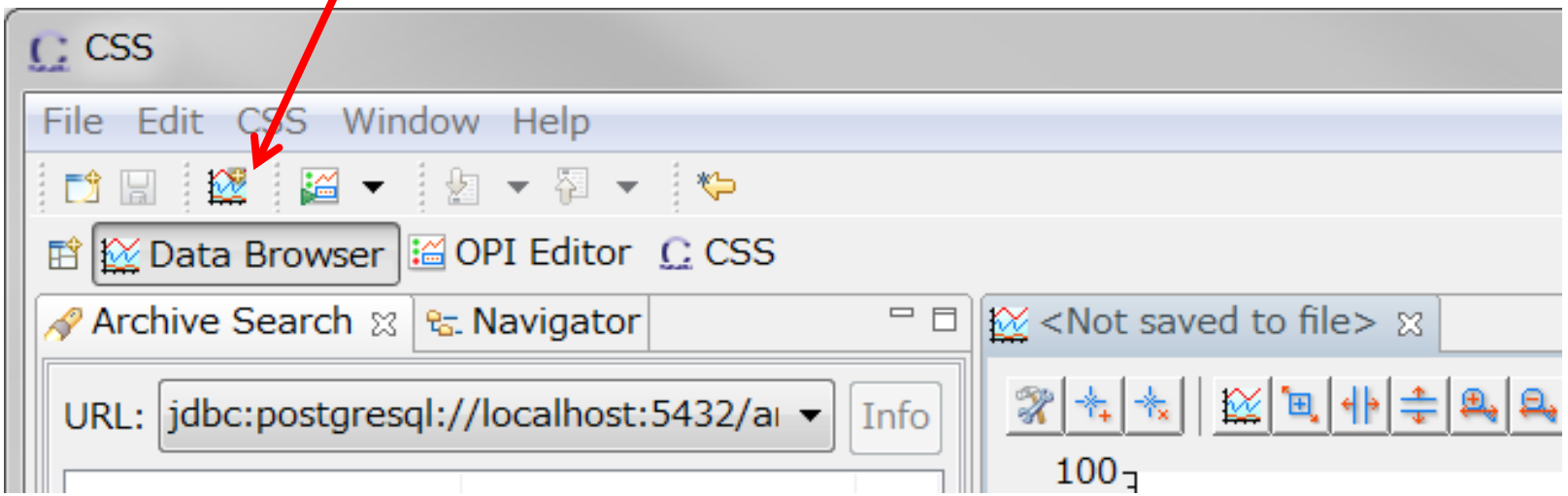
- ファイルで管理
  - Navigatorビューを使います
  - 1つのグラフで1ファイル
  - ファイルの拡張子は .plt
    - このボタンをクリックして新規作成



# 演習 1-1: Data Browserの基本

COSYLAB

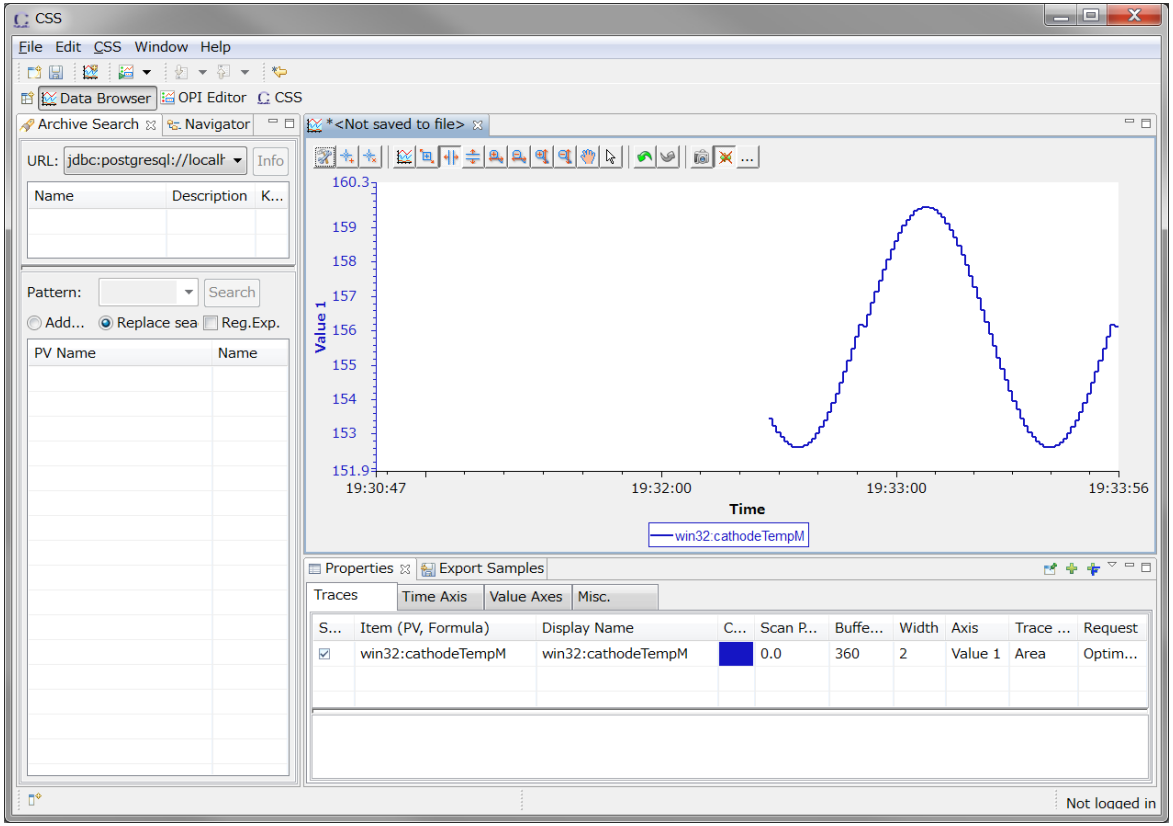
- Data Browserパースペクティブを開く
  - **Window → Open Perspective → Other → Data Browser**
- プロット画面の新規作成



# 演習 1-2: Live Dataの取得

COSYLAB

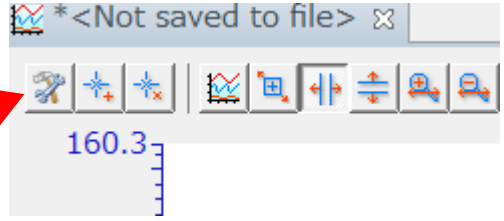
- PVのモニタ
  - **プロット画面を右クリック → Add PV**
  - *ET_kekb:cathodeTempM* を追加

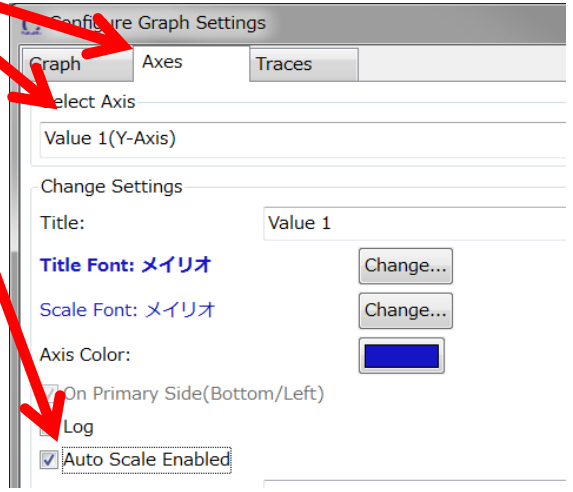


# 演習 1-3: とりあえず楽にモニタ

COSYLAB

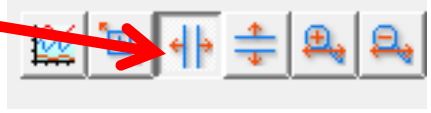
- 縦軸をオートスケールに
  - 一番左端のボタンをクリック
  - Axesタブを開き、Y軸を選択
  - Auto Scale Enabledにチェック
  - OKボタンをクリック




- スクロールをオンにする
  - このボタンをクリック

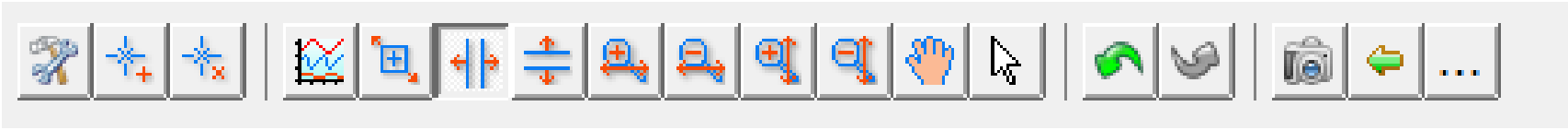
- 横軸の幅は
  - このボタンをクリックして、プロット画面で調整



# Data Browserの機能

COSYLAB

- プロットツールバー

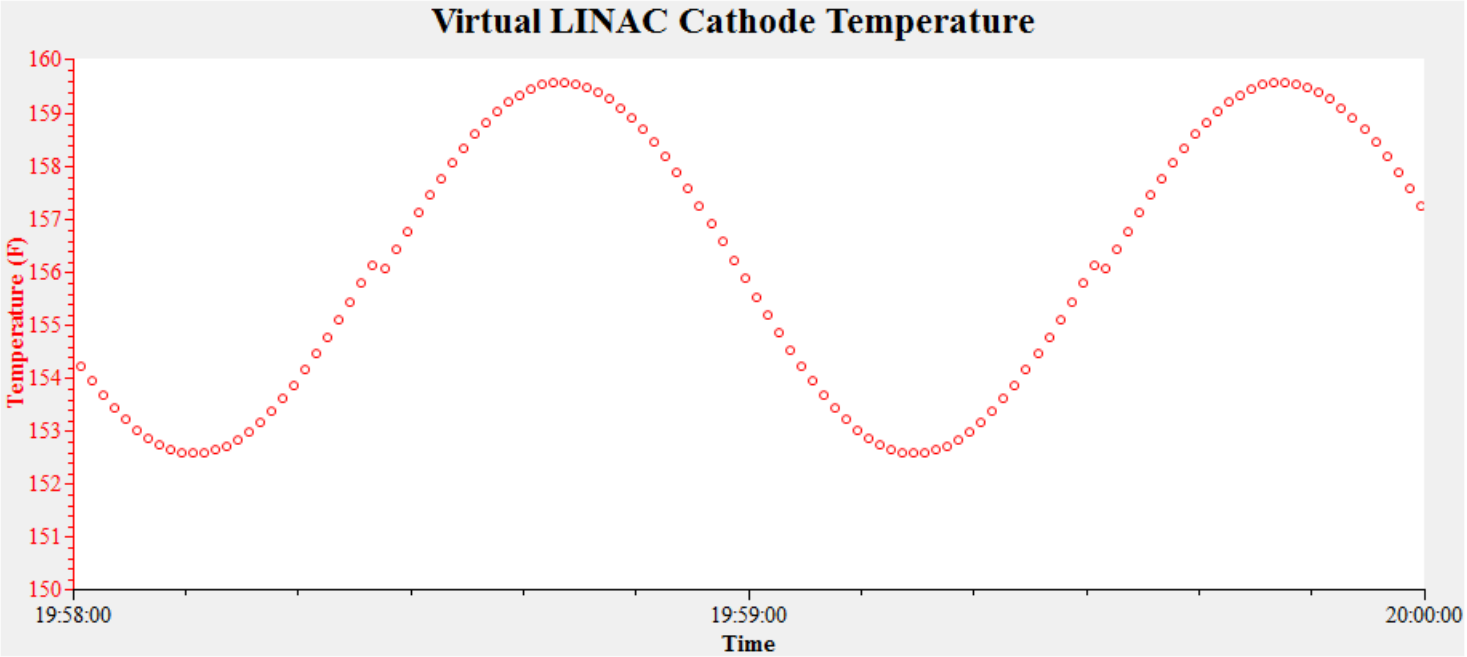


  - 「KBLog履歴データの閲覧方法」参照
    http://www-linac.kek.jp/cont/epics/css/css_databrowser_kblog_usage_public.pdf
  - CSSのヘルプを参照
- 系列や軸の調整は Properties ビューでも可能
  - **Tracesタブ**: PVの追加/削除、色、データソース等
  - **Time Axisタブ**: 横軸の調整
  - **Value Axesタブ**: 縦軸の調整
  - **Misc.タブ**: プロット画面の更新周期と背景色の調整

