Developer Documentation

Visual Comparison of Organism-Specific Metabolic Pathways

Katharina Unger (01325652)

February 21, 2019

1 Framework

This project is implemented using a Python Flask app as backend and JavaScript as frontend. In the following paragraphs we describe the implementation of the backend and frontend of the project.

1.1 Backend

In the backend the file app.py creates the flask app, manages the routes and receives the requests. The file util.py contains the converter class, converting requests like 'hsa+eco+sty' into python lists [hsa, eco, sty].
Requests which depend on the KEGG API will be passed to a KEGGQueryManager specified in the package named kegg. The KEGGQueryManager handles requests to the KEGG API from our webapp. It parses our requests into the right format to send them to the KEGG API. However, currently the implementation only handles the requests to get kgml files in the global context, which means in map01100 (Eg. [2]) and requests to get the KO entries of an organism (Eg. [1]). For further extension of the tool, this implementation may be extended for other request types.
The kgml files are parsed into objects which are defined inside the kegg package. entry.py specifies an entry in the kgml file. It contains the underlying data (name, type, reactions, ids,...) of the graphical elements which are specified in graphics.py. Graphical elements are Line, Circle, Rectangle and Roundrectangle, containing rendering information like coordinates, colors, sizes, and others and the corresponding entry. For the class structure in the backend see Figure 1 The object lists are sent to the frontend as JSON strings. Therefore, when an organism is first requested, it is converted into our objects once and is then saved in a JSON file on the server.

1.2 Frontend

The structure of the frontend can be seen in Figure 2.
We use Three.js to render the graph and d3.js for the zooming and panning events.
The JavaScript files in the subfolder \three\ are files from Three.js which are not included

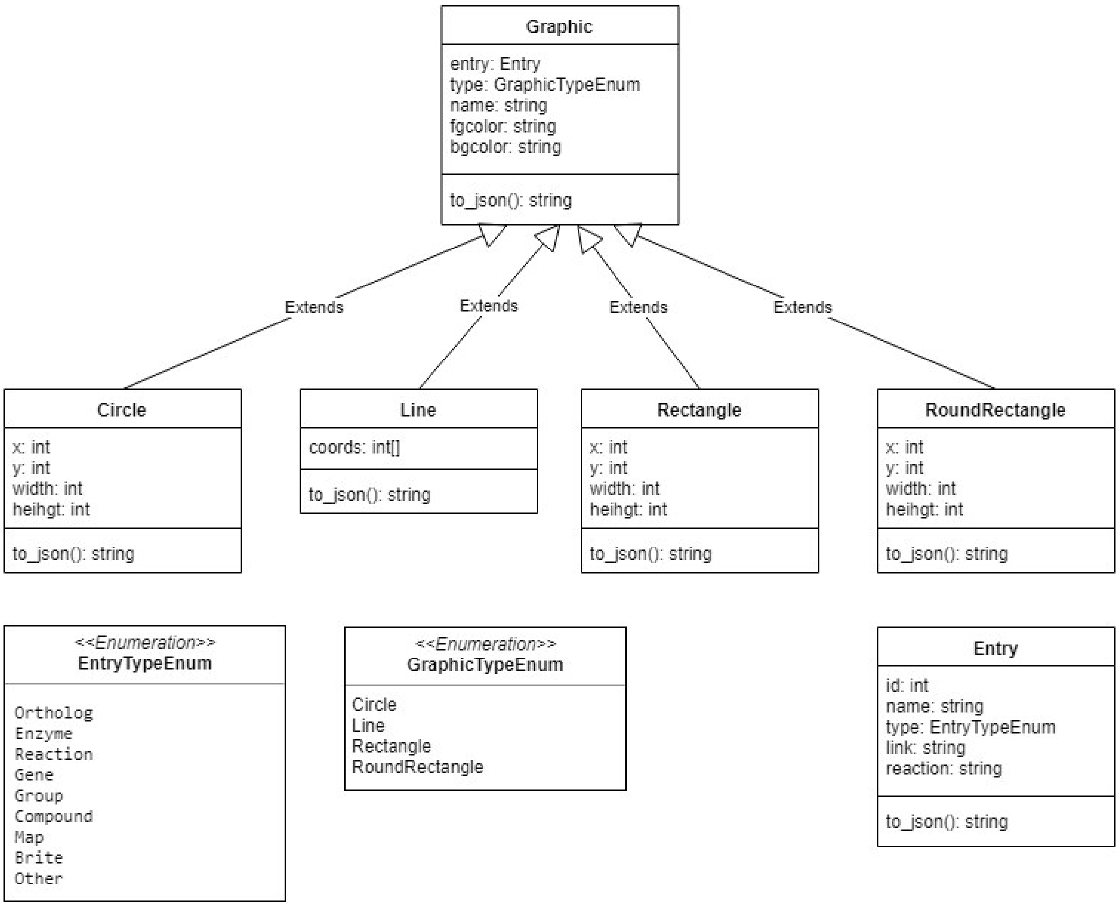


Figure 1: Class diagramm of backend.

