高精細画像公開のためのオープンソースによるプログラム開発

角 祐二郎　　梅田 順一　　原口 尚大

2015 年 3 月

目次

1　目的

図書館で所有する貴重書等の高精細画像を公開するためのプログラムを開発する。本学では、特定のフォーマット、特定のベンダー製品等によって生成したデータを公開しているが、フォーマットの仕様変更や製品の開発終了などに伴って公開を継続できなくなることなどが危惧されていた。これに対し、オープンソースによってデータを生成・公開できるプログラムを開発し、高精細画像の公開を永続的に行えるようにすることが目的である。

1.1　既存システムの参考

- KmView[*1]によるサービス提供 (動作環境 Adobe Flash Player) 、本学の OPAC と連携して御成敗式目などを公開している。[*2]
- 蘆田コレクションでは、Java applet と、これが利用できない環境のために画像縮小によって地図画像等を公開している。[*3]

1.2　既存システムの課題

1.2.1　サービスを利用する側からの課題

閲覧するために必要となる Adobe Flash Player や Java applet などのブラウザ用プラグインは無料で利用することができるが、実際には、OS のセキュリティーアップデートとは別に、 これらのプラグインに対して個別にセキュリティアップデートを定期的に実施しなければならないため、多数の PC を管理しなければいけない場面で管理コストを上げてしまう。これによって間接的にコストのかかるものとなってしまう。

1.2.2　サービス提供側からの課題

閲覧のための仕組みに直接的なコストがかからなくても、提供するコンテンツを増やしたいときにはコストが発生する。増やすコンテンツの規模や用意できる予算の調査を経てから実作業となるが、気軽に行えるものではない。また、定期的な機器のリプレイスを行えなかったりすると、サービスを提供するサーバを移行することができずにサービス継続が危ぶまれることもある。

2　実装について

特定のソフトウェアや、ライセンス等に縛られぬように、プログラムを開発する過程で利用する開発環境、モジュールやライブラリについて、自由に利用できるものかどうかが採用の判断基準となる。例えば、C#であれば Windows の OS 環境と Visual Studio などの開発ツールを使うことが前提となる、Objective-C であれば Mac の OS 環境と Xcode 等の開発環境が前提となる。特定の OS に依存するような環境のハードルは、OS のバージョンアップの際に問題となることが少なくないし、開発ツールが有料ライセンスだった場合、永

[*1] http://www.kmsym.com/kmview/top.htm
[*2] https://opac.lib.meiji.ac.jp/webopac/catdbl.do?pkey=TW00032824&initFlg=_RESULT_SET_NOTBIB
[*3] http://www.lib.meiji.ac.jp/perl/ashida/search_detail?detail_sea_param=atlas,1,10,0

続的にツールを利用できる保証はない。したがって、開発過程で利用する環境、モジュールやライブラリについてはそれぞれの利用範囲・ライセンス等から自由に利用できるものかどうか判断し採用可否を決める。

2.1 開発環境の準備

プログラム開発のためのサーバ環境構築では、従来のサーバ機器の購入によるものではなく、VPS[*4] を採用した。VPS を利用することで、高性能なサーバ環境を安価に導入できるし、物理的なサーバのようにハード自体の保守期限の影響を受けないメリットがある。VPS は仮想サーバの実装の一つであるが、仮想サーバの技術は、古い OS のシステムを最新の物理サーバ上で動かすような目的で利用されることもあり、システムの永続化に貢献できる技術の一つである。

2.2 プログラムの開発

必要になるプログラムは 2 つに分けられ、これらはメタデータによって連携するように設計した。一つは、画像を分割処理するためのプログラムで、もう一方は表示処理を行う。

分割処理では、大きい画像データ（元となる JPEG 画像データ等）を、並び順を保持しながら分割処理する。並び順を保持するデータは軽量なテキストベースの json[*5]フォーマットを用いて、これをメタデータとして扱うこととした。分割処理 (json データの生成を含む) はサーバー側の実装として構築し、状況により、WEB アプリケーションのサーバー側実装として稼働させたり、夜間バッチ処理などでの利用を想定したモジュールとして開発した。開発言語には、オープンソースライセンスで公開されている Python を採用した。[*6] [*7]

- meiji.lib.hriv.utils (参照:別紙 1.1)

表示処理は、特別なプラグイン無しで、ウェブブラウザから利用できるように、JavaScript で実装する。分割処理で生成されたメタデータを読み取り、画像データの縮小率や並び順を保持しながら、画面出力に必要な箇所のレンダリングを行う。JavaScript のモジュールとして有名な jQuery[*8]のプラグインの実装方法を取り入れ、これにならって実装することとした。したがって、最終的な表示システムを実装するのではなく、既存のホームページや OPAC などに配備できる表示用ライブラリとして実装する。JavaScript は、Chrome、FireFox、IE など主要なブラウザ上で動作するスクリプト言語であり、これを使って開発するのに特別な環境を必要としない。

- hriv/hriv.js (参照:別紙 2.1)
- hriv/hrivBase.js (参照:別紙 2.2)
- hriv/hrivImageCell.js (参照:別紙 2.3)
- hriv/hrivImageCellContainer.js (参照:別紙 2.4)
- hriv/hrivImageLayer.js (参照:別紙 2.5)

[*4] Virtual Private Server の略で仮想サーバのこと。

[*5] データ形式の一つ。

[*6] https://docs.python.org/2/license.html

[*7] 本プロジェクトでは、Python2.7 系を利用して開発したが、画像の分割処理と生成するメタデータの仕様を合わせることで任意のプログラム言語で開発できる。

[*8] http://jquery.com/

2.2.1 meiji.lib.hriv.utils の詳細

Python の画像処理ライブラリ pillow[*9]を利用して、画像の分割処理を行い、同時に Python 標準ライブラリを利用して json 形式のメタデータ作成を行う。[*10] 扱える画像データ形式は、pillow で扱える画像形式に従う。基本的な使い方は、 `convert_image` に変換対象となる画像と、出力先のフォルダを指定する。オリジナルのイメージ画像一枚から、分割処理をして複数のイメージデータを生成するので、出力先はフォルダとして指定する。また、一つの画像につき一つのフォルダを指定し、重複しないようにする必要がある。画像縮小処理は、オリジナルサイズの幅・高さを 1/2 ずつ縮小してゆき、size 引数（デフォルト値 512）と比較し、どちらか一方が size 引数よりも小さくなるまで実行する。