# 演習 1-4: 自分で使ってみる

COSYLAB

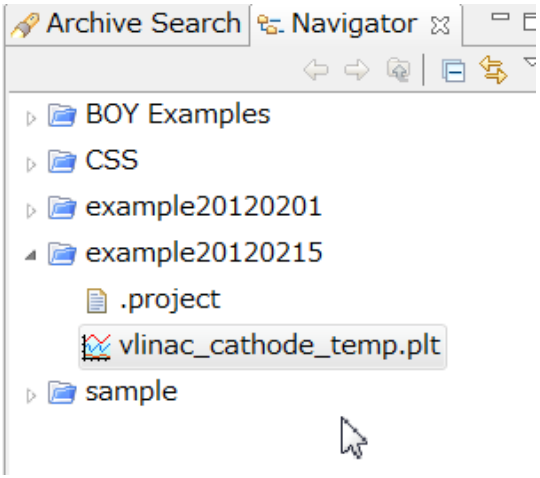
- こんなプロット画面を作り png ファイルとして保存してみてください



# 演習 1-5:ファイルへの保存

COSYLAB

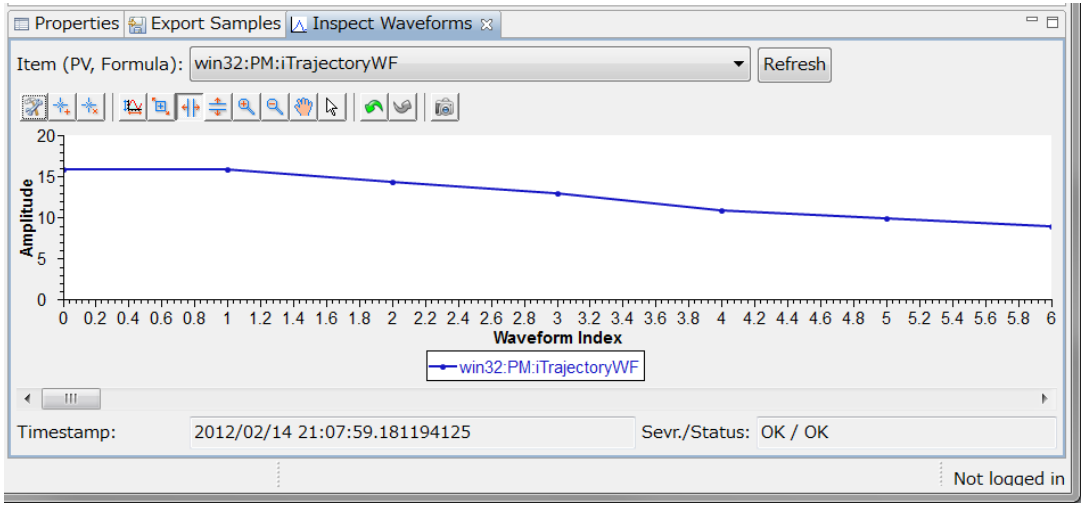
- 1つのプロット画面で1ファイル
  - Ctrl + S か  で保存
  - 拡張子は .plt （自動的に付加してくれます）
- 保存したファイルを再度開く
  - プロット画面を閉じる
  - 既存の.pltファイルを開くときは、Navigatorビューに
  - 先ほど保存したファイルを開きなおす
    - Live Dataは破棄されています



# 演習 1-6: Waveformの確認

COSYLAB

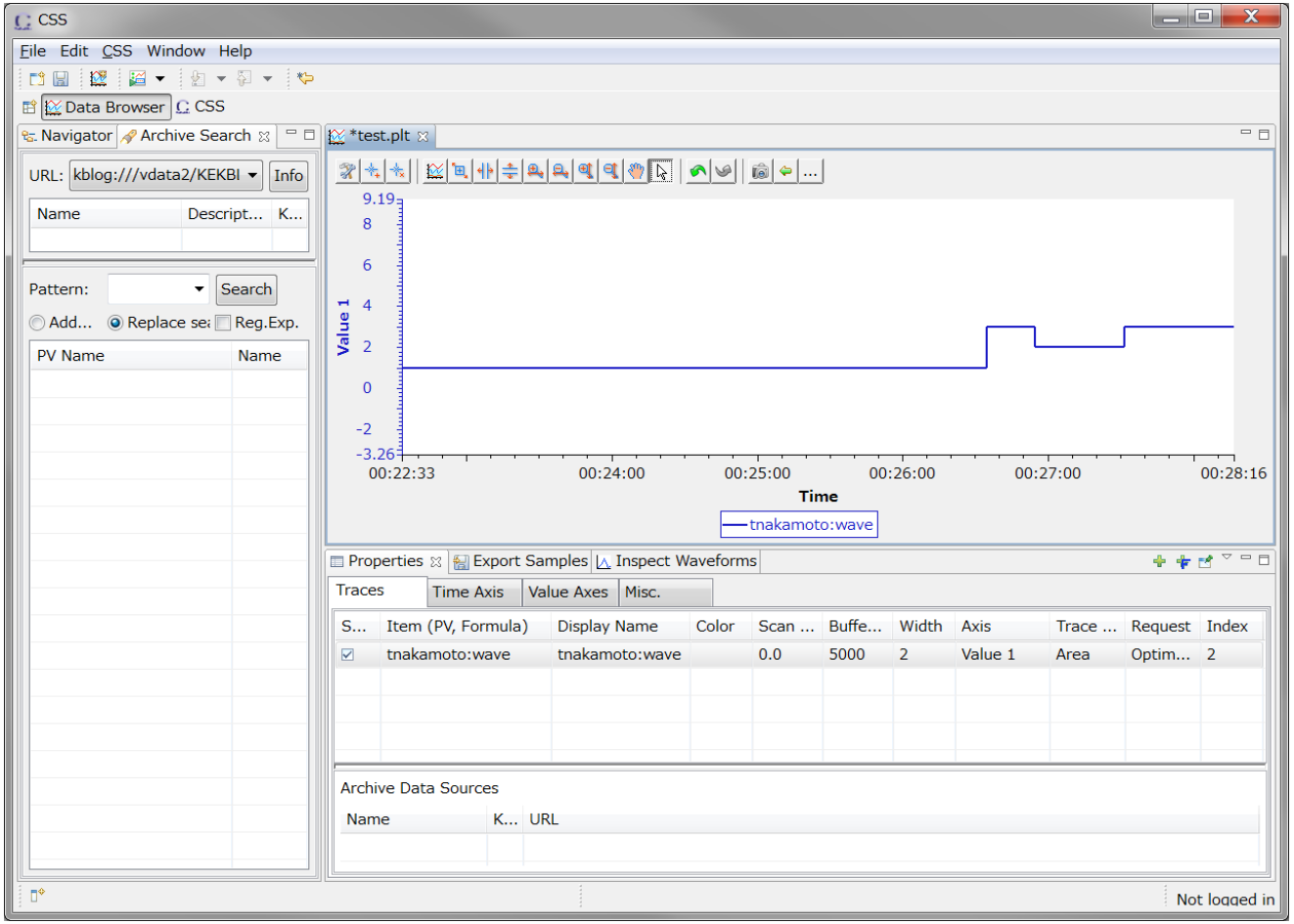
- Inspect Waveformsビュー
  - プロット画面に *ET_kekb:PM:iTrajectoryWF* を追加
    - プロット画面にはこの waveform の最初の要素が表示される
  - プロット画面を右クリック → **Inspect Waveforms**
  - Item (PV, Formula) にて 上記のPVを選択
    - 横軸は waveform の要素番号
    - 下のスクロールバーで時刻を調整



# 開発中…

COSYLAB

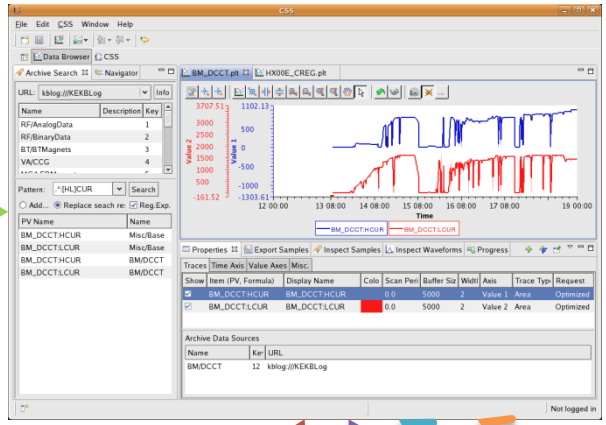
- プロット画面で waveform の2番目以降の要素もチェックできるように （KEK CSS 3.1）



# Data Source

COSYLAB

EPICS Channel Access



Data Sources

Channel Archiver

kblog

PostgreSQL

MySQL

EPICS IOC

EPICS IOC

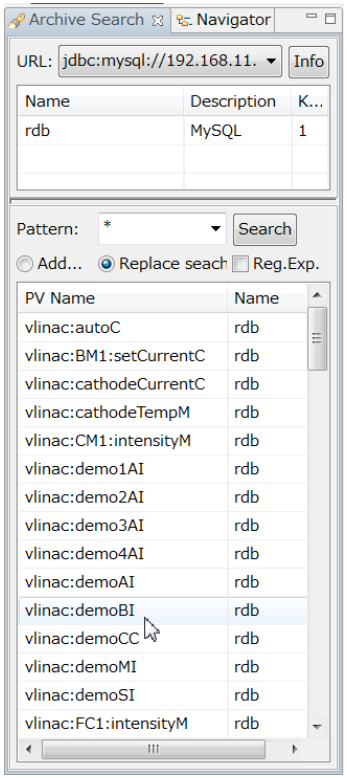
# 演習 1-7: Data Sourceの設定

COSYLAB

- 通常は制御システムの管理者が行います
  - KEK版では、起動スクリプトが自動的に設定
- **Edit → Preferences → CSS Applications → Trends → Data Browser**
  - Archive Data Server URLs:
    - jdbc:postgresql://192.168.0.1:5432/archive
  - Default Archive Data Sources:
    - jdbc:postgresql://192.168.0.1:5432/archive
    - jdbc:postgresql://dummy/
    - ※ key番号は1、2と順に振ってください
  - CSSを再起動
- 必要に応じて各種設定
  - RDBの設定: **CSS Applications → Trends → RDB Archive**
  - kblogの設定: **CSS Applications → Trends → KBLog**