in their package but in their examples. These are for rendering lines with linewidth greater than 1. Some of the scripts are changed, such that raycasting and cloning of the objects works.
To minify css and js files, the gulp tasks specified in \static\gulpfile.js are used.
The meshes to render are merged such that we have one mesh for all lines and one mesh for all circles, to reduce the number of draw calls. For the hover and click interaction we create single meshes for each graph element, against which we raycast. These meshes are not rendered, except single meshes when using them to highlight them on hover.

2 Project Structure

The project has the following structure:

```
/pr-pathway-exploration/
├── kegg
│   ├── __init__.py
│   ├── entry.py
│   ├── entryTypeEnum.py
│   ├── graphic.py
│   ├── graphicTypeEnum.py
│   └── keggQueryManager.py
├── static
│   ├── css
│   ├── fonts
│   ├── images
│   ├── js
│   │   ├── three
│   │   ├── hclVisualization.js
│   │   ├── kelpVisualization.js
│   │   ├── main.js
│   │   ├── pathway.js
│   │   └── visualizationManager.js
│   ├── node_modules
│   ├── resources
│   └── gulpfile.js
├── templates
├── venv
├── __init__.py
├── app.py
├── package-lock.json
├── pr-pathway-exploration.wsgi
└── util.py
```

References

[1] Kanehisa Laboratories. Human KO Entries. http://rest.kegg.jp/link/hsa/ko, accessed 20.02.2019.

[2] Kanehisa Laboratories. Human-Specific Pathway. http://rest.kegg.jp/get/hsa01100/kgml, accessed 20.02.2019.

[3] Kanehisa Laboratories. KEGG Organism List. http://rest.kegg.jp/list/organism, accessed 20.02.2019.

[4] Armin Ronacher. Flask Quickstart. http://flask.pocoo.org/docs/1.0/quickstart/a-minimal-application, accessed 20.02.2019.

