HyJavaImages Full Documentation

the java api for highcharts

OVERVIEW

The HyJavaImages library is a pure Java library that allows you to create PNG, JPEG, WEBP and PDF images of your JavaScript charts. There is built-in support for Highcharts, Highcharts Stock, ApexCharts and Vaadin charts JavaScript charting libraries.

There are Java methods to convert:

- Highcharts, Highcharts Stock, Apexcharts and Vaadin charts chart options to a base64 encoded image in PNG, JPEG, WEBP or PDF format
- HTML to a base64 encoded image in PNG, JPEG, WEBP or PDF format
- A URL to a base64 encoded image in PNG, JPEG, WEBP or PDF format

The chart options method above allows for direct integration with HyJavaCharts, HyJavaStock, HyJavaApex and Vaadin charts. You are now able to generate JavaScript charts and create an image, all from within a Java application on the server, with no JavaScript coding!

The library also allows you to create an image of any web page or supplied HTML text. This allows you to create images of charts generated using any JavaScript charting library.

CHROME HEADLESS

The library will start a Chrome headless server/or connect to an existing running Chrome headless server. It then establishes a Web Socket connection to the Chrome headless server and uses the DevTools Protocol to control Chrome.

There are methods to tune Chrome in high volume applications. See the Chrome.setPoolSize() method in the API.

LIBRARY DOCUMENTATION

The best sources of documentation on how to use the libary are:

- This document.
- The HyJavaImages Javadoc. The Javadoc contains a large amount of information on how best to use the library.
- The Demo application included in the download. The demo application contains many examples on how to use the library.

JAVADOC API

The HyJavaImages API can be viewed online or refer to the API jar provided in the product download.

HOW TO USE THE LIBRARY

Ensure Chrome is installed on your system.
Include the HyJavaImages jar in your project.

Example code is shown below.

```
// Startup Chrome headless - (1)
Chrome chrome = Chrome.getInstance();
chrome.start();

// Generate as many chart images as needed - (2)
String base64;
ImageGenerator img = new ImageGenerator();

// Highchart png example - (3)
img.setJavaScript(LibraryType.HIGHCHARTS, "11.2.0", "highcharts.js");
base64 = img.toBase64Image("{your chart options...}");

// Highchart pdf example - (3)
img.setJavaScript(LibraryType.HIGHCHARTS, "11.2.0", "highcharts.js");
base64 = img.toBase64PDF("{your chart options...}");

// Highstock png example - (3)
img.setJavaScript(LibraryType.HIGHCHARTS_STOCK, "11.2.0", "stock/highstock.js");
base64 = img.toBase64Image("{your chart options...}");

// ApexCharts png example - (3)
img.setJavaScript(LibraryType.APEXCHARTS, "3.44.0");
base64 = img.toBase64Image("{your chart options...}");

// HTML text png example
base64 = img.htmlToBase64Image("<html> your html... </html>");

// URL png example
base64 = img.toBase64Image(new URL("https://www.google.com"));

// Close down Chrome headless - (4)
chrome.stop();
```

Code description:

(1) *Get the instance of Chrome and start the Chrome headless server using all defaults.*
In a web servlet application chrome.start() should be called once in Servlet.init().

(2) *The ImageGenerator class provides all methods to generate chart images.*

(3) *The setJavaScript method tells the library which JavaScript libraries to use to render the chart. The 2nd parameter is the library version or null for the latest.*

(4) *Stop the Chrome headless server.*
In a web servlet application chrome.stop() should be called once in Servlet.destroy().

MAIN LIBRARY CLASSES

Chrome Class

The Chrome class is a singleton that provides methods to control the Chrome headless server.

Main functionality:

- Starts and Stops a new instance of a Chrome headless server. See the start() and stop() methods.
- Chrome headless is started with a set of default Chrome arguments. Chrome arguments can be specified using the setChromeArguments() method.
- When starting Chrome headless, the library will search in the standard install locations for the standalone “chrome-headless-shell” binary first, then the “Chrome” binary. If the library cannot find the binary or you wish to use a non-standard install you will need to specify the Chrome binary with the setChromeBinaryPath() method.
  Standard search locations:
  Linux:
  “/opt/hyjavaimages/chrome-headless-shell-linux64/chrome-headless-shell”
  “/opt/google/chrome/chrome”;
  Windows:
  “\hyjavaimages\chrome-headless-shell-win64\chrome-headless-shell.exe”
  “\hyjavaimages\chrome-headless-shell-win32\chrome-headless-shell.exe”
  “\Program Files\Google\Chrome\Application\chrome.exe”
  “\Program Files (x86)\Google\Chrome\Application\chrome.exe”;
  MacOS:
  “/Users/Shared/hyjavaimages/chrome-headless-shell-mac-arm64/chrome-headless-shell”
  “/Users/Shared/hyjavaimages/chrome-headless-shell-mac-x64/chrome-headless-shell”
  “/Applications/Google Chrome.app/Contents/MacOS/Google Chrome”.
- The library can connect/disconnect from an existing running Chrome headless server. See the connect() and disconnect() methods. Chrome headless would need to be started by another process. I.E. not by the HyJavaImages library.
- When starting/connecting with the start() or connect() methods the library will create a pool of Chrome pages ready for re-use.
  The default pool size value of 2 will suit most applications. Large volume applications can set this to a higher value to handle a higher concurrency.
  See the setPoolSize() method.
  When generating images the library will acquire a page from the pool, generate the image, release the page back to the pool. If there are no available pages in the pool the library will wait until one becomes free. In most cases a page is locked for a few hundred milliseconds before being released back to the pool.
- By default the Chrome headless server will listen on a random available port. You can specify the port using the setPort() method.
- By default the image size created will be 800 x 600 pixels. You can specify the image size using the setWindowSize() method.
- The library creates Java temporary files that the Chrome headless server will need **permission** to access.
  The files will be created in the Java system-dependent default temporary-file directory.
  If required this directory can be specified by using the setLibraryTempFileDir() method.
- By default the library and Chrome headless will output messages to System.out. This can be filtered and controlled. See the setQuietMode(), setBrowserLoggingLevel() and setLogFile() methods.
- HyJavaImages can also be run as a HTTP server. To route HyJavaImages requests to a shared HTTP server use the setViaHTTPServerURL() method. Refer to the HTTP Server section below.

ImageGenerator Class

The ImageGenerator class provides methods to generate the required PNG/JPEG/WEBP/PDF images.

Main functionality:

- To generate a chart image Base64 encoded string from Highchart, Highchart stock or Apexcharts chart options see the toBase64Image() methods.
- To generate a chart image Base64 encoded string from html text see the htmlToBase64Image() methods.
- To generate a chart image Base64 encoded string from a URL see the toBase64Image() methods.
- To generate a chart pdf Base64 encoded string see the toBase64PDF() and htmlToBase64PDF() methods.
- One or more sets of global chart options can be specified with the setGlobalOptions() method.
- By default the image created will be 800 x 600 pixel in PNG format. If required use the ImageOptions class to define the image format (PNG/JPEG/WEBP); the JPEG quality; the image scale (resolution); a full page image; the x, y, width and height to determine the rectangular area that the image will be taken from.
- By default the PDF created will be portrait, 8.5 x 11 inch with 0.4 inch margins. If required use the PdfOptions class to define the paper size; the paper width and height, the margins; portrait or landscape; the header/footer; continuous page; page ranges; scale. There are static helper methods cm() and px() to convert cm and pixels to inches.
- The setJavaScript() and setJavaScriptAlternate() methods enable the library to load the required JavaScript to render the chart. Also see the setJavaScriptAdditional() and setJavaScriptStatements() methods.
- If required use the setCSSExternal() and setCSSInternal() methods to enable the library to load any required CSS.
- Each chart is contained within its own div tag. If required use the setChartDivAttributes() method to define the chart div tag attributes. The method can be used to specify values for the style or class attributes of the div. Also see the setChartDivSize() method.
- To emulate a device, media type, locale, timezone or geolocation see the setEmulate...() methods.
- To include multiple charts in the one image see the setAdditionalChartOptions() method. Also the setAdditionalChartDivAttributes() and setChartGroupDivAttributes() methods may be of interest.

IMAGE GENERATION TEMPLATE

The image generated from the toBase64...() methods is based on the following html template.

```
<body>

[[intial html element]] - use the setInitialHtmlElement() method to configure one or more html elements. The default is no html element is defined.

<div id="container-group">
<div id="container" ></div> - will contain the main chart defined with one of the toBase64...() methods.
<div id="container1"></div> - will contain the first additional chart defined with the setAdditionalChartOptions() method (if defined).
<div id="container2"></div> - will contain the second additional chart defined with the setAdditionalChartOptions() method (if defined).
...
</div>

[[final html element]] - use the setFinalHtmlElement() method to configure one or more html elements. The default is no html element is defined.

</body>

OR

<body>

[[Body Html Elements]] - use the setBodyHtmlElements() method to configure the whole set of html elements in the body.

</body>
```

A basic single chart image would be generated from the html below.

```
<body>
<div id="container-group">
<div id="container" ></div> - will contain the main chart defined with one of the toBase64...() methods.
</div>
</body>
```

There are ImageGenerator methods provided if you wish to configure the div tag attributes or css for the chart group div tag and chart div tag(s).

To support more complex images, you can configure the [intial html element] and [final html element] using ImageGenerator methods.

Chart options specified in the toBase64...() methods and the setAdditionalChartOptions() method can also be html elements.
This gives you the option of defining one or html elements within the container-group div tag along with your charts.