# 演習 1-8: Archive Searchビュー

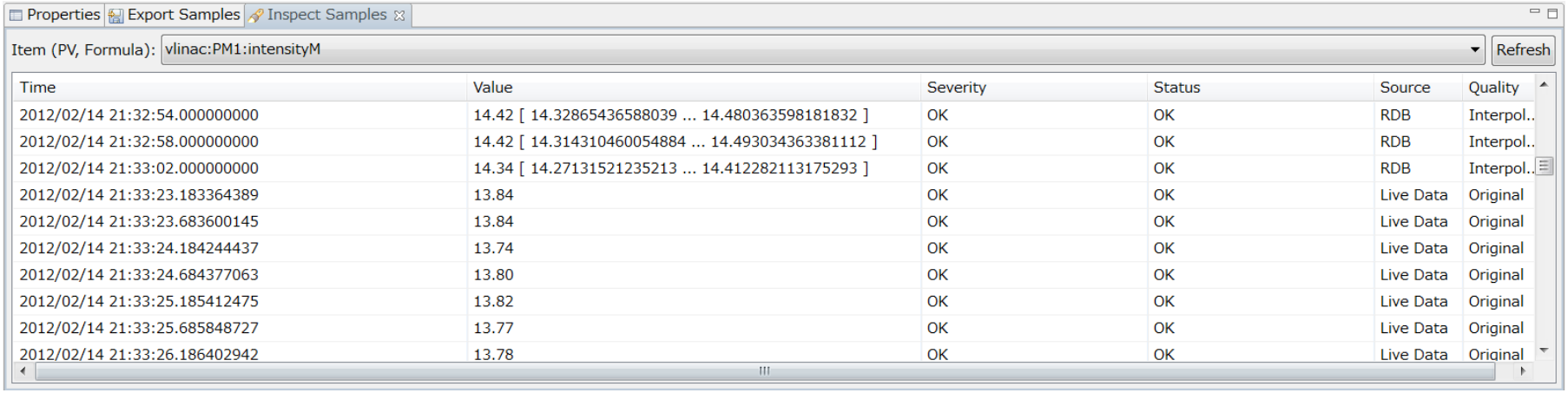
- 特定のアーカイバにあるPVを検索
  - URLで jdbc:postgresql://192.168.0.1:3306/archive が選択されていることを確認
  - Pattern欄に “*” を入力して Search ボタンをクリック
    - Pattern欄には * と？が使えます
    - Reg.Exp. にチェックをすることで正規表現も使えます
- プロット画面への表示
  - PVを選択して、プロット画面にドラッグ
  - 過去のデータと Live Data が同時に表示される



# 演習 1-9: Inspect Samplesビュー

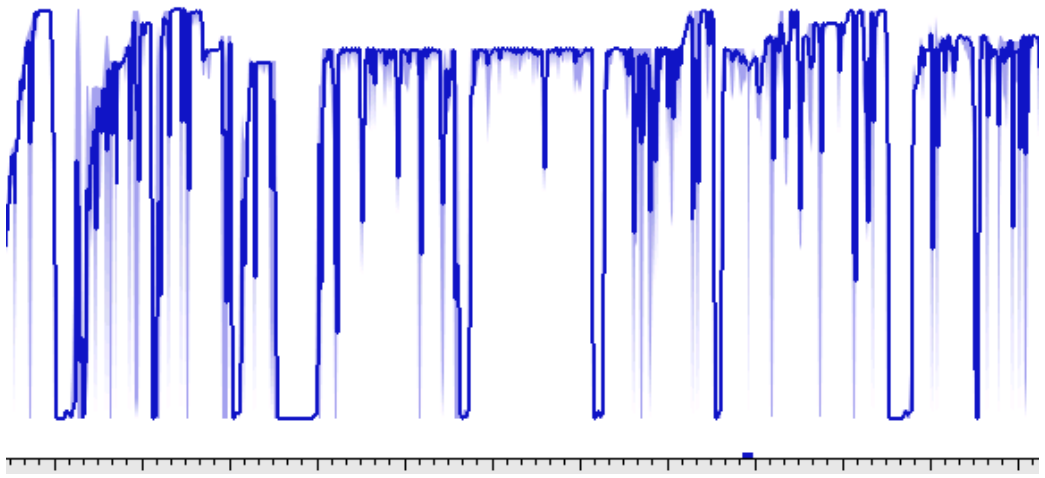
COSYLAB

- Inspect Samplesビュー
  - プロット画面を右クリック Inspect Samples を選択
  - Inspect Samplesビューにて PV 名を選択
    - PostgreSQLから取得したデータとChannel Accessから取得したデータの見分けがつく
  - 最新の Live Data を表示するには Refresh ボタンをクリック



# Qualityについて

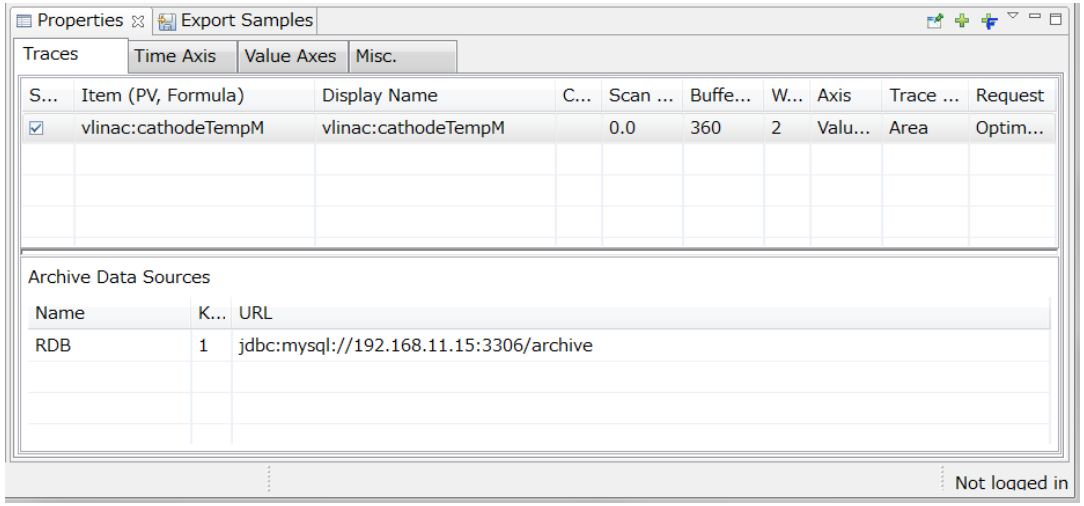
- Quality
  - Original: 生のデータそのまま
  - Interpolated:
    - プロットするのに必要な全てのデータをアーカイバから取得するのは非効率的なため、時間軸をある間隔で区切り、その間隔ごとに最大値、最小値、平均値を取得している。
    - Inspect Samplesビュー上では avg [min … max] という表示
    - プロット画面での表示↓



# 演習 1-10: Default Archive Data Sources

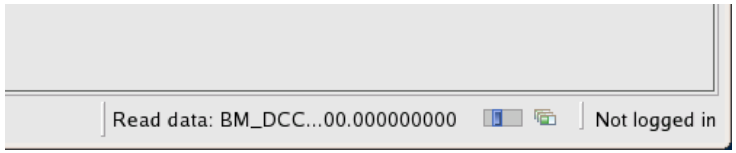
COSYLAB

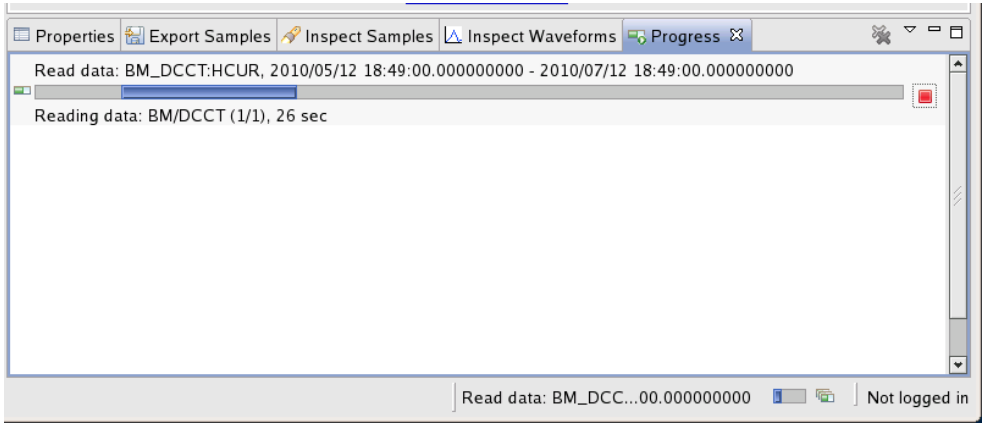
- データソースを指定せずにプロット画面にPVを追加
  - Default Archive Data Sourcesからデータを取得しようとする
    - Dummyの方からは取得に失敗するので、そのエラーが表示される → エラーが表示されないようにするには？
    - 取得に成功した Data Source のみ Archive Data Sources に表示される



# 運用上の注意

- データの取得に長い時間がかかる場合が…
  - 横軸のスパンが長いとき
  - waveformのデータをアーカイバから取得するとき
  - 多数の Data Source から値の取得しようとするとき
- 対処法
  - 上記の状態にならないように気をつける
  - データの取得を中断する
    - 「KBLog履歴データの閲覧方法」の「2.7 履歴データ取得の中断」を参照