2.2.2 hriv/hriv.js の詳細

JavaScript による、表示処理を行うモジュール郡を、hriv[*11]というグローバル変数に全て格納し、hriv をパッケージ名として扱っている。この手法は、1-グローバルのアプローチ [8, p71-77] を参考にしている。また、JavaScript のモジュールとして広く利用されている jQuery のプラグインとして実装している (参照:別紙 2-1 17 行目以降)。これによって、一般的な手法で実行時にオプション引数を受け取ることなどが可能になる。

2.2.3 hriv/hrivBase.js の詳細

`hriv.hrivBase` というクラスとして定義し、`hriv/hriv.js` で指定された表示先の div エレメントに対して、呼び出し時のオプションを受け取りつつインスタンス化する。`init` メソッドは、インスタンス化の時に呼び出され、`meiji.lib.hriv.utils` が生成した `hriv_data.js`(メタデータ) をロードし、`hriv.hrivImageCell` , `hriv.hrivImageCellContainer` をインスタンス化する。また、マウス操作からの指示であるイベントを適切に処理するために、イベントハンドラを実装して、マウスのスクロールや、ドラッグされた時の情報を `hriv.hrivImageCell` , `hriv.hrivImageCellContainer` に伝達し操作する事などを行う。

2.2.4 hriv/hrivImageCell.js の詳細

`hriv.hrivImageCell` というクラスとして定義し、`meiji.lib.hriv.utils` によって生成・分割された画像 1 枚の情報を格納する。拡大・縮小のための `resizeByIndex` メソッドや、表示・非表示のための `show`、`hide` メソッドなどは、主に `hriv.hrivBase` から呼び出されることを想定している。基本的には、デフォルトとなる 512×512 (px) の画像一枚を保持し、`hriv.hrivBase` からの命令に従って、画像の表示・非表示や拡大・縮小処理を行う。

2.2.5 hriv/hrivImageCellContainer.js の詳細

`hriv.hrivImageCellContainer` というクラスとして定義し、`hriv.hrivImageCell` のコレクションを格納する。保持するコレクションを全て並べると、縮小率に応じた画像が完成し 1 階層のレイヤーとしての役割を担う。

[*9] https://github.com/python-pillow/Pillow

[*10] http://docs.python.jp/2.6/library/json.html

[*11] high resolution image viewer の頭文字をとって hriv と命名した。

2.2.6 hriv/hrivImageLayer.js の詳細

hriv.hrivImageLayer というクラスとして定義し、hriv.hrivImageCellContainer のコレクションを格納する。レイヤーとしての役割を担う hriv.hrivImageCellContainer を複数保持し、各縮小率の重なりを管理しつつ拡大や縮小に伴うレイヤー管理・変更処理等を担当する。

2.3 json メタデータ仕様

メタデータは meiji.lib.hriv.utils で生成され、デフォルトのファイル名は hriv_data.js とし固定値としている。トップレベル要素キーは data とし、cell_size の値に、分割処理した際のサイズを示す。縮小処理は、元の画像が十分に小さくなるまで繰り返し実行されるので、元の画像サイズにより、画像縮小率の階層が変わる。元の画像が大きいと十分に小さくなるまでの階層が多くなり、画像が小さいと階層が少なくなる。一つの階層は、2 次元配列に画像ファイル名とサイズを格納する。画像サイズは、512×512 px を基本とするため、画像の右端、下端が異なるサイズになることがある。ファイル名は、サイズとポジションを記しているが、これを表示処理で利用することはなく、任意のファイル名をつけることができる。ポジションは 2 次元配列で保持している。hriv_data.js の 4 行目にあるような{"w":512, "file":"4043x1108_0x0.jpg", "h":512} が hriv.hrivImageCell と対応し、3 行目から 13 行目が hriv.hrivImageCellContainer と対応し最も大きい画像のレイヤーの 1 枚を表す。

Listing 1　sample hriv_data.js

```
{"data":
  [
    [ /* オリジナルのデータサイズで、一番大きいレイヤーとなる */
      [{"w":512, "file":"4043x1108_0x0.jpg", "h":512},
       {"w":512, "file":"4043x1108_512x0.jpg", "h":512},
       {"w":512, "file":"4043x1108_1024x0.jpg", "h":512}, ... ],
      [{"w":512, "file":"4043x1108_0x512.jpg", "h":512},
       {"w":512, "file":"4043x1108_512x512.jpg", "h":512},
       {"w":512, "file":"4043x1108_1024x512.jpg", "h":512}, ... ],
      [{"w":512, "file":"4043x1108_0x1024.jpg", "h":83},
       {"w":512, "file":"4043x1108_512x1024.jpg", "h":83},
       {"w":512, "file":"4043x1108_1024x1024.jpg", "h":83}, ... ]
    ],
    [ /* オリジナルサイズの縦・横それぞれ 1/2 サイズの大きさ */
      [ {"w":512, "file":"2021x554_0x0.jpg", "h":512},
        {"w":512, "file":"2021x554_512x0.jpg", "h":512},
        {"w":512, "file":"2021x554_1024x0.jpg", "h":512}, ... ],
      [ {"w":512, "file":"2021x554_0x512.jpg", "h":41},
        {"w":512, "file":"2021x554_512x512.jpg", "h":41},
        {"w":512, "file":"2021x554_1024x512.jpg", "h":41}, ... ]
    ],
    ... /* 縮小レイヤーの繰り返し */
    [ /* 縮小された画像で、最も小さいレイヤーとなる。*/
      [ {"w":512, "file":"1010x277_0x0.jpg", "h":276},
        {"w":497, "file":"1010x277_512x0.jpg", "h":276} ]
    ]
  ],
 "cell_size" : 512
}
```

3 成果と課題

分割処理で採用した Python はオープンソースのプログラム言語で、利用に際してとても自由である。Windows、Mac、Linux など様々なプラットフォームで動作するため、開発した meiji.lib.hriv.utils もそれぞれの環境で動作し、これを利用するにあたりコストはかからない。したがって、不定期にコンテンツを

増やしたいといった場面でも追加のコストは不要であるし、プラットフォームを選ばないため、処理するサーバーを専用に用意する必要も無い。

表示処理については、一般的なブラウザで動作する JavaScript で実装しているため、Adobe Flash Player や Java applet などのプラグインとは違い、それ専用のセキュリティアップデートは不要で、管理コストを引き上げてしまうようなことも無い。繰り返しになるが、JavaScript を使って開発するのに特別な環境が必要になることもない。