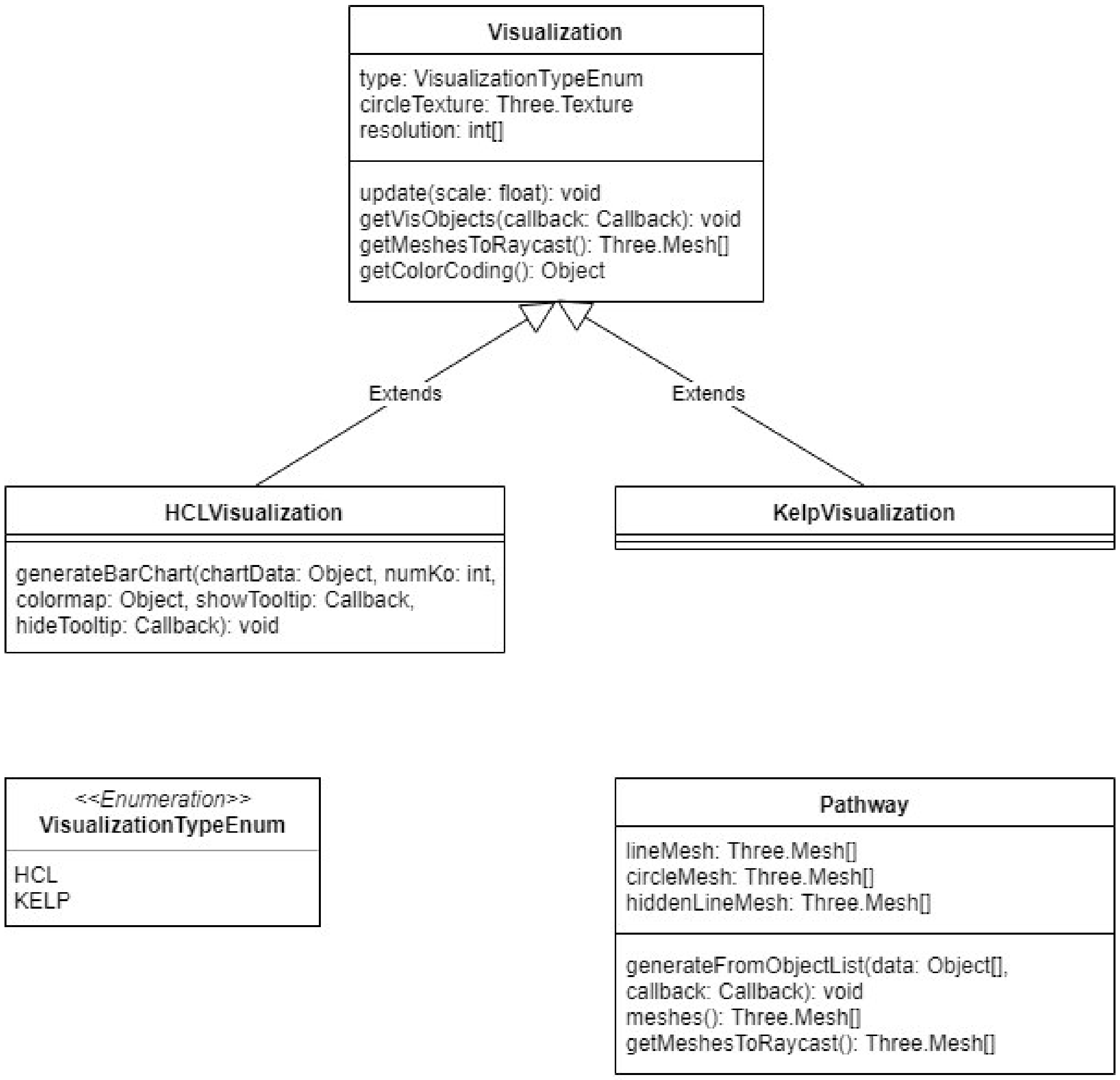

Visualization
type: VisualizationTypeEnum
circleTexture: Three.Texture
resolution: int[]
update(scale: float): void
getVisObjects(callback: Callback): void
getMeshesToRaycast(): Three.Mesh[]
getColorCoding(): Object
Extends
Extends
HCLVisualization
generateBarChart(chartData: Object, numKo: int, colormap: Object, showTooltip: Callback, hideTooltip: Callback): void
KelpVisualization
<<Enumeration>>
VisualizationTypeEnum
HCL
KELP
Pathway
lineMesh: Three.Mesh[]
circleMesh: Three.Mesh[]
hiddenLineMesh: Three.Mesh[]
generateFromObjectList(data: Object[], callback: Callback): void
meshes(): Three.Mesh[]
getMeshesToRaycast(): Three.Mesh[]

Figure 2: Class diagramm of frontend.

Pathway Exploration

Generated by Doxygen 1.8.15

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 pr Namespace Reference

-pathway-exploration::kegg::entryTypeEnum

4.1.1 Detailed Description

-pathway-exploration::kegg::entryTypeEnum

-pathway-exploration::kegg::keggQueryManager

-pathway-exploration::kegg::graphicTypeEnum

```
Enum representing the possible types of an element in the pathway map
```

```
Enum representing the possible types of a graphical element in the pathway map
```

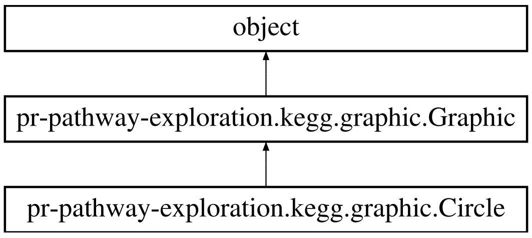
```
This Class handles requests to the KEGG API and their response
```

Chapter 5

Class Documentation

5.1 pr-pathway-exploration.kegg.graphic.Circle Class Reference

Inheritance diagram for pr-pathway-exploration.kegg.graphic.Circle:


object
pr-pathway-exploration.kegg.graphic.Graphic
pr-pathway-exploration.kegg.graphic.Circle

Public Member Functions

- def **__init__** (self, entry, $**$kwargs)
- def **to_json** (self)