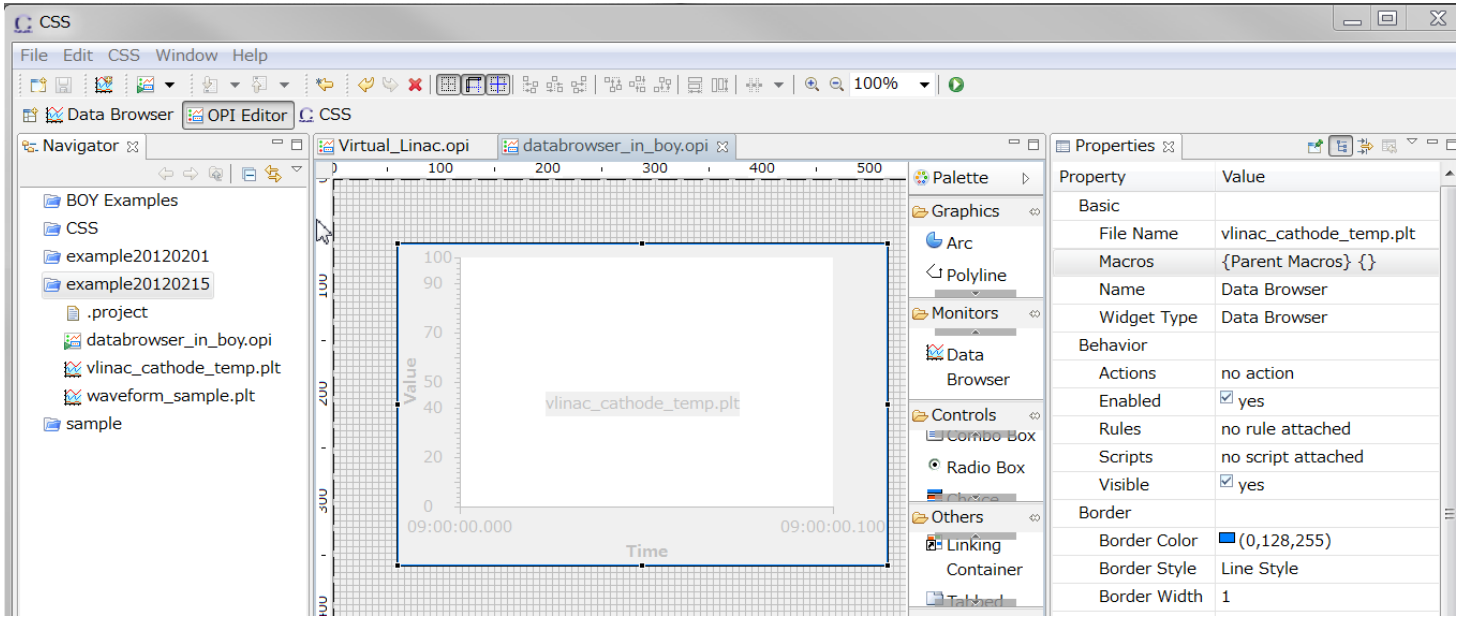


# 演習 1-11: Data Browser in BOY

COSYLAB

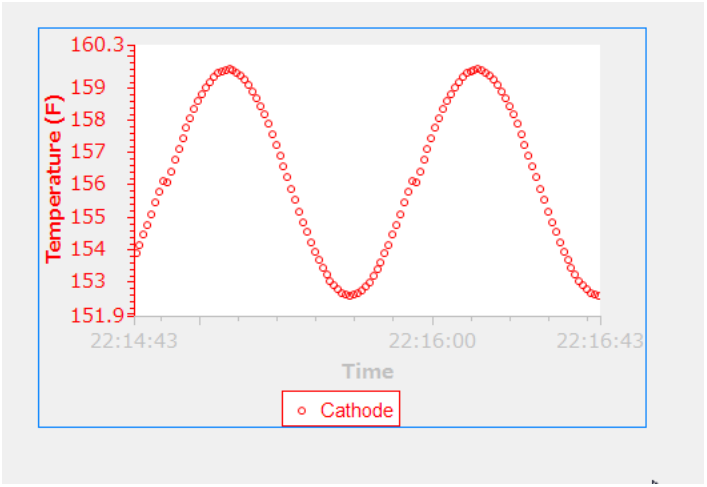
- BOYの画面の中にData Browserのグラフ
  - Data Browserのファイルはあらかじめ作っておく
    - PV、軸、Data Sourceなどの設定は全て済ませておく
  - OPI Editorにて Monitors の中にある Data Browser をドラッグ＆ドロップ
  - File Nameプロパティにて .plt ファイルを指定

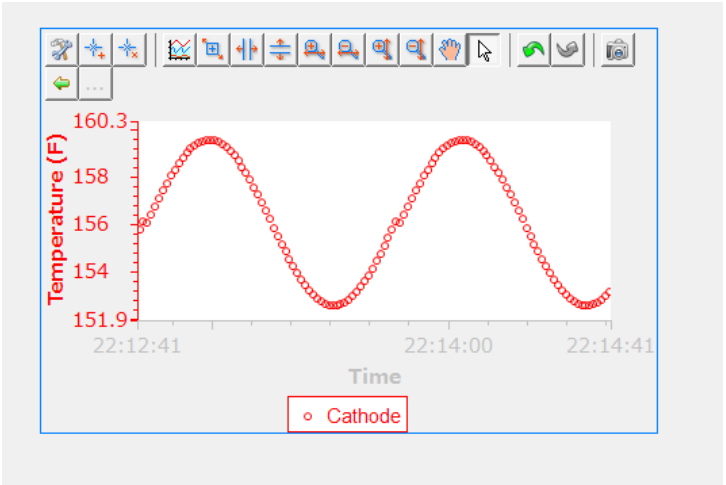


# 演習 1-12: 軸などの調整

COSYLAB

- Toolbarの表示
  - OPI Runtimeで右クリック → **Toolbar** or
  - Data Browserウィジェットの **Show Toolbar** プロパティを yes にする
- あとはData Browserと同じ使い方



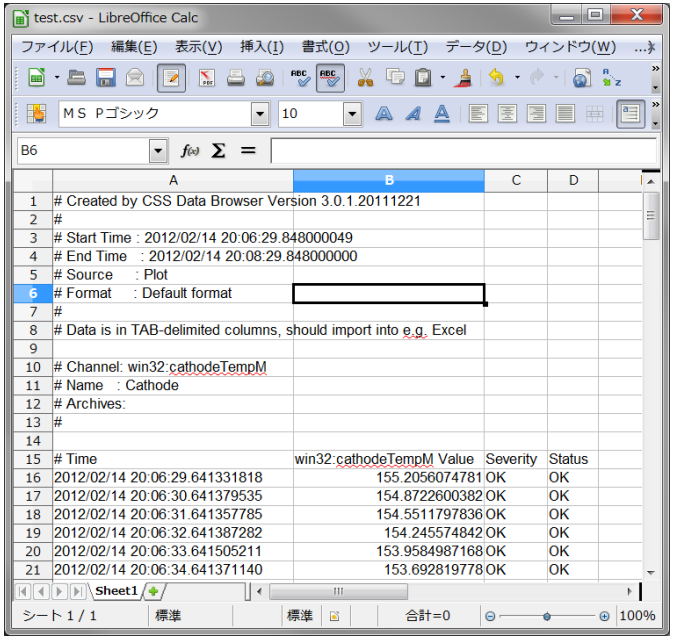



# 他の機能

- Formula
  - いくつかのPVを用いて演算した結果をプロット
- Export Samples
  - プロット画面中のデータ、または、Data Sourceから取得したデータを別ファイルに保存
  - 形式
    - タブ区切り形式
    - Matlab

```
%
% It defines a 'Time Series' object for each channel
% which can be displayed via the 'plot' command.
% Time series can be analyzed further with the Matlab
% Time Series Tools, see Matlab manual.

% Channel: win32:cathodeTempM
% Name   : Cathode
% Archives:
%
clear t;
clear v;
clear q;
t(1)='2012/02/14 20:09:04.665';
v(1)=155.46243257019168;
q(1)=0;
t(2)='2012/02/14 20:09:05.665';
v(2)=155.8091884252828;
q(2)=0;
t(3)='2012/02/14 20:09:06.665';
v(3)=156.15884998683924;
```



初心者向けCSS 講習会2@KEK

# BOYのMACRO

COSYLAB

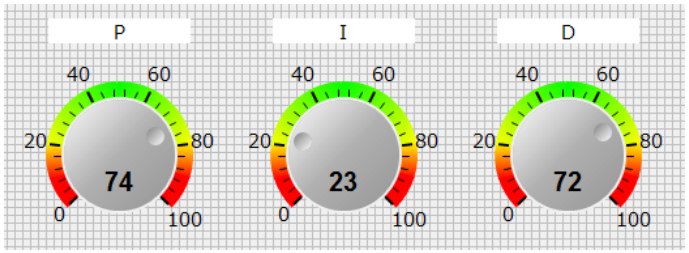
# Macroとは

COSYLAB

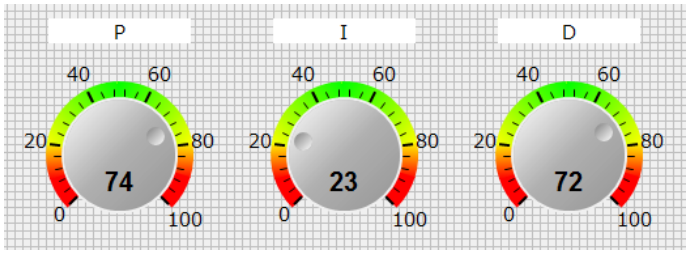
- プロパティ中にある $(xxx) という部分を何か別の文字列で置き換えてくれる機能
  - 何に置き換えるかはCSS、ディスプレイあるいはコンテナごとに指定できる
- Virtual LINACの場合
  - 接頭辞が $(user) となっている
    - $(user):cathodeTempM
    - $(user):cathodeCurrentM
  - 別のユーザーで立ち上げたVirtual LINACを監視するには、$(user)の部分をマクロで置き換える
- 似たようなデバイスの監視・操作画面を作るのに使いまわすのに使う