これらの成果について、当初の目的であったオープンソースとして実装できたと考える。ライセンスに縛られるようなツールの利用を避け、多くのオープンソースプロジェクトの成果の上に、画像を分割処理し、最終的に表示できるまでの仕組みを構築できた。そして、このプロジェクトの成果であるソースコードは、OSI 承認ライセンス[*12]である MIT ライセンス[*13]を採用して公開する。

3.1 成果の公開

開発したプログラムは Python の distutils[*14]を利用してアーカイブし `http://www.meiji-library.jp/downloads/hriv/` からダウンロードできるようにしている。パッケージに含めたデモサイトを `http://www.meiji-library.jp/project01` で稼働させ、ツールを利用して処理した、約 7,800 件の蘆田コレクション画像の一部を公開している。また、プログラムを利用するためのドキュメントの整備も `http://www.meiji-library.jp/docs/hriv` で同時に進めているところである。

3.2 今後の課題

ライセンスについて調査する過程で、オープンソースやフリーソフトウェアの概念に対する理解をより深める必要があることがわかった。特に、GNU 一般公衆ライセンス[*15]や、MIT ライセンスは、有名なライセンスであるが、自由の概念が異なる。オープンソースの定義 [10] との関係についても学ばなければいけないことだと感じた。

また、今回の研究でオープンソースプロジェクトにおけるドキュメントの重要性と、一般的な手法を取り入れることの重要性を学んだ。多くのオープンソースプロジェクトでは、ソフトウェアが素晴らしいだけではなく、利用してもらうためのドキュメントも同様に優れている。Write The Docs[5] ではドキュメントの重要性と記述方法のプラクティスが紹介されている。次いで、一般的な手法を取り入れることについては、例えば表示側の実装で jQuery プラグインとして開発したことにより、jQuery を利用したことのある開発者にとって導入が容易なったと思われるし、jQuery プラグインへの理解も深まり、相乗効果があったと感じる。プログラム言語のトレンドもあるが、記述方法や実装方法のプラクティスに対しても継続的に学習し、積み上げてゆくことが重要であろう。

今後は、既存のサービスの置き換えや、コンテンツの拡充のために、実サービスの中に組み込んでいくことと、さらにドキュメントを充実させ、他の機関でも利用してもらえるようにしてゆきたい。また、高精細画像の公開を永続的に行えるようにすることも目的の一つであるので、プログラムのメンテナンスも継続してゆきたい。

[*12] オープンソースの定義に承認されたライセンスであるということ。
[*13] http://opensource.org/licenses/mit-license.php
[*14] http://docs.python.jp/2/library/distutils.html
[*15] https://www.gnu.org/licenses/gpl.html

参考

[1] Secret Labs AB, Lundh Fredrik, and Clark Alex. pillow. https://pillow.readthedocs.org/. Accessed: 2015-03-15.

[2] Raffaele Cecco, 相川愛三. JavaScript グラフィックス：ゲーム・スマートフォン・ウェブで使う最新テクニック. オライリー・ジャパン, オーム社 (発売), 2012.

[3] Python Software Faundation. python. https://www.python.org/. Accessed: 2015-03-15.

[4] Django Software Foundation. django. https://www.djangoproject.com/. Accessed: 2014-04-20.

[5] Eric Holscher and Troy Howard. Write the docs. http://conf.writethedocs.org/. Accessed: 2014-08-15.

[6] Mark Lutz, 夏目大. 初めての Python. オライリー・ジャパン, オーム社 (発売), 2009.

[7] Alex Martelli, Anna Martelli Ravenscroft, David Ascher, 鴨澤眞夫, 當山仁健, 吉田聡, 吉宗貞紀. Python クックブック. オライリー・ジャパン, オーム社 (発売), 2007.

[8] Nicholas C. Zakas, 豊福剛. メンテナブル JavaScript：読みやすく保守しやすい JavaScript コードのための作法. オライリー・ジャパン, オーム社 (発売), 2013.

[9] 平初, 森若和雄, 鶴野龍一郎, まえだこうへい. KVM 徹底入門：Linux カーネル仮想化基盤構築ガイド. 翔泳社, 2010.

[10] 八田真行. オープンソースの定義日本語訳. http://www.opensource.jp/osd/osd-japanese.html. Accessed: 2015-03-17.

別紙

2015 年 3 月

目次

1 Python モジュール

1.1 meiji.lib.hriv.utils

Listing 1 meiji.lib.hriv.utils.py

```
"""␣高精細画像処理システム用モジュール.␣"""

import os,re,sys,json,logging
from PIL import Image

Image.MAX_IMAGE_PIXELS = None

log = logging.getLogger("meiji.lib.hriv.utils")

def convert_image(org_file_name,out_dir_name,size=512,build_index_html=False):

    log.debug( os.path.isfile( org_file_name ) )

    if not os.path.isdir( out_dir_name ) :
        os.makedirs( out_dir_name )
        log.info( "mkdir␣:␣[%s]␣" % out_dir_name )

    img = Image.open( org_file_name )

    (org_image_w ,org_image_h) = img.size

    #: 全てのレイヤーを格納する配列
    layers = []

    # オリジナルイメージファイルから処理を開始
    layer = []
    for offset_y in range( 0 , org_image_h , size ): # Y からループします。

        x_array = []
        for offset_x in range( 0 , org_image_w , size ):
            sizeW = org_image_w - offset_x
            sizeH = org_image_h - offset_y

            if sizeW > size : sizeW = size
            if sizeH > size : sizeH = size

            new_file_name = "{0}x{1}_{2:0>6}x{3:0>6}".format(
                org_image_w , org_image_h , offset_x , offset_y
            )

            log.debug( ( new_file_name ) )
            log.debug( ( [ sizeW , sizeH ] ) )

            box = ( offset_x , offset_y , offset_x + sizeW , offset_y + sizeH )
            new_img = img.crop( box )
            new_img.save( os.path.join( out_dir_name , new_file_name ) , "jpeg")

            x_array.append( dict({ "w" : sizeW , "h" : sizeH , "file" : new_file_name }) )

        layer.append( x_array )

    layers.append( layer )

    # 縮小処理の実行
    while True:

        org_image_w /= 2
        org_image_h /= 2

        if org_image_w < size : break
        if org_image_h < size : break

        image_size = ( org_image_w , org_image_h )
        img = img.resize( image_size )

        layer = []
        for offset_y in range( 0 , org_image_h , size ): # Y からループします。