For even more complex images, you can configure the whole set of html elements in the body.
Chart div ids must follow the naming convention of id=”container” for main chart and id=”container1″, id=”container2″, … for any additional charts.

For very complex images that do not fit this template, you can define the full required html and use one of the htmlToBase64...() methods.

VAADIN CHARTS SUPPORT

If you are using the library with Vaadin Charts, you will need to include some additional JavaScript as Vaadin Charts modifies the JSON generated.

Include the following method calls:

```
img.setJavaScriptAdditional("https://www.hyjavacharts.com/content/js/vaadin-inflate.js");
img.setJavaScriptChartOptions("__inflateFunctions(chartOptions);");
```

We recommend you download the vaadin-inflate.js JavaScript and store it locally.

RUNNING CHROME AS ROOT

On Linux based systems if you wish to run Chrome as root and are seeing this error: “ERROR: Running as root without –no-sand-box is not supported.”, you will need to include a Chrome Argument before starting Chrome.

```
chrome.setChromeArguments(new ChromeArguments().addArgument("--no-sandbox"));
chrome.start();
```

JAVASCRIPT CHARTING LIBRARIES

To reduce external dependencies and also any intermittent internet delays in loading the JavaScript, we have found that storing the required JavaScript charting libraries locally or within your application will give best results. You can do this be using the ImageGenerator Class setJavaScriptAlternate() method.

LIBRARY PERFORMANCE TESTING

See the HyJavaImages performance testing results.

REQUIREMENTS

Chrome

You must have Chrome 71 or later installed on your system. The library requires a Chrome version that supports the DevTools protocol v1.3. We recommend that you install the most recent Chrome version.

Highcharts

If you wish to to render Highcharts chart images, the HyJavaImages library requires a commercial or free Non-commercial license for the Highcharts JavaScript library.

Highcharts Stock

If you wish to to render Highcharts Stock chart images, the HyJavaImages library requires a commercial or free Non-commercial license for the Highcharts Stock JavaScript library.

Please refer to the Highcharts license page for further details.

CHROME HEADLESS VERSIONS

Chrome Versions 90 and Older

For Chrome versions 90 and older, the Chrome createTarget() method to create pages does not operate correctly. If you are experiencing startup errors and are using Chrome version 90 or older, use the Chrome class setChromeFix(“force-use-json-new;”) method to rectify this Chrome issue.

Chrome Versions 111 and Older

For Chrome versions 111 and older, there is only one version of Chrome headless available. The Chrome argument “–headless” will start Chrome in headless mode.

The HyJavaImages library v1.5.6 and older automatically adds the argument “–headless”.

Chrome Versions 112 and Newer

As of Chrome version 112, the Chrome development team have added a new Chrome headless. There are now “old” and “new” Chrome headless versions available.

The Chrome argument “–headless” will start Chrome in the “old” headless mode. This will change in the future. See Chrome 128 below.
The Chrome argument “–headless=old” will start Chrome in the “old” headless mode.
The Chrome argument “–headless=new” will start Chrome in the “new” headless mode.

For HyJavaImages library v1.6.0 – v1.11.0 the default is to use the “old” Chrome headless. The library automatically adds the argument “–headless=old”.

In preparation for the removal of “old” headless, for HyJavaImages library v1.12.0 and newer the default is to add the argument “–headless”. The headless version used will depend on the Chrome version you are using. The default can be changed by using the ChromeArguments class setChromeHeadlessType() method.

Chrome Versions 120 and Newer

For Chrome versions 120 and newer, the “old” Headless implementation is now available as a standalone chrome-headless-shell binary.

Our recommendation is that you download and install the standalone chrome-headless-shell binary.

For HyJavaImages library v1.9.0 or newer, the library will search for the chrome-headless-shell binary first.

Chrome Versions 128 and Newer

For Chrome versions 128 and newer, the Chrome argument “–headless” will now start Chrome in the “new” headless mode.

JAVA VERSION

The HyJavaImages library requires Java 8 or greater.

HyJavaImages HTTP SERVER

The HyJavaImages HTTP server is a high performance HTTP server that allows all HyJavaImages requests to be routed to a shared HTTP server. The HTTP server will handle all image generation requests and return the base64 image format.

The HTTP server is useful in situations where you have multiple applications generating images using the HyJavaImages library. It allows for efficient use of server resources as you will have only one Chrome headless server and one set of pool pages shared by multiple applications.

To use the HTTP server the only code change required is to replace the Chrome.start() method with the Chrome.setViaHTTPServerURL() method. Your applications would not be required to start a Chrome headless server.

For more information see the HyJavaImages HTTP Server Product page.

SUPPORT

If you have any questions or issues on the HyJavaImages library, please contact our support team via the Contact Us area on the home page.