# Macroの例

- PIDコントローラの例
  - マクロを使わない場合



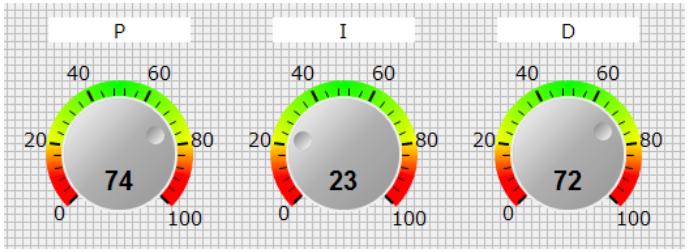

CON1:P　CON1:I　CON1:D




CON2:P　CON2:I　CON2:D

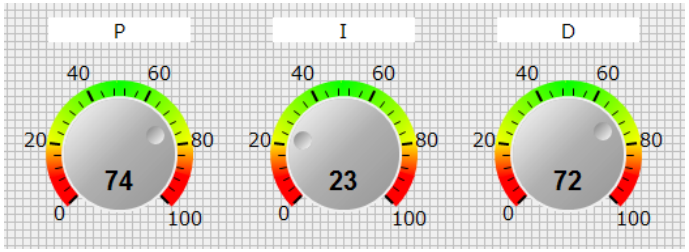
  - マクロを使う場合

$(DEV) = CON1




$(DEV):P　$(DEV):I　$(DEV):D

$(DEV) = CON2

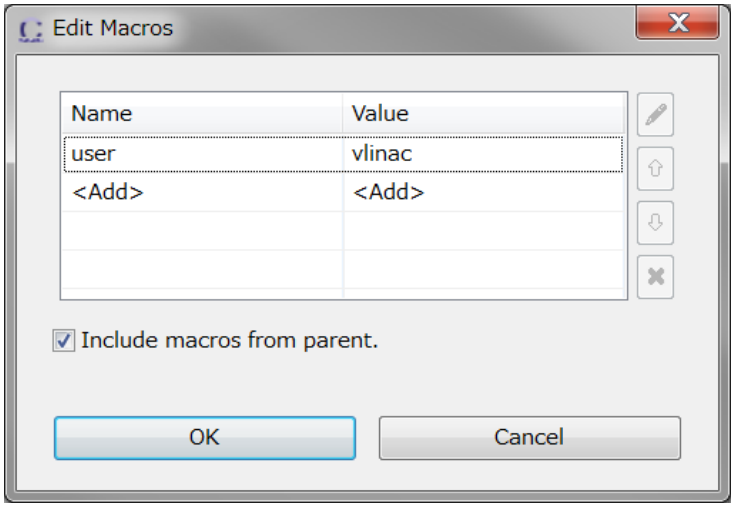



$(DEV):P　$(DEV):I　$(DEV):D

# 演習 2-1: とりあえずマクロ

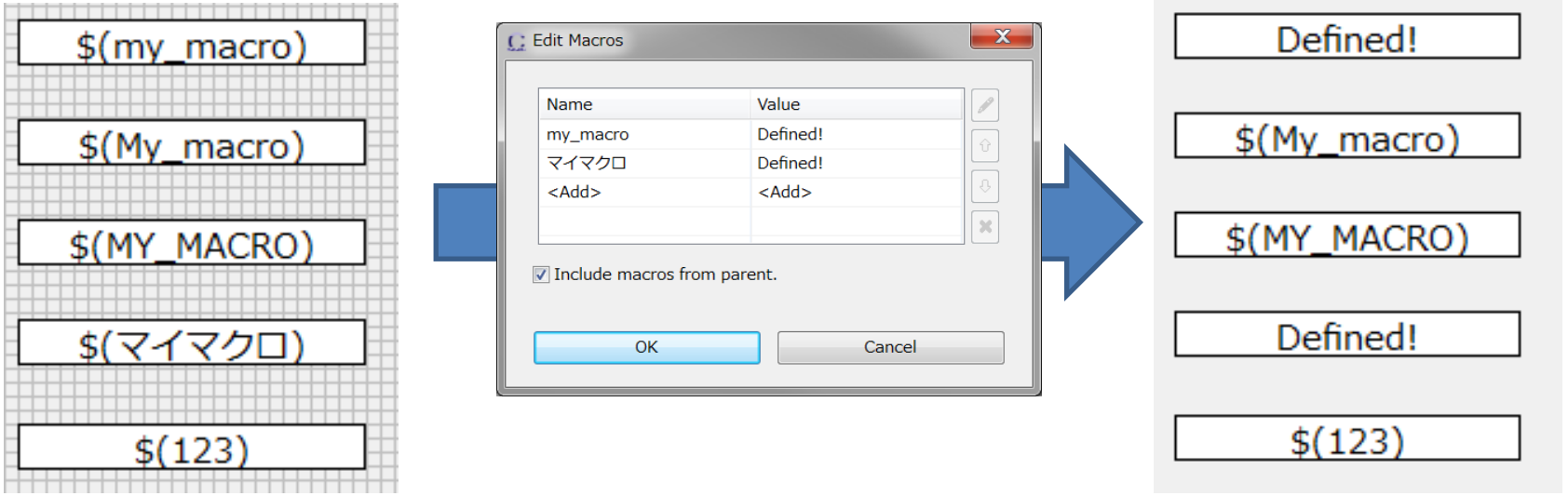
COSYLAB

- 共通の接頭辞を持つ複数のPVの監視
  - 新たにOPIを作成
  - 2つのText Updateを配置
  - それぞれのPV名を
    - $(user):cathodeTempM
    - $(user):cathodeCurrentC
  - ディスプレイの Macro プロパティにて以下のマクロを定義
    - $(user)　=　ET_kekb
  - 実行



# Macro名について

- Case Sensitive
- 日本語も使えるようです (推奨しません)
- 数字から始めることはできません
  - 定義しようとするとエラーが出ます
- 使うときは $() または ${} で囲う



# Macroのレベル

COSYLAB

- レベル
  - **1** CSS全体
    - **Edit → Preferences → CSS Applications → Display → BOY → OPI Runtime**
  - **2** ディスプレイ全体
    - ディスプレイの Macro プロパティ
  - **3** コンテナ内
    - 後述
  - **4..** 入れ子のコンテナ
- 上位からマクロを継承できる
- 上位のマクロは下位で上書き可能

# 演習 2-2: マクロの継承

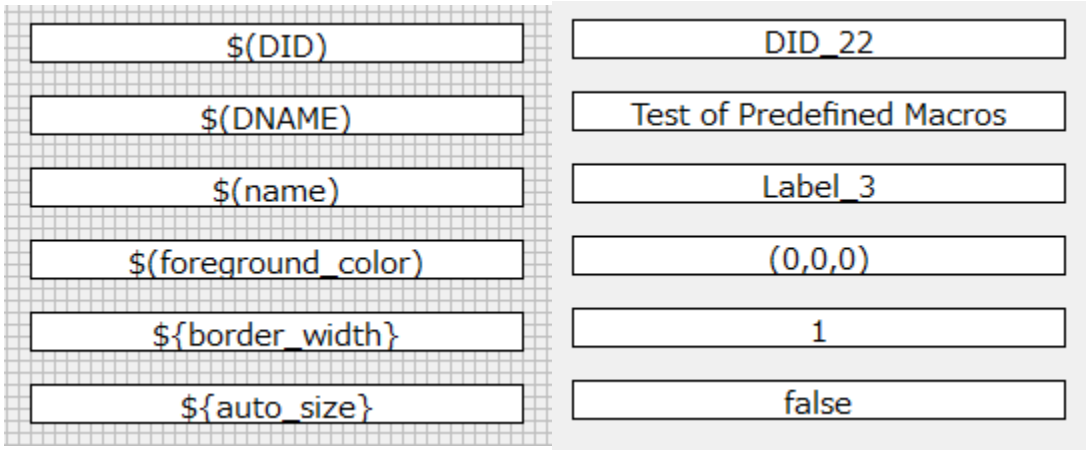
COSYLAB

- マクロが継承されることを確認
  - CSS 全体にて以下のマクロを定義
    $(operator_name) = 自分の名前
  - ディスプレイの Macro プロパティにて
    Include macros from parent.
    にチェックが入っていることを確認
  - Label を配置し、Text プロパティに
    $(operator_name)
    と入力
- Include macros from parentのチェックをはずすとどうなるか？
- ディスプレイのマクロで $(operator_name) を再定義するとどうなるか？

# あらかじめ定義されたマクロ

COSYLAB

- ディスプレイごとのマクロ
  - $(DID): 各ディスプレイに固有のID
  - $(DNAME): ディスプレイの Name プロパティ
- ウィジェットごとのマクロ
  - 各種プロパティ
    - $(pv_name)
    - $(pv_value)
    - $(name)
    - $(foreground_color)
    - $(border_width)
    - $(auto_size)
    - etc…




- Toolbar プロパティはデフォルトで
  **$(pv_name)$(pv_value)**
  となっている

# 演習 2-3: あらかじめ定義されたマクロ

COSYLAB

- あらかじめ定義されたマクロを使ってみる
  - ラベルを4つ配置して、それぞれ下記を Text プロパティに設定
    - $(DID)
    - $(DNAME)
    - $(name)
    - $(border_width)
  - 実行してみる
  - 一度閉じて、再度実行してみる
    - $(DID)の部分はどうなりましたか？

# コンテナ

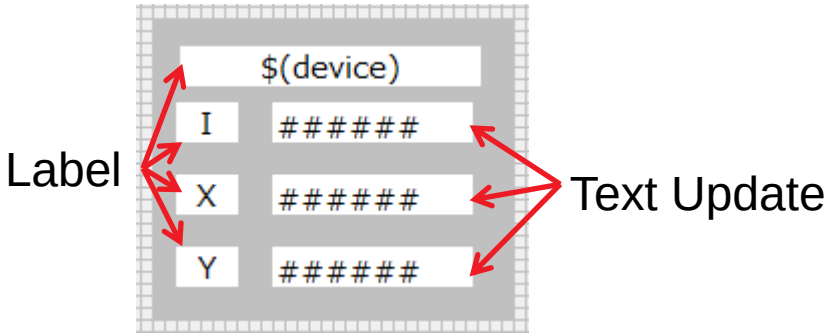
COSYLAB

- コンテナ
  - マクロが定義できる
  - ディスプレイも一種のコンテナ
- Grouping Container
  - 複数のウィジェットを束ねるためのコンテナ
  - 複数のウィジェットを束ねて移動させるときなどに使うと便利
- Linking Container
  - 別のOPIファイルを表示させるためのコンテナ

# 演習 2-4: Grouping Container

COSYLAB

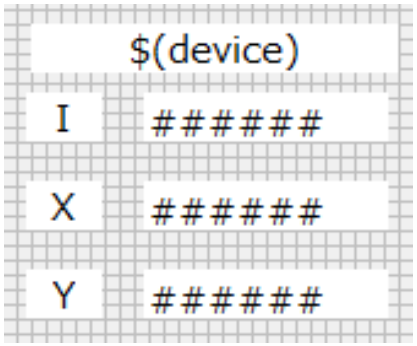
- Grouping Containerを用いて複数点のビーム強度、ビーム位置をモニタ
  - Grouping Containerを配置
  - 右のようにウィジェットをGrouping Containerに配置
  - Text UpdateのPV Name
    - $(user):$(device):intensityM
    - $(user):$(device):X:positionM
    - $(user):$(device):Y:positionM
  - Grouping Containerのマクロ
    - $(device) = PM1
  - ここで一旦実行
  - Grouping Containerごとコピペして、$(device)の定義を PM2、PM3… と変える



# 演習 2-5: OPIの中のOPI

COSYLAB

- OPIの中にOPIを埋め込む
  - 新しいOPIファイルを作成
    - 先ほどと同じく右のようにウィジェットを配置、PV Name を設定
    - child.opi などとい名前で保存
    - テストのため、
      $(user) = ET_kekb
      $(device) = PM1
      と定義して実行
  - もう一つ新しいOPIファイルを作成
    - Linking Containerを配置
    - OPI File プロパティで child.opi を指定
    - ディスプレイで $(user) = ET_kekb を定義
    - 各コンテナで $(device) = PM1，PM2，PM3, … を定義



# コンテナの使い分け

COSYLAB

- Grouping Container
  - 少数のウィジェットをまとめてマクロを使うのには便利
  - マクロ以外の部分で変更がある場合には、全てのコンテナに対し変更の必要あり
- Linked Container
  - 多くのウィジェットをまとめる場合には便利
  - マクロ以外の部分で変更があっても、OPIファイルを一つ直すだけで済む
  - 別ファイルにしなければならない

初心者向けCSS 講習会2@KEK

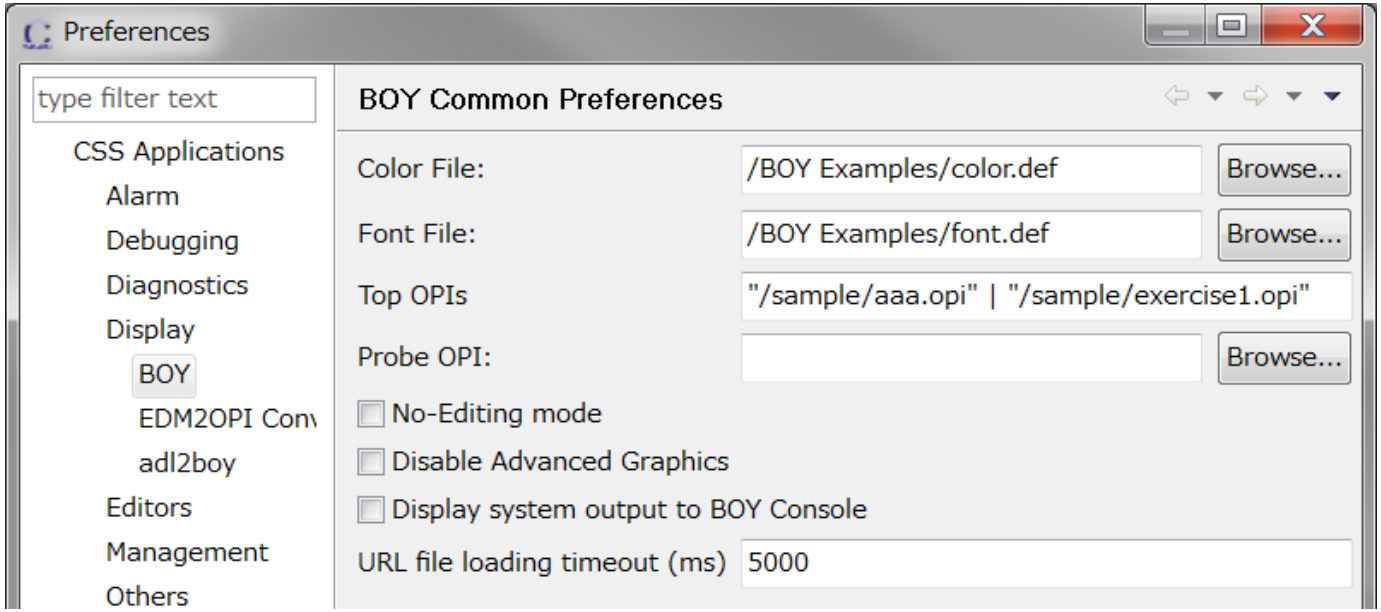
# BOYの設定

COSYLAB

# BOYの設定画面1

COSYLAB

- **Edit → Preferences → CSS Applications → Display → BOY**



# 色とフォント

COSYLAB

- color.def / font.def
  - 知らない間にBOY Examplesのものを使っています
    - デフォルトの設定なので
    - BOY Examplesをインストールしていない場合にはエラーが出ていたかもしれません。
  - 本来は、加速器ごと、グループごとにきちんと定義し Look & Feel を統一するためのものです
  - まずは BOY Examples のものを見てみましょう
    - color.def / font.def を右クリックし **Open With** → **Text Editor** を選択