```

```
69              x_array = []
70              for offset_x in range( 0 , org_image_w , size ):
71                  sizeW = org_image_w - offset_x
72                  sizeH = org_image_h - offset_y
73
74                  if sizeW > size : sizeW = size
75                  if sizeH > size : sizeH = size
76
77                  new_file_name = "{0}x{1}_{2:0>6}x{3:0>6}".format(
78                     org_image_w , org_image_h , offset_x , offset_y
79                  )
80                  log.debug( ( new_file_name ) )
81                  log.debug( ( [ sizeW , sizeH ] ) )
82
83                  box = ( offset_x , offset_y , offset_x + sizeW , offset_y + sizeH )
84                  new_img = img.crop( box )
85                  new_img.save( os.path.join( out_dir_name , new_file_name ) , "jpeg")
86
87                  x_array.append( dict({ "w" : sizeW , "h" : sizeH , "file" : new_file_name }) )
88
89              layer.append( x_array )
90
91          layers.append( layer )
92
93      log.debug("org␣size␣:␣%sx%s" % ( org_image_w ,org_image_h ) )
94      open(os.path.join( out_dir_name , "hriv_data.js" ),"w").write(
95          json.dumps( { "data" : layers , "cell_size" : size , "layer_count" : len( layers ) } )
96      )
97
98      return None
```

2 JavaScript モジュール

2.1 hriv/hriv.js

Listing 2 hriv/hriv.js

```
/**
 *
 * 高精細画像表示モジュール
 *
 * hriv 名前空間を作成し、名前空間内でクラスを宣言する。
 *
 * @module hriv
 * @namespace hriv
 *
 */
var hriv = hriv || {} ;

/**
 * jQuery プラグイン設定
 * @main
 */
jQuery.fn.hriv = function(prm){

    "use strict";

    return this.each( function(){

        hriv.Base( this , jQuery.extend( {

            VERSION             : "0.0.1" ,
            HRIV_JSON_FILE_NAME : "hriv_data.js" ,
            SCALE_RATES         : [1.0, 0.99609375, ... ,0.5078125, 0.50390625] ,
            mode                : "center" , // normal => なにもしない。 center => センタリングする
            images              : [ "../img/2014_04_18_172322_380_/" ] ,
            height              : 0 ,
            width               : 0 ,
            imageLevel          : 1 ,
            imageScaleIndex     : 127 ,
            imageScaleSpeed     : 1 ,
            backgroundColor     : "#000",
            controllerBoxFlag   : true
        } , prm ) );
    } );

};
```

2.2 hriv/hrivBase.js

Listing 3 hriv/hrivBase.js

```
/**
 * 画像を表示・管理し、イベントを操作をするベース div
 *
 * @class Base
 * @constructor Base
 * @param {HTMLDivElement} canvas 画像を表示する div 要素。jQuery プラグインで指定された要素となる。
 * @param {object} prm オプションパラメータ、jQuery プラグインから引き渡されるオプションを取得。
 *
 */

hriv.Base = function( canvas , prm ){

    "use strict";

    // jQuery のプラグインから引き渡された div 要素
    canvas.style.width  =  prm.width  ? prm.width  + 'px' : window.innerWidth   + 'px' ;
    canvas.style.height =  prm.height ? prm.height + 'px' : window.innerHeight  + 'px' ;
    canvas.className   += ' hriv-canvas' ;
    canvas.style.backgroundColor = prm.backgroundColor ;

    var base = document.createElement("div");
    base.className    += ' hriv-base' ;
    base.style.width   =  canvas.style.width  ;
    base.style.height  =  canvas.style.height ;
    var baseTop  = 0 ;
    var baseLeft = 0 ;
    var basePanX = 0 ;
    var basePanY = 0 ;

    var mouseFilterScreen = document.createElement("div");
    var mouseFilterScreenTop  = 0 ;
    var mouseFilterScreenLeft = 0 ;

    var $base = jQuery(base);
    canvas.appendChild( base );

    var mouseClickStartPositionX = 0 ;
    var mouseClickStartPositionY = 0 ;

    var mouseMoveCachePositionX = 0 ;
    var mouseMoveCachePositionY = 0 ;

    var imageLevel      = prm.imageLevel ;
    var imageScaleIndex = prm.imageScaleIndex ; // SCALE_RATES 配列の添え字番号スクロール処理で増減
    var imageScale      = prm.SCALE_RATES[imageScaleIndex]; // 縮小率

    // 表示領域を判断するための値
    var scopeMinX = 0 ; // = -1 * prm.CELL_SIZE ;
    var scopeMinY = 0 ; // = -1 * prm.CELL_SIZE  ;
    var scopeMaxX = 0 ; // = parseInt( canvas.style.width  ) + prm.CELL_SIZE ;
    var scopeMaxY = 0 ; // = parseInt( canvas.style.height ) + prm.CELL_SIZE ;

    var imageLayer = null;

    // ControllerBox の初期化
    var cb = null;

    var that = {

        /**
         *
         * 初期化メソッド
         *
         * @method init
         *
         */
        init : function(){

            // overwrite pram value
            prm.CELL_SIZE = that.json.cell_size ;