Public Attributes

- **x**
- **y**
- **width**
- **height**

5.1.1 Detailed Description

```
Circle object representing circle entries in the pathway maps
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/graphic.py

5.2 pr-pathway-exploration.kegg.entry.Entry Class Reference

Public Member Functions

- def **__init__** (self, ∗∗kwargs)
- def **to_json** (self)

Public Attributes

- **id**
- **name**
- **type**
- **link**
- **reaction**

5.2.1 Detailed Description

```
Entry object representing the entries in the pathway maps
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/entry.py

5.3 pr-pathway-exploration.kegg.entryTypeEnum.EntryType Class Reference

Static Public Attributes

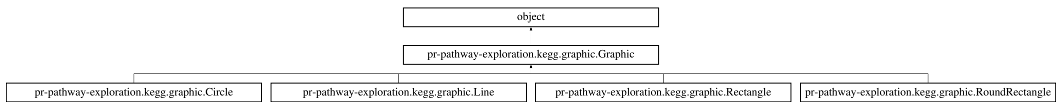
- int **ORTHOLOG** = 1
- int **ENZYME** = 2
- int **REACTION** = 3
- int **GENE** = 4
- int **GROUP** = 5
- int **COMPOUND** = 6
- int **MAP** = 7
- int **BRITE** = 8
- int **OTHER** = 9

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/entryTypeEnum.py

5.4 pr-pathway-exploration.kegg.graphic.Graphic Class Reference

Inheritance diagram for pr-pathway-exploration.kegg.graphic.Graphic:


object
pr-pathway-exploration.kegg.graphic.Graphic
pr-pathway-exploration.kegg.graphic.Circle
pr-pathway-exploration.kegg.graphic.Line
pr-pathway-exploration.kegg.graphic.Rectangle
pr-pathway-exploration.kegg.graphic.RoundRectangle

Public Member Functions

- def **__init__** (self, ∗∗kwargs)
- def **to_json** (self)

Public Attributes

- **entry**
- **type**
- **name**
- **fgcolor**
- **bgcolor**

5.4.1 Detailed Description

```
Graphic object representing the graphical objects in the pathway maps
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/graphic.py

5.5 pr-pathway-exploration.kegg.graphicTypeEnum.GraphicType Class Reference

Static Public Attributes

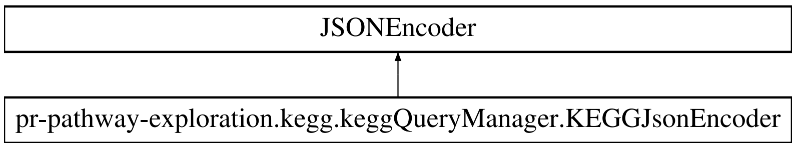
- int **CIRCLE** = 1
- int **LINE** = 2
- int **RECTANGLE** = 3
- int **ROUNDRECTANGLE** = 4

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/graphicTypeEnum.py

5.6 pr-pathway-exploration.kegg.keggQueryManager.KEGGJsonEncoder Class Reference

Inheritance diagram for pr-pathway-exploration.kegg.keggQueryManager.KEGGJsonEncoder:


JSONEncoder
pr-pathway-exploration.kegg.keggQueryManager.KEGGJsonEncoder

Public Member Functions

- def **default** (self, o)

5.6.1 Detailed Description

```
JsonEncoder for Graphic Objects
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/keggQueryManager.py

5.7 pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager Class Reference

Public Member Functions

- def keywords_to_request (self, keywords)
- def global_kgml_of_organism (self, organism)
- def kgml_to_graphic_list (self, kgmlText)
- def get_ko_ids (self, organism, entries)
- def **add_to_koidmap** (self, idx, entry_type, name, mapping)
- def calculate_id (self, firstArg, secondArg=None)
- def load_ko_ids (self)
- def load_ids (self)
- def write_ids (self)
- def write_ko_ids (self)

Static Public Member Functions

- def to_info_request (keywords)
- def to_list_request (keywords)
- def to_find_request (keywords)
- def to_get_request (keywords)
- def to_conv_request (keywords)
- def to_link_request (keywords)
- def to_ddi_request (keywords)
- def send_request (request)
- def handle_kgml_response (kgmltext)

Static Public Attributes

- string **url** = 'http://rest.kegg.jp/'
- **organisms** = None
- **idMap** = None
- **koIdMap** = None

5.7.1 Member Function Documentation

5.7.1.1 calculate_id()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.calculate_id (
              self,
              firstArg,
              secondArg = None )
```