# color.def

COSYLAB

- BOY Examplesの例

  Red = 255, 0, 0
  Green = 0,255,0
  Blue = 0,0,255
  Yellow = 255,255,0
  Purple = 128,0,255

- 書式
  - テキストファイル
  - color_name = red_value, green_value, blue_value
  - # または //で始まる行はコメント
- あらかじめ定義されているもの (上書きできる)
  - Major, Minor, Invalid, Disconnected

# 演習 3-1: color.defの定義

COSYLAB

- color.defを自分で定義してみる
  - 新しいテキストファイルを作る
    - **右クリック → New → Other → General → Untitled Text File**
  - 下記の内容にする

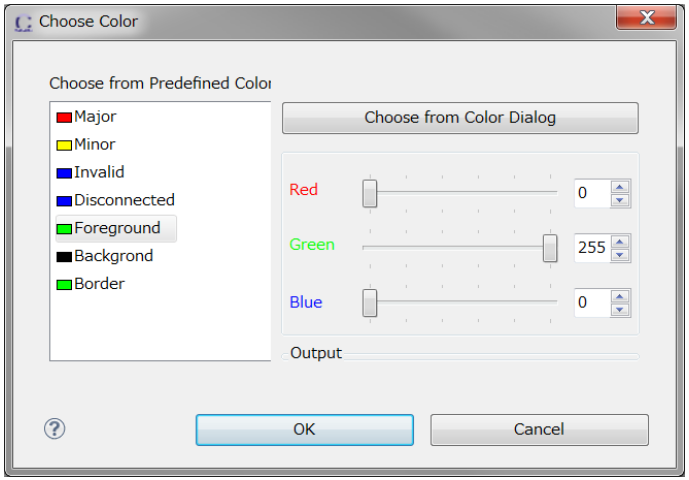
```
Foreground = 0, 255, 0
Backgrond = 0,0,0
Border = 0, 255, 0
Major = 255, 0, 0
Minor = 255, 255, 0
Invalid = 0, 0, 255
Disconnected = 0, 0, 255
```

  - 自分のプロジェクト内に color.def として保存
  - **Edit → Preferences → CSS Applications → Display → BOY** でcolor.defを自分が作ったものに変更
  - CSSを再起動

# 演習 3-2: color.defの使用

COSYLAB

- 自分で定義したcolor.defを使う
  - 新しくOPIファイルを作り、ウィジェットを配置
    - 各ウィジェットの Border Color、Background Color、Foreground Colorを自分で定義したものに変更




  - 実行
    - アラームに対しても、自分が定義したとおりになっているか？
  - color.def をさらに変えて、CSSを再起動して、再度実行

# 色の運用

COSYLAB

- 色をウィジェットごとに RGB値 で指定するのはあまり好ましくない
  - color.def で定義した意味のある色のみを使う
    - それぞれの色に意味を持たせて分かりやすくする
    - 後から色の変更が容易になる
  - color.def にて RED とかいう色を定義してもあまり意味がない

# font.def

COSYLAB

- color.defのフォントバージョン
- BOY Exampleの例
- Defaultというのだけ事前に定義されている
- 書式
  - テキストファイル
  - # または // で始まる行は
  - font_name = name-style-height
  - font_name(OS) = name-style-height
    - Linux: linux_gtk
    - Mac OS X: macosx
    - Windows 7 : windows7_win32

# 演習 3-3: font.defの定義

COSYLAB

- font.defを自分で定義して使ってみる
  - 新しいテキストファイルを作る
    - **右クリック → New → Other → General → Untitled Text File**
  - 下記の内容にする

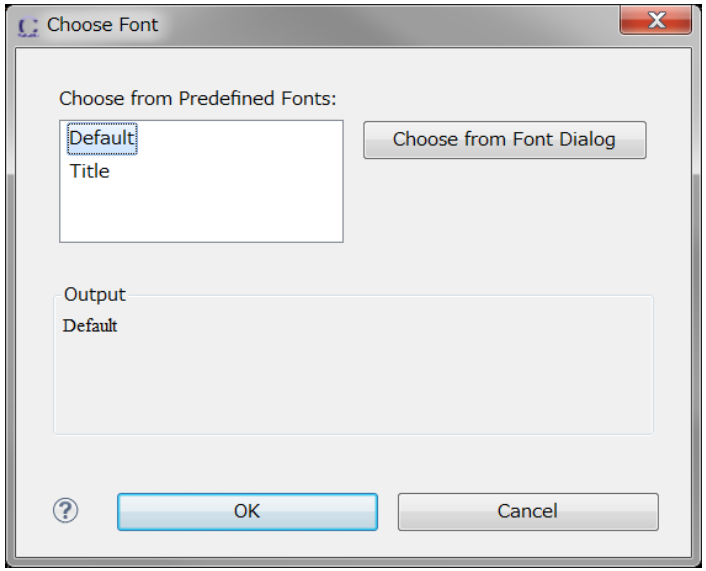
    Default = Times New Roman-regular-9
    Title = Arial-bold-16

  - 自分のプロジェクト内に color.def として保存
  - **Edit → Preferences → CSS Applications → Display → BOY** でfont.defを自分が作ったものに変更
  - CSSを再起動

# 演習 3-4: font.defの使用

COSYLAB

- 自分で定義したfont.defを使う
  - 既存のOPIウィジェットのフォントを確認
    - 全てのウィジェットのフォントが Times New Romanに変わっているはず
  - Label ウィジェットを配置して、フォントを Title に設定



# フォントの運用

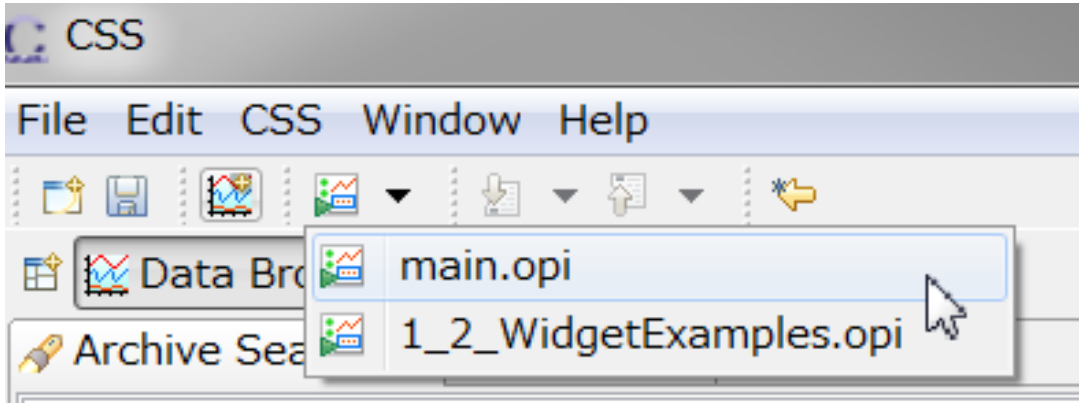
COSYLAB

- フォント名をBOYのウィジェットで指定するのは基本的に良くない
  - OSによってインストールされているフォントが違う
  - font.defで定義したフォントのみを使い、font.defにてOSによる違いを吸収させる
    - せめて同じ幅のフォントを使うように
- フォントの統一
  - 同じフリーフォントを全ての端末にインストールしておくと良いかもしれません
    - 日本語の場合「IPAフォント」など
      http://ossipedia.ipa.go.jp/ipafont/index.html

# 演習 3-5: Top OPIs

COSYLAB

- ツールバーに表示されるOPI
  - 一番上にあるものは、クリックするだけで開く

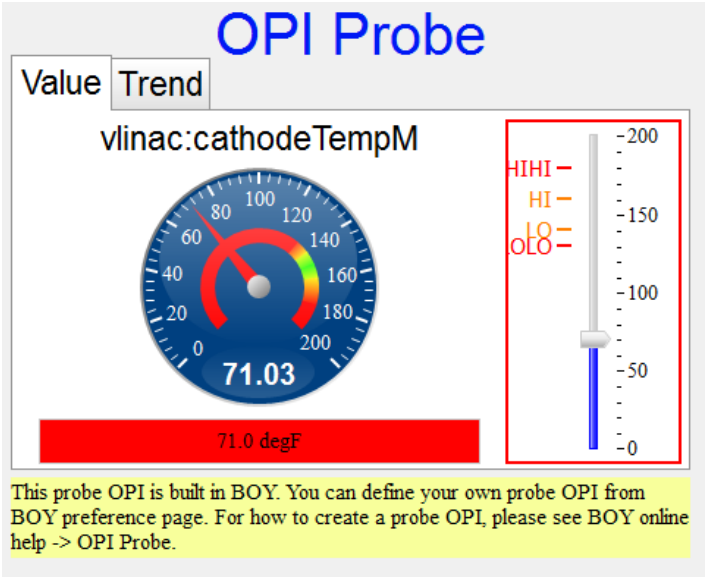



- 書式
  - ワークスペース内でのopiファイルへのパス
    - 例) /sample/aaa.opi
  - 複数のOPIファイルは | で区切る
    - 例)  /sample/aaa.opi | /sample/bbb.opi

# OPI Probe

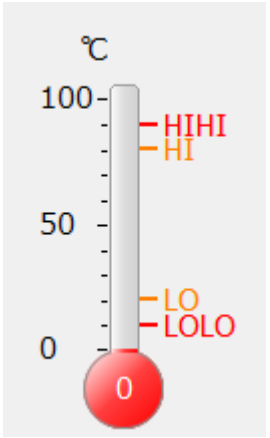
COSYLAB

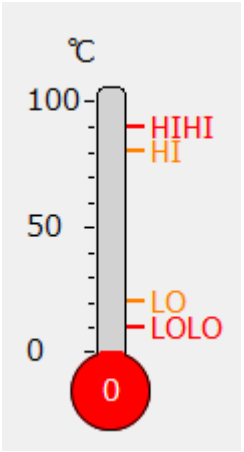
- 自分でProbeを作ることができる
  - 今日はやりません
  - 詳細はCSSのヘルプの **CSS Applications → Display → best OPI Yet (BOY)**を参照
  - 普通のOPIを作る
    - ただし、PV Nameプロパティを $(probe_pv) にしておく
- デフォルトのProbe OPI
  - 何かPV関連のウィジェットなどを右クリックして **Process Variable → OPI Probe** を選択



# その他の設定

- No-Editing mode
  - **Open With → OPI Editor**を選択しても OPI Runtime で開かれるように
- Disable Advanced Graphics
  - 3Dライクな表示をやめる
    - 3D Effectプロパティと同じ
  - BOYのCPU負荷を下げる