```

```
72              // jQuery パラメータからの指定レベルサイズがメタデータのレベル階層外だった場合の補正
73              if( prm.imageLevel > ( that.json.layer_count - 1 ) ){
74                  imageLevel = that.json.layer_count - 1 ;
75              }
76              if( prm.imageLevel < 0 ){ imageLevel = 0 ; }
77
78              // 表示領域を判断するための値
79              // 下記の値の初期化を init メソッド内へ移動。
80              // メタデータで持つセルサイズの値を活かすため、ajax のコールバックで実装する。
81              scopeMinX = -1 * prm.CELL_SIZE ;
82              scopeMinY = -1 * prm.CELL_SIZE  ;
83              scopeMaxX = parseInt( canvas.style.width  ) + prm.CELL_SIZE ;
84              scopeMaxY = parseInt( canvas.style.height ) + prm.CELL_SIZE ;
85
86              imageLayer = hriv.ImageLayer();
87
88              // ControllerBox の初期化
89              cb = hriv.ControllerBox( canvas , prm );
90              if( prm.controllerBoxFlag === true ){ cb.show(); }else{ cb.hide(); }
91
92              console.log("hriv.base.init called");
93
94              /*
95               * 表示用グリッドオブジェクトの初期化
96               */
97              for( var imageLevelNumber = 0 ; imageLevelNumber < that.json.data.length ;
                     imageLevelNumber++ ){
98
99                  var icc = hriv.ImageCellContainer();
100
101                 for( var imageIndexY = 0 ; imageIndexY < that.json.data[ imageLevelNumber ].length ;
                         imageIndexY++ ){
102
103                     for( var imageIndexX = 0 ; imageIndexX < that.json.data[ imageLevelNumber ][
                             imageIndexY ].length ; imageIndexX++ ){
104                     var fileInfo = that.json.data[ imageLevelNumber ][ imageIndexY ][ imageIndexX
                             ] ;
105                     var ic = hriv.ImageCell( imageIndexX, imageIndexY , fileInfo , prm );
106                     base.appendChild( ic.e );
107                     var cellTop  = imageIndexY * ( prm.CELL_SIZE * prm.SCALE_RATES[
                             imageScaleIndex] ) ;
108                     var cellLeft = imageIndexX * ( prm.CELL_SIZE * prm.SCALE_RATES[
                             imageScaleIndex] ) ;
109                     ic.setTop( cellTop );
110                     ic.setLeft( cellLeft );
111                     ic.resizeByIndex( prm.imageScaleIndex );
112                     icc.add( ic );
113                     if( imageIndexY === 0 ){ icc.w += fileInfo.w ; icc.indexCountX += 1 ; }
114                     if( imageIndexX === 0 ){ icc.h += fileInfo.h ; icc.indexCountY += 1 ; }
115                 }
116
117                 }
118
119                 // console.log( icc.len() , icc.w , icc.h , icc.indexCountX , icc.indexCountY  );
120                 imageLayer.set( imageLevelNumber , icc );
121
122             }
123
124             // console.log( imageLayer.get(0).len() );
125
126             imageLayer.setScope( [ scopeMinX , scopeMaxX , scopeMinY , scopeMaxY ] );
127             imageLayer.setCurrentLayer( imageLevel );
128
129             if( prm.mode === "center" ){
130               var icc = imageLayer.get( imageLevel );
131               var differenceW = ( parseInt(canvas.style.width ) / 2 >> 0 ) - ( icc.w * imageScale / 2
                         >> 0 ) ;
132               var differenceH = ( parseInt(canvas.style.height) / 2 >> 0 ) - ( icc.h * imageScale / 2
                         >> 0 );
133               that.move( differenceW , differenceH );
134             }
135
136             imageLayer.show();
137
138             /*
139              * 綺麗にマウスイベントを拾うために一枚の div をフィルターとして敷いておく。
140              * z-index で最前面にいる必要があるので最後に追加する。
141              */
```

```
142             mouseFilterScreen.className += " hriv-mouse-filter-screen" ;
143             base.appendChild( mouseFilterScreen );
144 
145             mouseFilterScreen.style.width    = canvas.style.width  ;
146             mouseFilterScreen.style.height   = canvas.style.height ;
147 
148             mouseFilterScreen.addEventListener( "mousedown" , that.mouseDownHandler );
149             mouseFilterScreen.addEventListener( "mouseup"   , that.mouseUpHandler   );
150             mouseFilterScreen.addEventListener( "mouseout"  , that.mouseUpHandler   );
151             mouseFilterScreen.addEventListener( "mousemove" , that.mouseMoveHandler );
152 
153             if( window.MouseScrollEvent ){
154                 mouseFilterScreen.addEventListener( "DOMMouseScroll" , that.mouseScrollHandlerFF );
155             }
156             else{
157                 mouseFilterScreen.addEventListener( "mousewheel" , that.mouseScrollHandler );
158             }
159 
160         } ,
161 
162         flagMove  : 0 ,
163 
164         stopMove  : function(){
165             that.flagMove = 0 ;
166         },
167 
168         startMove : function( clientX , clientY ){
169             that.flagMove = 1  ;
170             mouseClickStartPositionX = clientX ;
171             mouseClickStartPositionY = clientY ;
172 
173             mouseMoveCachePositionX = clientX ;
174             mouseMoveCachePositionY = clientY ;
175         },
176 
177         /**
178          *
179          * 画像移動処理
180          *
181          * @method move
182          * @param {int} mouseCurrentPositionX 移動中マウスの X 座標、あるいは、移動先の X 座標。
183          * @param {int} mouseCurrentPositionY 移動中マウスの Y 座標、あるいは、移動先の Y 座標。
184          *
185          */
186         move : function(mouseCurrentPositionX,mouseCurrentPositionY){
187 
188             var currentCellSize = prm.CELL_SIZE * prm.SCALE_RATES[imageScaleIndex] ;
189 
190             basePanY += mouseCurrentPositionY - mouseMoveCachePositionY ;
191             basePanX += mouseCurrentPositionX - mouseMoveCachePositionX ;
192 
193             var flagPan = false ;
194             if( basePanX < ( -1 * currentCellSize ) ){
195                 flagPan = true ;
196                 basePanX  += currentCellSize ;
197                 scopeMinX += currentCellSize ;
198                 scopeMaxX += currentCellSize ;
199             }
200             else if( basePanX > 0 ){
201                 flagPan = true ;
202                 basePanX  -= currentCellSize ;
203                 scopeMinX -= currentCellSize ;
204                 scopeMaxX -= currentCellSize ;
205             }
206 
207             if( basePanY < ( -1 * currentCellSize ) ){
208                 flagPan = true ;
209                 basePanY  += currentCellSize ;
210                 scopeMinY += currentCellSize ;
211                 scopeMaxY += currentCellSize ;
212             }
213             else if( basePanY > 0 ){
214                 flagPan = true ;
215                 basePanY  -= currentCellSize ;
216                 scopeMinY -= currentCellSize ;
217                 scopeMaxY -= currentCellSize ;
218             }
219 
```

```
220             if( flagPan === true ){
221                 imageLayer.setScope( [ scopeMinX , scopeMaxX , scopeMinY , scopeMaxY ] );
222                 imageLayer.show();
223                 flagPan = false ;