```
Calculates the id for the coordinates of an entry
:param firstArg: for line, list of coords, for other first coord
:param secondArg: for line None, for other second coord
:return: id for the given arguments
```

5.7.1.2 get_ko_ids()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.get_ko_ids (
              self,
              organism,
              entries )
```

```
Requests the list of KO entries for an organism.
:param organism: KEGG organism identifier (3-4 letter)
:param entries: list of Graphic entries of the organism pathway
:return: Map of KO entries with the Graphic ids as values
```

5.7.1.3 global_kgml_of_organism()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.global_kgml_of_organism (
              self,
              organism )
```

```
Requests the gloabal pathway map of an organism from the KEGG API
:param organism: KEGG organism identifier (3-4 letters)
:return: response text of the request or None
```

5.7.1.4 handle_kgml_response()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.handle_kgml_response (
              kgmltext )  [static]
```

```
Handles the response for a request of a KGML file

:param kgmltext: response text for KGML request
:return: Parsed XML file
```

5.7.1.5 keywords_to_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.keywords_to_request (
              self,
              keywords )
```

```
categorizes the keywords into the request types of the KEGG API
:param keywords: keywords which are used to request from the KEGG API
:return: response from KEGG API
```

5.7.1.6 kgml_to_graphic_list()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.kgml_to_graphic_list (
              self,
              kgmlText )
```

```
Converts a kgml tree into an array of Graphic objects
:param kgmlText:
:return:
```

5.7.1.7 load_ids()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.load_ids (
              self )
```

```
Loads the idMap from the resources
:return: True if the Map was found in the resources, else False
```

5.7.1.8 load_ko_ids()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.load_ko_ids (
              self )
```

```
load KO id map from resources
:return: sets the class attribute koIdMap, returns True if the map was found in the resources, else false
```

5.7.1.9 send_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.send_request (
              request )  [static]
```

```
Sends a request to the KEGG API
:param request: request to be sent to the KEGG API
:return: response of the KEGG API if successful or None
```

5.7.1.10 to_conv_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.to_conv_request (
              keywords )  [static]
```

```
transforms the keywords to an CONV request according to the KEGG API
:param keywords:
:return: CONV request link for the KEGG API
```

5.7.1.11 to_ddi_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.to_ddi_request (
              keywords )  [static]
```

```
transforms the keywords to an DDI request according to the KEGG API
:param keywords:
:return: DDI request link for the KEGG API
```

5.7.1.12 to_find_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.to_find_request (
              keywords )  [static]
```

```
transforms the keywords to an FIND request according to the KEGG API
:param keywords:
:return: FIND request link for the KEGG API
```

5.7.1.13 to_get_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.to_get_request (
              keywords )  [static]
```

```
transforms the keywords to an GET request according to the KEGG API
:param keywords:
:return: GGET request link for the KEGG API
```

5.7.1.14 to_info_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.to_info_request (
              keywords )  [static]
```

```
transforms the keywords to an INFO request according to the KEGG API
:param keywords:
:return: INFO request link for the KEGG API
```

5.7.1.15 to_link_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.to_link_request (
              keywords )  [static]
```

```
transforms the keywords to an LINK request according to the KEGG API
:param keywords:
:return: LINK request link for the KEGG API
```

5.7.1.16 to_list_request()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.to_list_request (
              keywords )  [static]
```

```
transforms the keywords to an LIST request according to the KEGG API
:param keywords:
:return: LIST request link for the KEGG API
```

5.7.1.17 write_ids()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.write_ids (
              self )
```

```
Helper function to write the idMap to a file in the resources
:return:
```

5.7.1.18 write_ko_ids()

```
def pr-pathway-exploration.kegg.keggQueryManager.KEGGQueryManager.write_ko_ids (
              self )
```

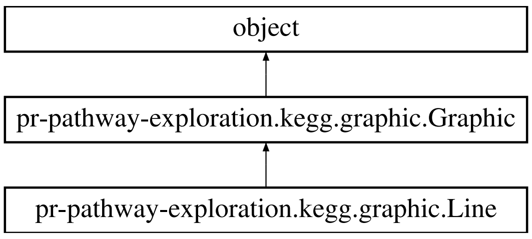
```
Helper function to write the koIdMap to a file in the resources
:return:
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/keggQueryManager.py

5.8 pr-pathway-exploration.kegg.graphic.Line Class Reference

Inheritance diagram for pr-pathway-exploration.kegg.graphic.Line:


object
pr-pathway-exploration.kegg.graphic.Graphic
pr-pathway-exploration.kegg.graphic.Line