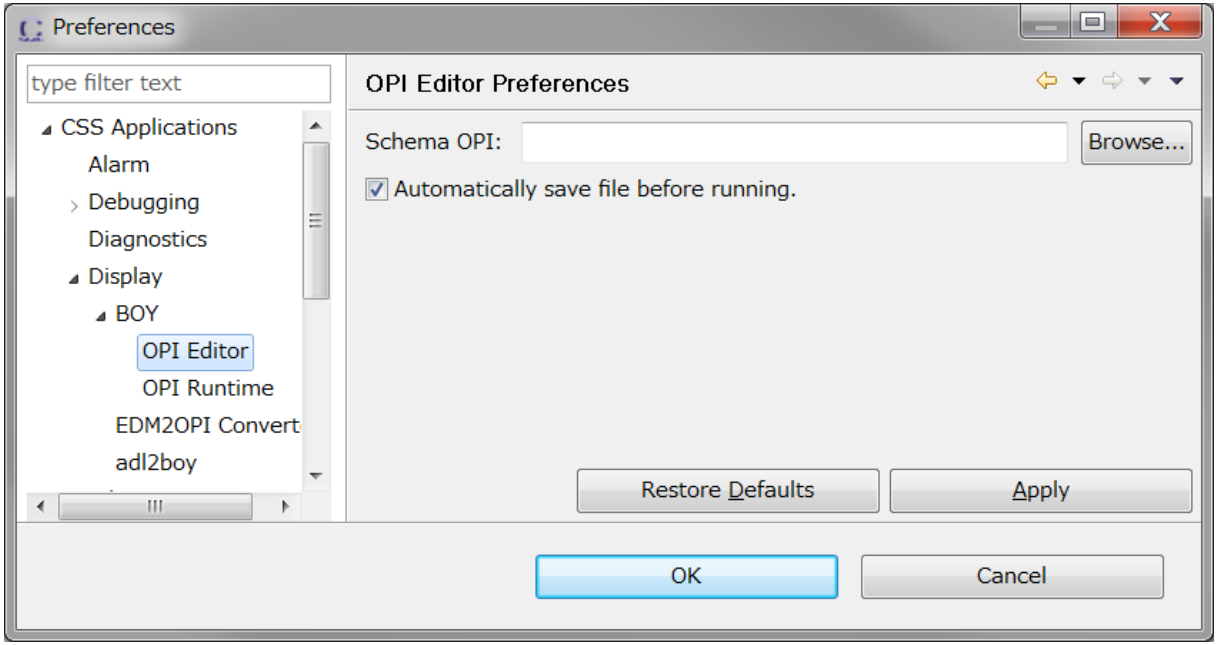




- Display system output to BOY Console
  - スクリプト中での標準出力をBOY Consoleにリダイレクト

# BOYの設定画面2

COSYLAB

- **Edit → Preferences → CSS Applications → Display → BOY → OPI Editor**




  - Automatically save file before running
    - 実はいちいち保存しなくても良かった…
    - 勝手に上書きされることになるので、人によっては外しておく方がよいかもしれません。

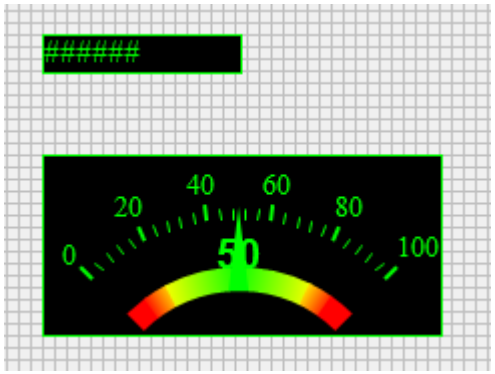
# Schema OPI

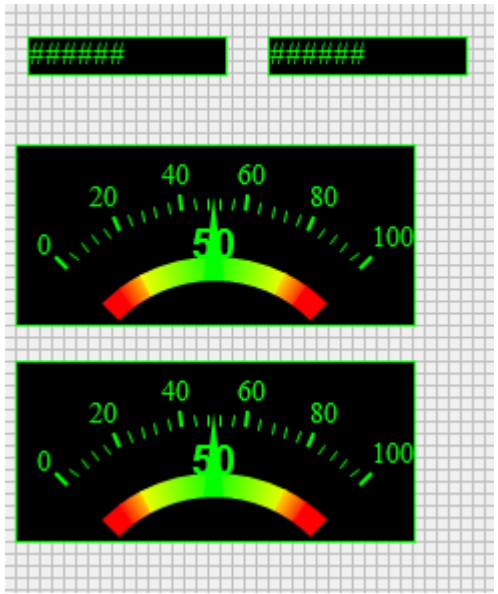
COSYLAB

- 各ウィジェットのプロパティのデフォルト値を決めるOPI
  - 各ウィジェットを最大1つまで配置
    - （全てのウィジェットを配置する必要はない）
  - そのウィジェットのプロパティがデフォルトとして使われる

普通のOPIでウィジェットを追加すると…

Schema OPI



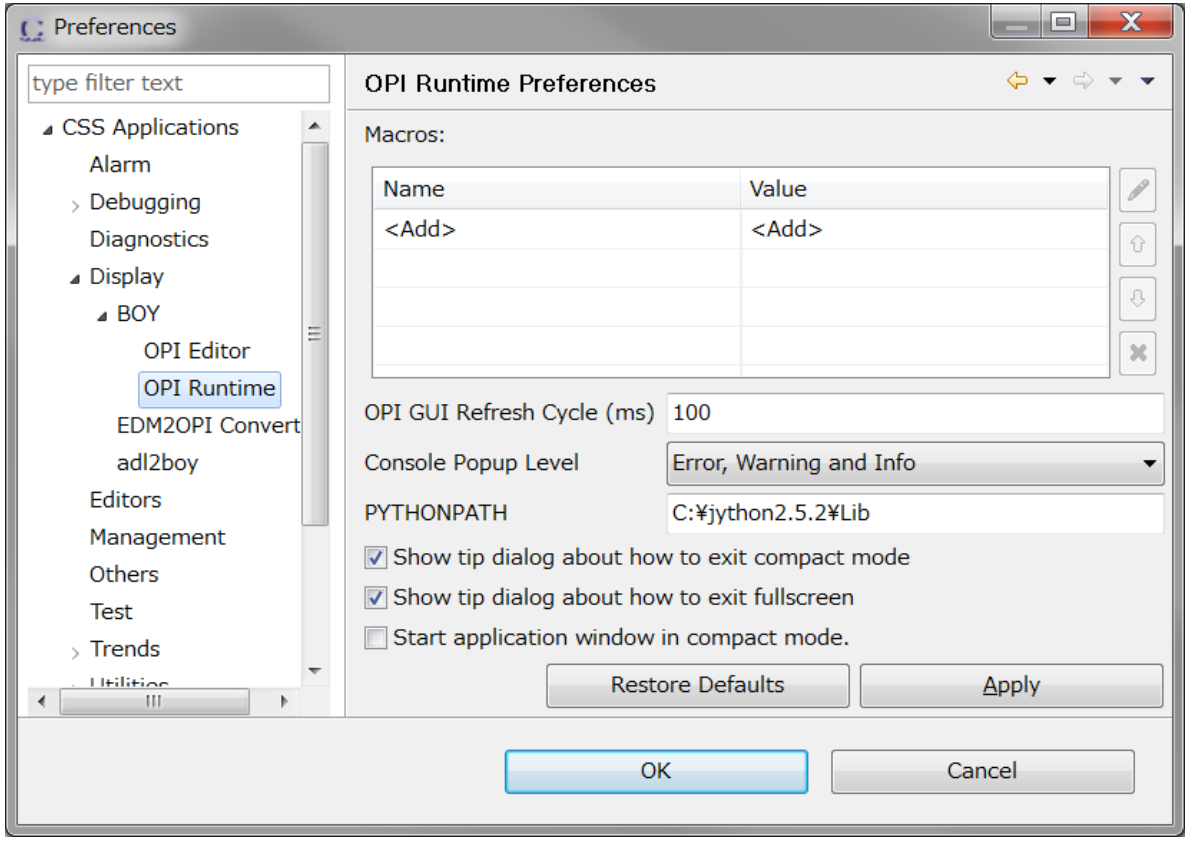
# 演習 3-6: Schema OPI

COSYLAB

- Schema OPIの定義
  - OPIファイルを新規作成
  - Text Updateを配置し、プロパティを変更
    - Border Style を Line Style に
    - Border Color を Border に
    - Foreground Color を Foreground に
    - Background Color を Background に
  - **Edit → Preferences → CSS Applications → Display → BOY → OPI Editor**で作成したOPIを設定
- すると…
  - 別のOPIファイルを新規作成
  - Text Updateを配置してみる

# BOYの設定画面3

- **Edit → Preferences → CSS Applications → Display → BOY → OPI Runtime**



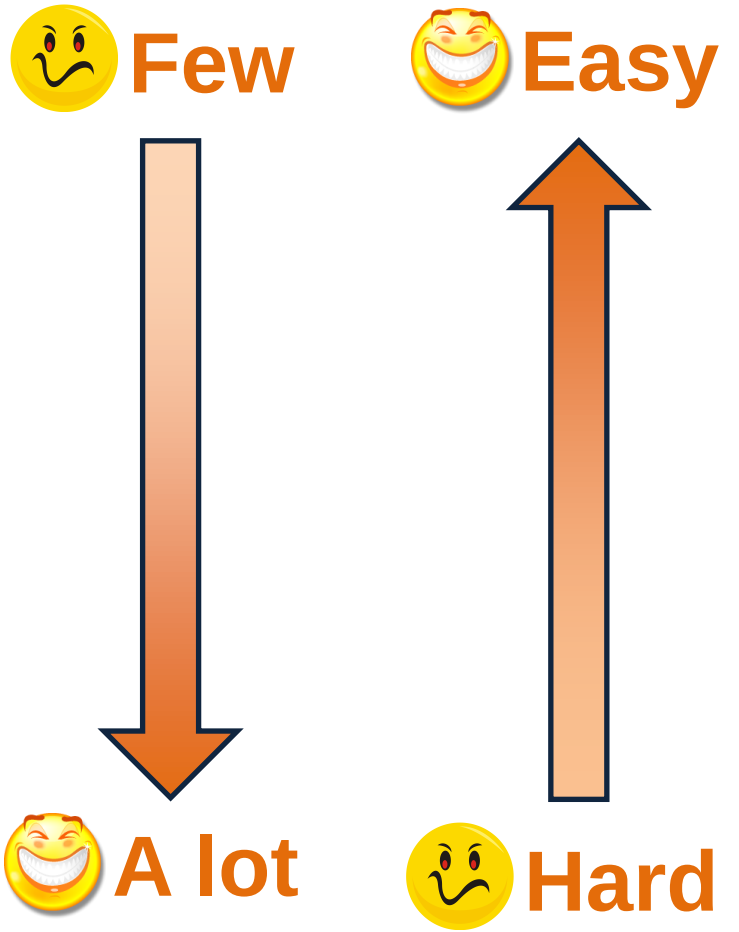
初心者向けCSS 講習会2@KEK

# BOYのRULEを使ってみる

COSYLAB

# BOYの便利な機能

できること　　難易度




- 基本的な編集機能
- Macro
- Action
- Rule
- Script

# 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、ファイルの読み書き
    → **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**