224             }
225 
226             var baseMoveDistanceY = mouseCurrentPositionY - mouseMoveCachePositionY ;
227             var baseMoveDistanceX = mouseCurrentPositionX - mouseMoveCachePositionX ;
228 
229             this.moveBaseAndFilter( baseMoveDistanceX , baseMoveDistanceY );
230 
231             mouseMoveCachePositionX = mouseCurrentPositionX ;
232             mouseMoveCachePositionY = mouseCurrentPositionY ;
233 
234         },
235 
236         /**
237          *
238          * 画像拡大縮小処理
239          *
240          * @method scale
241          * @param {int} i マウスホイールのスクロール値（イベントハンドラによって正規化済み）
242          * @param {int} pointX スクロール操作時のマウスの  X 座標
243          * @param {int} pointY スクロール操作時のマウスの  Y 座標
244          *
245          */
246         scale : function( i , pointX , pointY ){
247 
248             imageScaleIndex += ( i * prm.imageScaleSpeed ) ;
249 
250             try{
251                 if( imageScaleIndex < 0 ){
252                     console.log("expand change level");
253                     imageScaleIndex += prm.SCALE_RATES.length ;
254                     imageLevel -= 1 ;
255                     imageLayer.setCurrentLayer( imageLevel );
256                 }
257                 else if( imageScaleIndex >= prm.SCALE_RATES.length ){
258                     console.log("change level");
259                     imageScaleIndex -= prm.SCALE_RATES.length ;
260                     imageLevel += 1 ;
261                     imageLayer.setCurrentLayer( imageLevel );
262                 }
263 
264             }catch(ex){
265                 console.log( [ ex , "error listen!!! setCurrentLayer error" , imageLevel ] );
266                 if ( imageLevel <= 0 ){
267                     alertこれ以上拡大できません。("");
268                 }
269                 else{
270                     alertこれ以上縮小できません。("");
271                 }
272                 return;
273             }
274 
275             /*
276              * スケール率の配列が 2 ピクセルずつ動く仕様なので、[ * 2 ] を追加するとうまく計算が合う。
277              */
278             var baseMoveDistanceX = ((( pointX - baseLeft ) * i * prm.imageScaleSpeed * 2 ) / ( prm.
                     CELL_SIZE * prm.SCALE_RATES[imageScaleIndex] ) ) >> 0 ;
279             var baseMoveDistanceY = ((( pointY - baseTop  ) * i * prm.imageScaleSpeed * 2 ) / ( prm.
                     CELL_SIZE * prm.SCALE_RATES[imageScaleIndex] ) ) >> 0 ;
280 
281             /*
282              * ベースの移動と逆方向へのフィルタの移動。
283              */
284             this.moveBaseAndFilter( baseMoveDistanceX , baseMoveDistanceY );
285 
286             /*
287              * 再描画処理
288              */
289             imageScale = prm.SCALE_RATES[ imageScaleIndex ];
290             imageLayer.resize( imageScaleIndex );
291             scopeMinX -= baseMoveDistanceX ;
292             scopeMaxX -= baseMoveDistanceX ;
293             scopeMinY -= baseMoveDistanceY ;
294             scopeMaxY -= baseMoveDistanceY ;
295             imageLayer.setScope( [ scopeMinX , scopeMaxX , scopeMinY , scopeMaxY ] );
```

```
296             imageLayer.show();
297
298         },
299
300         /**
301          *
302          * ベースの移動と逆方向へのフィルタの移動。
303          *
304          * @method moveBaseAndFilter
305          * @param {int} baseMoveDistanceX X 方向の移動距離
306          * @param {int} baseMoveDistanceY Y 方向の移動距離
307          *
308          */
309         moveBaseAndFilter : function( baseMoveDistanceX , baseMoveDistanceY ){
310
311             baseTop  += baseMoveDistanceY ;
312             baseLeft += baseMoveDistanceX ;
313
314             mouseFilterScreenTop  -= baseMoveDistanceY ;
315             mouseFilterScreenLeft -= baseMoveDistanceX ;
316
317             base.style.top  = baseTop  + "px" ;
318             base.style.left = baseLeft + "px" ;
319
320             mouseFilterScreen.style.top  = mouseFilterScreenTop  + "px" ;
321             mouseFilterScreen.style.left = mouseFilterScreenLeft + "px" ;
322
323             console.log(baseMoveDistanceX,baseMoveDistanceY,baseTop,baseLeft);
324
325         },
326
327         /**
328          * イメージレベル（レイヤー）のセット処理
329          *
330          * @method setImageLevel
331          * @param {int} i レベル値
332          */
333         setImageLevel : function( i ){
334             // validation : 存在するイメージレベルかを判定する。
335             if( i < 0 ){ throw new Error("too small image level."); }
336             if( i >= that.json.data.length  ){ throw new Error("too bin image level."); }
337
338             imageLevel = i ;
339
340         },
341
342         /**
343          * 現在のイメージレベル（レイヤー）の値を取得
344          *
345          * @method getImageLevel
346          * @return {int} i レベル値
347          */
348         getImageLevel : function(){
349             return imageLevel ;
350         },
351
352
353         /**
354          * 系用マウススクロールハンドラFireFox
355          *
356          * @method getImageLevel
357          * @param {event} ev イベントオブジェクト
358          *
359          */
360         mouseScrollHandlerFF : function( ev ){
361             // console.log( ev.clientX ) ;
362             // that.scaleGrid( ev.detail / 3 , ev.clientX , ev.clientY );
363             ev.preventDefault() ;
364             ev.stopPropagation() ;
365             that.scale( ev.detail / 3 , ev.clientX , ev.clientY );
366         },
367
368         /**
369          * 系以外用マウススクロールハンドラFireFox
370          *
371          * @method getImageLevel
372          * @param {event} ev イベントオブジェクト
373          *
```

```
374             */
375             mouseScrollHandler   : function( ev ){
376                 // console.log( ev.clientX ) ;
377                 // that.scaleGrid( -1 * ( ev.wheelDelta / 120 ) , ev.clientX , ev.clientY );
378                 ev.preventDefault() ;
379                 ev.stopPropagation() ;
380                 that.scale( -1 * ( ev.wheelDelta / 120 ) , ev.clientX , ev.clientY );
381             },
382 
383             mouseUpHandler   : function( ev ){
384                 console.log("handler called " );
385                 that.stopMove();
386             },
387 
388             mouseDownHandler : function( ev ){
389                 console.log("handler called " );
390                 that.startMove( ev.clientX , ev.clientY );
391             },
392 
393             mouseMoveHandler     : function( ev ){
394                 if( that.flagMove ){
395                     that.move( ev.clientX , ev.clientY );
396                 }
397 
398             },
399 
400             end : true
401 
402         } ;
403 
404         jQuery.ajax({
405           url      : prm.images[0] + prm.HRIV_JSON_FILE_NAME ,
406           async    : true ,
407           cache    : true ,
408           dataType : "json" ,
409           error    : function(){ $base.text("url not found"); } ,
410           // json 型メタデータの取得後に、オブジェクト初期化メソッドをコールします。
411           success  : function( res ){ that.json = res; console.log( res ); that.init(); }
412         });
413 
414         return that ;
415 
416 };
```