Public Member Functions

- def **__init__** (self, entry, ∗∗kwargs)
- def **to_json** (self)

Public Attributes

- **coords**

5.8.1 Detailed Description

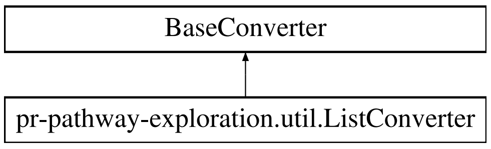
```
Line object representing line entries in the pathway maps
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/graphic.py

5.9 pr-pathway-exploration.util.ListConverter Class Reference

Inheritance diagram for pr-pathway-exploration.util.ListConverter:



Public Member Functions

- def **to_python** (self, value)
- def **to_url** (self, values)

5.9.1 Detailed Description

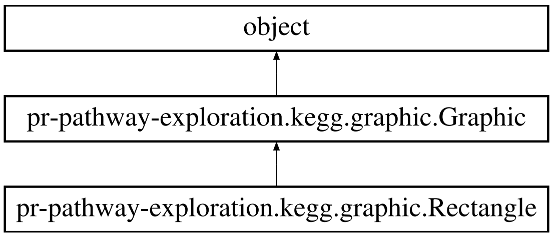
```
Converts lists of keywords concatenated with '+'
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/util.py

5.10 pr-pathway-exploration.kegg.graphic.Rectangle Class Reference

Inheritance diagram for pr-pathway-exploration.kegg.graphic.Rectangle:


object
pr-pathway-exploration.kegg.graphic.Graphic
pr-pathway-exploration.kegg.graphic.Rectangle

Public Member Functions

- def **__init__** (self, entry, ∗∗kwargs)
- def **to_json** (self)

Public Attributes

- **x**
- **y**
- **width**
- **height**

5.10.1 Detailed Description

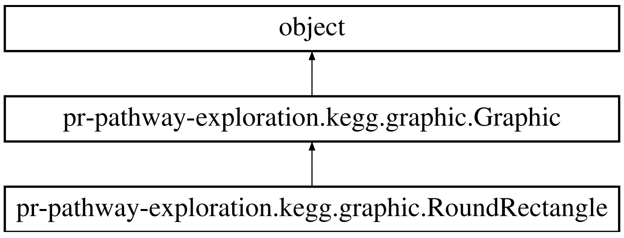
```
Rectangle object representing rectangle entries in the pathway maps
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/graphic.py

5.11 pr-pathway-exploration.kegg.graphic.RoundRectangle Class Reference

Inheritance diagram for pr-pathway-exploration.kegg.graphic.RoundRectangle:


object
pr-pathway-exploration.kegg.graphic.Graphic
pr-pathway-exploration.kegg.graphic.RoundRectangle

Public Member Functions

- def **__init__** (self, entry, **kwargs)
- def **to_json** (self)

Public Attributes

- **x**
- **y**
- **width**
- **height**

5.11.1 Detailed Description

```
RoundRectangle object representing roundrectangle entries in the pathway maps
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/kegg/graphic.py

5.12 pr-pathway-exploration.util.Util Class Reference

Public Member Functions

- def **compare_files_write_duplicates** (file1, file2, file_out, file_combination)

5.12.1 Detailed Description

```
Helper class to compare two files each containing a list of the KO entries of an organism
Outputs the shared entries in file file_out and all entries with a coloring to enter to the KEGG Search&Color
```

The documentation for this class was generated from the following file:

- pr-pathway-exploration/util.py

Class: Pathway

Pathway

new Pathway()

Represents a pathway map

Methods

circleColors()

Returns an array of the colors of the circle mesh

Returns:

array of Three Colors

Type
Array

circles()

Returns the circle objects of this pathway

Returns:

returns an array of all Circle objects of this pathway

Type
Array

circleTexture()

Returns the texture used for rendering the circles.

Returns:

returns the texture used for rendering the circles

Type
*

generateFromObjectList(data, callback)

Generates the pathway and its meshes from a list of graphic objects

Parameters:

Name	Type	Description
`data`		data to generate the meshes
`callback`		Callback is called when all meshes were generated

getMeshesToRaycast()

Returns the single meshes, which can be used to raycast

Returns:

returns an array of all meshes to raycast

Type
array

getOrthologs()

Returns the KO entries of this pathway

labels()

Returns the label objects of this pathway

Returns:

array of label objects

Type
Array

lineColors()

Returns an array of the colors of the line mesh

Returns:

array of Three Colors

Type
Array

lines()

Returns the line objects of this pathway

Returns:

returns an array of all Line objects of this pathway

Type
Array

---

meshes()

Returns all meshes which should be rendered.