# Rule

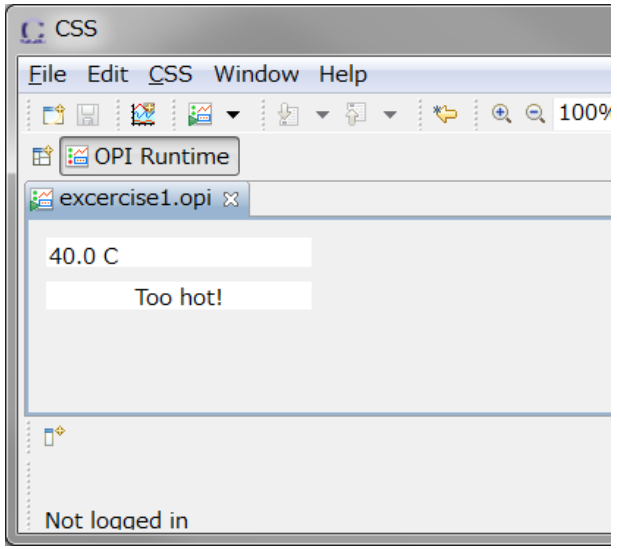
COSYLAB

- 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4-1: 表示・非表示の切替

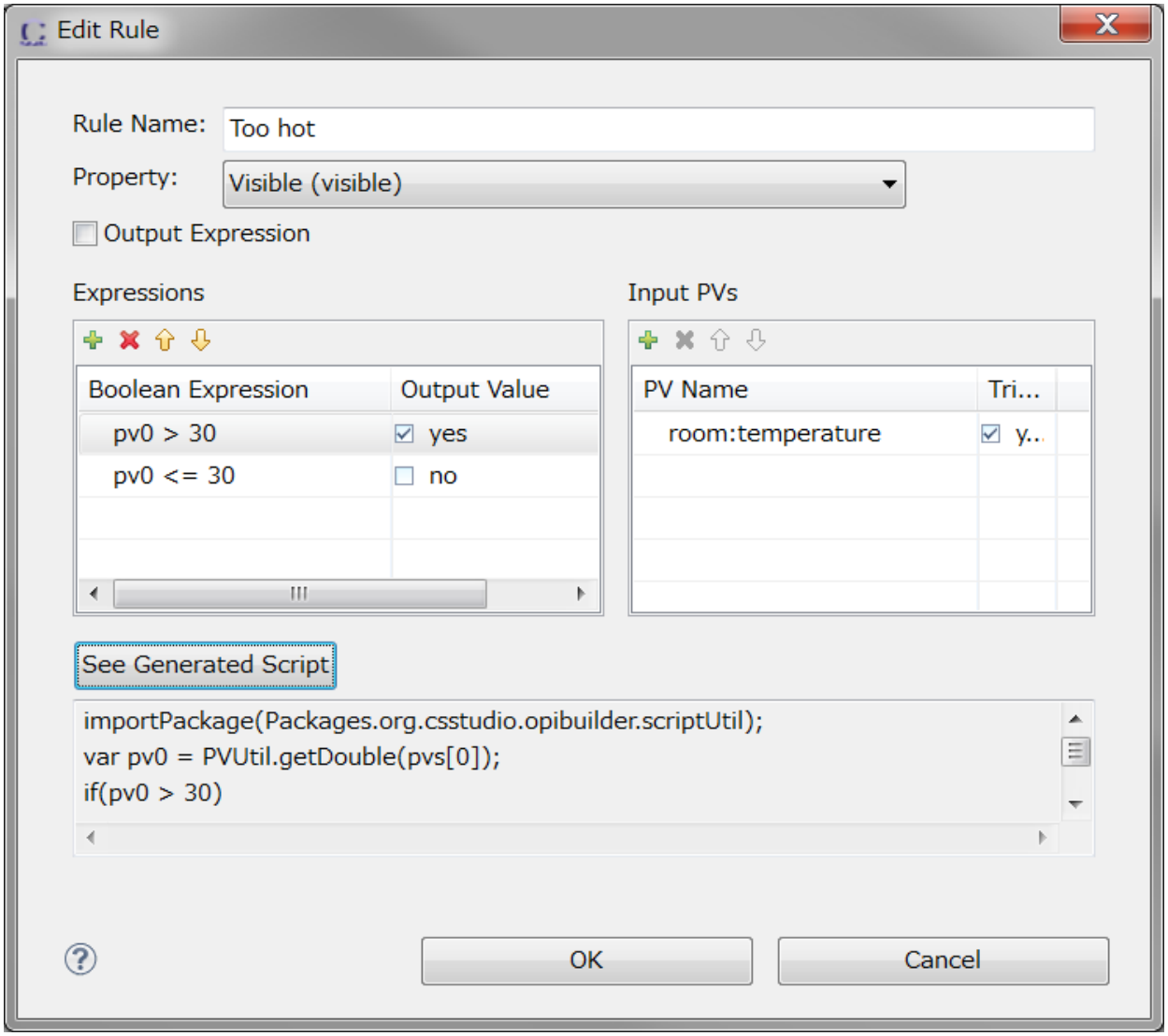
COSYLAB

- 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4-1: 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

初心者向けCSS 講習会2@KEK

# BOYのSCRIPTの初歩

COSYLAB

# 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

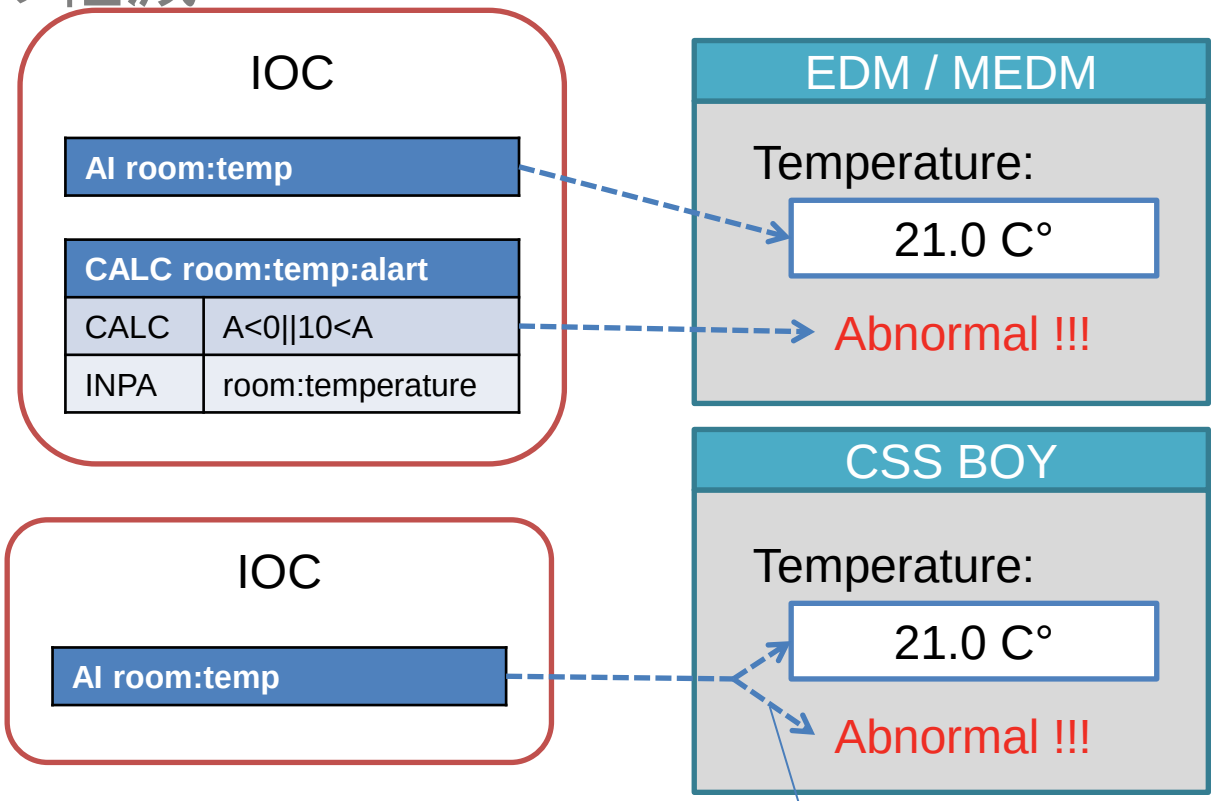
COSYLAB

- 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




**RuleまたはScriptを使う**

# 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
  - 処理系はJython
  - 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)
```

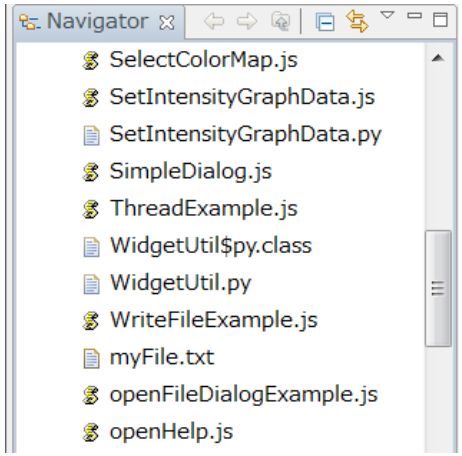
## あとは好みの問題

本日は、少しだけJavaScriptを
あとは、Pythonを使います

# スクリプトファイル

COSYLAB

- 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 5-1: 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

# PVへのアクセス

COSYLAB

- **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5-2: PVの操作

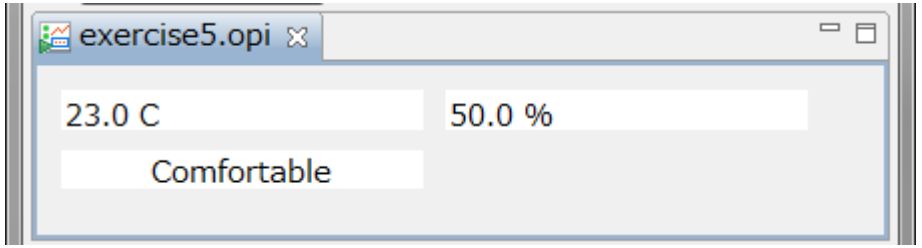
COSYLAB

- 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):
        widget.setPropertyValue("visible", True)
else:
        widget.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

初心者向けCSS 講習会@KEK

# 最後に

COSYLAB

# 今日のまとめ

COSYLAB

- 今日演習で行ったこと
  - Data Browser
  - BOYのMacro
  - BOYの設定 (color.def, font.def, Schema OPIなど)
  - BOYのRule
  - BOYのScriptの第一歩

# 次回 （予定）

COSYLAB

- 次回の内容
  - Scriptを使ってPVにアクセス（read & write）
  - Scriptを使ってファイルの読み書き
  - 他リクエストがあれば
- KEK CSS 3.1を推奨
  - KEK版は2月中にリリースする予定です。
  - PyDevをインストールすることをお勧めします。
    - コード補完・色付きエディタが使えます。
    - インストール方法は、ヘルプの **CSS Applications → Display → Best OPI Yet (BOY) → Script → Python Script**
- 日時は要相談
  - 3月上旬（？）

# 参考文献


- CSSのヘルプ
- K.U. Kasemir, “CONTROL SYSTEM STUDIO (CSS) DATA BROWSER”, Proceedings of PCaPAC08.
- 「KBLog履歴データの閲覧方法」
  http://www-linac.kek.jp/cont/epics/css/css_databrowser_kblog_usage_public.pdf

初心者向けCSS 講習会@KEK

# 質疑応答

COSYLAB