2.3 hriv/hrivImageCell.js

Listing 4 hriv/hrivImageCell.js

```
/**
 * 裁断した画像を保持するイメージセル
 *
 * @class ImageCell
 * @constructor ImageCell
 * @param {object} fileInfo 画像ファイル名及び、画像の幅と高さの値。
 * @param {object} prm      jQuery プラグインオプションから継承するパラメータオプション。
 *
 */
hriv.ImageCell = function( indexX , indexY , fileInfo , prm ){
    "use strict";

    // console.log( fileInfo.file );

    var imageFileURL  = prm.images[0] + fileInfo.file ;
    var element       = document.createElement("div") ;
    element.className = "hriv-image-cell-hide" ;

    element.style.top    = "0px" ;
    element.style.left   = "0px" ;
    element.style.width  = "256px" ;
    element.style.height = "256px" ;

    var edgeY = fileInfo.h === prm.CELL_SIZE ? false : true ;
    var edgeX = fileInfo.w === prm.CELL_SIZE ? false : true ;

    // console.log(fileInfo.h,fileInfo.w,edgeY,edgeX);

    var that = {

        /**
         * @property {element} e DOM エレメント
         */
        e : element ,

        /**
         * イメージセルの表示
         * @method show
         */
        show : function(){
            element.className = "hriv-image-cell-show" ;
            element.style.backgroundImage = "url(\"" + imageFileURL + "\")" ;
        },

        /**
         * イメージセルの非表示
         * @method hide
         */
        hide : function(){
            element.className = "hriv-image-cell-hide" ;
        },

        /**
         * のを設定CSStop
         * @method setTop
         * @param {int} i ピクセル値
         */
        setTop  : function(i){ element.style.top  = i + "px" ;},

        /**
         * のを設定CSSleft
         * @method setLeft
         * @param {int} i ピクセル値
         */
        setLeft : function(i){ element.style.left = i + "px" ;},

        getTop  : function(){ return parseInt( element.style.top  ); },
        getLeft : function(){ return parseInt( element.style.left ); },

        /**
         *
```

```
72              * イメージセルのリサイズスケール倍率を収めた配列のインデックス値で処理
73              * - 処理負荷が高いので効率化の必要があるかもしれません。
74              *
75              * @method resizeByIndex
76              * @param {int} i インデックス値
77              *
78              */
79             resizeByIndex : function(i){
80                 var cellSize = ( prm.CELL_SIZE * prm.SCALE_RATES[i] ) >> 0 ;
81
82                 element.style.width  = cellSize + "px";
83                 element.style.height = cellSize + "px";
84                 element.style.top    = ( indexY * cellSize ) + "px";
85                 element.style.left   = ( indexX * cellSize ) + "px";
86
87                 if( edgeY === false && edgeX === false ){
88                     // console.log("not edge");
89                     element.style.backgroundSize = cellSize + "px " + cellSize + "px" ;
90                 }
91                 else{
92                     // console.log("edge");
93                     var backgroundImageW = ( fileInfo.w * prm.SCALE_RATES[i] ) >> 0 ;
94                     var backgroundImageH = ( fileInfo.h * prm.SCALE_RATES[i] ) >> 0 ;
95                     element.style.backgroundSize = backgroundImageW + "px " + backgroundImageH + "px" ;
96                 }
97             },
98
99             resizeByPixel : function(i){
100                var scaleRateIndex = prm.CELL_SIZE - i ;
101                that.resizeByIndex( scaleRateIndex );
102            },
103
104            end : true
105
106        };
107
108        return that ;
109
110    };
```

2.4 hriv/hrivImageCellContainer.js

Listing 5 hriv/hrivImageCellContainer.js

```
/**
 * イメージセルのコンテナクラス
 *
 * - 画像の1レイヤーとしての役割
 * - コンテナに属するイメージセルを全てつなぎ合わせると1枚の画像が完成します
 *
 * @class ImageCellContainer
 * @constructor ImageCellContainer
 *
 */
hriv.ImageCellContainer = function(){
    "use strict";

    var imageCells = [] ;
    var w = 0 ;
    var h = 0 ;

    var that = {

        /**
         * コンテナへのイメージセルの追加
         *
         * @method add
         * @param {hriv.ImageCell} ic
         *
         */
        add : function(ic){
            imageCells.push(ic);
        },

        /**
         *
         * コンテナが所持するイメージセルの数を返却
         *
         * @method len
         * @return {int}
         */
        len : function(){
            return imageCells.length ;
        },

        /**
         *
         * 指定されたインデックスのイメージセルを返却
         *
         * @method get
         * @return {hriv.ImageCell}
         *
         */
        get : function(i){
            return imageCells[i]  ;
        },

        /**
         *
         * 一枚の画像としたときの幅
         *
         * @property w
         * @type {int}
         *
         */
        w   : 0 ,

        /**
         *
         * 一枚の画像としたときの高さ
         *
         * @property h
         * @type {int}
         *
         */
```

```
        h   : 0 ,

        /**
         *
         * X 方向へのセル枚数
         *
         * @property indexCountX
         * @type {int}
         *
         */
        indexCountX : 0 ,

        /**
         *
         * Y 方向へのセル枚数
         *
         * @property indexCountY
         * @type {int}
         *
         */
        indexCountY : 0 ,

        end : true

    };

    return that ;

};
```

2.5 hriv/hrivImageLayer.js

Listing 6 hriv/hrivImageLayer.js

```
1
2  /**
3   * レイヤークラス
4   *
5   * - 各拡大率のをレイヤーとして保持します。ImageCellContainer
6   *
7   * @class ImageLayer
8   * @constructor ImageLayer
9   *
10  */
11 hriv.ImageLayer = function(){
12     "use strict";
13
14     var currentLayer = 0 ;
15     var layers = {} ;
16     var layerLength = 0 ;
17     var scope  = [ 0 , 600 , 0 , 600 ] ;
18
19     var that={
20
21         /**
22          *
23          * レイヤーの階層にイメージコンテナをセットする
24          *
25          * @method set
26          * @param {int} imageLevel イメージレベル（レイヤー）
27          * @param {hriv.ImageCellContainer} imageCellContainer イメージセルを格納したイメージコンテナ
28          */
29         set : function( imageLevel , imageCellContainer ){
30             layerLength += 1 ;
31             layers[imageLevel] = imageCellContainer ;
32         },
33
34         /**
35          *
36          * レイヤーの階層からインデックスを指定してイメージコンテナを取得する
37          *
38          * @method get
39          * @param {int} imageLevel イメージレベル（レイヤー）
40          * @return {hriv.ImageCellContainer}
41          */
42         get : function(i){
43             return layers[i] ;
44         },
45
46         /**
47          * レイヤー数を取得する
48          *
49          * @method len
50          * @return {int}
51          */
52         len : function(){
53             return layerLength ;
54         },
55
56         /**
57          * カレントのレイヤーを表示する
58          *
59          * - 同時に別のレイヤーを表示することを想定していないので、他のレイヤーは自動で非表示にする
60          *
61          * @method show
62          *
63          */
64         show : function(){
65             var icc = layers[ currentLayer ] ;
66             for( var ii = 0 ; ii < icc.len() ; ii++ ){
67                 var ic = icc.get(ii)  ;
68                 var t  = ic.getTop();
69                 var l  = ic.getLeft();
70                 if( scope[0] <= l && l <= scope[1] && scope[2] <= t && t <= scope[3] ){
71                     ic.show();
```

```
                }
                else{
                     ic.hide();
                }
            }
        },

        /**
         * 表示領域をセットする
         *
         * @method setScope
         * @param {object} a
         * @example
         *  setScope([0,600,0,600])
         */
        setScope : function( a ){
            scope = a ;
        },

        /**
         * カレントのレイヤーをセットする
         *
         * @method setCurrentLayer
         * @param {int} i レイヤーのインデックス
         *
         */
        setCurrentLayer : function(i){
            // console.log( [ "setCurrentLayer" , layerLength , i ]);
            if( i < 0 || i >= layerLength ){ throw new Error("out of image level"); }
            for( var ii = 0 ; ii < layerLength ; ii++ ){
                if( i === ii ){ currentLayer = i ;  continue; }
                var icc = layers[ii];
                for( var iii = 0 ; iii < icc.len() ; iii++ ){
                   var ic = icc.get(iii)  ;
                   ic.hide();
                }
            }
        },

        /**
         * レイヤーの拡大縮小処理
         *
         * @method resize
         * @param {int} i スケール倍率を収めた配列のインデックス値
         */
        resize : function(i){
            var icc = layers[ currentLayer ] ;
            for( var ii = 0 ; ii < icc.len() ; ii++ ){
               var ic = icc.get(ii)  ;
               ic.resizeByIndex(i);
            }
        },

        end : true

    };

    return that ;

};
```

3　利用方法

3.1　インストール方法

以下の事項を前提としてインストール方法を示します。

- CentOS-6
- Python-2.7
- pip が利用可能であること
- virtualenv が利用可能であること
- pillow が必要とする画像ライブラリが OS にインストールされていること
- 基本的な Linux コマンド操作ができること

任意の作業ディレクトリで、virtualenv 環境を用意し、その環境下で `hriv-x.y.z.tar.gz` パッケージのインストールと基底となる Django プロジェクトの生成をします。

```
$ virtualenv --no-site-packages test_virtualenv
$ cd test_virtualenv/
$ source bin/activate
$ wget http://www.meiji-library.jp/downloads/hriv/hriv-x.y.z.tar.gz
$ pip install hriv-x.y.x.tar.gz
$ django-admin startproject mysite
```

Django の基底となるプロジェクトの `mysite/setting.py` を編集し、hriv アプリケーションを登録する。

```
# -*- coding: utf-8 -*-
# settings.py ファイルの先頭行に追記
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
```

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'meiji.lib.hriv', # ==> この行を追記
)
```

```
# 最終行あたりに以下を追記
MEDIA_URL = '/media/' # アップロードした画像の配信領域
MEDIA_ROOT = os.path.join( BASE_DIR , "media") # アップロードした画像が保存されるディレクトリ
```

Django プロジェクトの “urls.py“ に下記の変更を加える。

```
# 2 行を追記
from django.conf import settings
from django.conf.urls.static import static

# 静的ファイル配信のための設定を追記
urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'mysite.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^hriv/', include( "meiji.lib.hriv.urls" )), # この行を追記
    url(r'^admin/', include(admin.site.urls)),
) + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT) + static(settings.MEDIA_URL,
    document_root=settings.MEDIA_ROOT)
```

Django のデータベース初期化と、開発用 Django サーバーの起動。

```
$ ./manage.py syncdb
$ ./manage.py runserver
```

ブラウザで “http://127.0.0.1:8000/hriv“ へアクセスし、画面が表示されればインストール完了です。利用者登録を行うと、マイページを利用できるようになり、マイページから画像をアップロードすることができます。処理が完了した画像は一覧にある image リンクから表示システムを利用した画像閲覧をすることができます。

3.2 HTML 実装例

表示システムの HTML は下記のように実装されています。

Listing 7 sample.html

```
<!DOCTYPE html>
<html>
  <head>

    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"/>

    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

    <!--[if lt IE 9]>
      <script type="text/javascript" src=" type="text/javascript" src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->

    <!-- ========= -->
    <!-- test mode -->
    <!-- ========= -->
    <link rel="stylesheet" type="text/css" href="/static/hriv/hriv.css"/>
    <script type="text/javascript" charset="utf-8" src="/static/hriv/hriv.js"></script>
    <script type="text/javascript" charset="utf-8" src="/static/hriv/hrivBase.js"></script>
    <script type="text/javascript" charset="utf-8" src="/static/hriv/hrivImageCell.js"></script>
    <script type="text/javascript" charset="utf-8" src="/static/hriv/hrivImageCellContainer.js"></script>
    <script type="text/javascript" charset="utf-8" src="/static/hriv/hrivImageLayer.js"></script>

    <script>
      jQuery(function(){

        jQuery("div.base").hriv({
            width : 0 ,
            height : 0 ,
            imageLevel : 3 ,
            imageScaleSpeed : 8  ,
            controllerBoxFlag   : false,
            images : ["/media/hriv/images/2015/03/16/001-010-00-003-00/"] ,
        });

      });
    </script>

  </head>
  <body style="padding:0px;margin:0px;">
    <div class="base" style="padding:0px;margin:0px;"> </div>
  </body>
</html>
```