Returns:

array of meshes to render

Type
array

---

name()

Returns the name of this pathway.

Returns:

name of the pathway.

Type
*

---

setCircleTexture(texture)

Sets the texture used for rendering the circles.

Parameters:

Name	Type	Description
`texture`		texture for rendering the circles

---

visHeight()

Returns the visualization height.

Returns:

height of the visualization

Type
number

---

visWidth()

Returns the visualization width.

Returns:

width of the visualization

Type
number

<inner> createBitmap(text, textWidth, textHeight, color, callback)

Creates a bitmap for the generation of text labels.

Parameters:

Name	Type	Description
`text`		text to write on the label
`textWidth`		width of the text
`textHeight`		height of the text
`color`		text color
`callback`		Callback is called when the label is created

<inner> createTextLabel(posX, posY, color, txtWidth, txtHeight, txt, data)

Creates the text labels for visualization.

Parameters:

Name	Type	Description
`posX`		position x of the label
`posY`		position y of the label
`color`		text color
`txtWidth`		text width
`txtHeight`		text height
`txt`		text
`data`		data to add as user data

<inner> createThreeLine(color, points, data)

Create a Line2 object from the points with the color and data

Parameters:

Name	Type	Description
`color`		color of the Line2 objects
`points`		points of the geometry
`data`		data to set as user data

<inner> nextPowerOfTwo(number)

Calculate the next power of two for the given number.

Parameters:

Name	Type	Description
`number`		

Returns:

Type
number

<inner> wrapText(context, text, x, y, maxWidth, lineHeight)

Function to wrap text, such that it fits into the specified rectangle.
https://www.html5canvastutorials.com/tutorials/html5-canvas-wrap-text-tutorial/
(http://www.html5canvastutorials.com/tutorials/html5-canvas-wrap-text-tutorial/)

Parameters:

Name	Type	Description
`context`		
`text`		
`x`		
`y`		
`maxWidth`		
`lineHeight`		

Class: VisualizationManager

VisualizationManager

new VisualizationManager(map, callback)

Handles rendering meshes to the scene, and interaction like zoom, pan, hover and click.

Parameters:

Name	Type	Description
`map`		initial map to show
`callback`		Callback to call, when initial map is set up

Methods

getKoPathway()

Returns the KO pathway

Returns:

KO Pathway (ko01100)

Type
Pathway (Pathway.html)

getMousedown()

Returns true if the mousebutton is currently pressed, else false

Returns:

True if mousebutton is pressed, else false

Type
boolean

getPathway()

Returns the current pathway

Returns:

pathway

Type
Pathway (Pathway.html)

---

getVisualization()

Returns the current visualization.

Returns:

either KelpVisualization or HCLVisualization

Type
Visualization

---

height()

Returns the height of the render container

Returns:

height of the render container

Type
number

---

setKoPathway(pathway)

Sets the KO pathway (ko01100)

Parameters:

Name	Type	Description
`pathway`		

---

setMousedown(value)

Setter for mousedown

Parameters:

Name	Type	Description
`value`		boolean specifying if the mouse is pressed or not

---

setPathway(p, callback)

Sets the pathway

Parameters:

Name	Type	Description
`p`		pathway object
`callback`		Callback is called, when all necessary objects are generated

setVisualization(vis, callback)

Sets a visualization to display

Parameters:

Name	Type	Description
`vis`		visualization which should be rendered to the render container
`callback`		

width()

Returns the width of the render container

Returns:

width of the render container

Type
number

<inner> checkIntersects(mouse_pos)

Raycasts the objects in intersectObjects at mouse_pos.

Parameters:

Name	Type	Description
`mouse_pos`		position to raycast

Returns:

intersection result

Type
*

<inner> checkTooltipPosition(mouse, position, offset)

Checks if the tooltip element is inside the window size

Parameters:

Name	Type	Description
`mouse`		mouse position
`position`		screen position
`offset`		offset from top and left

<inner> getCurrentScale()

Calculate the current zoom level.

Returns:

current zoom level

Type
number

<inner> handlePan(event)

Handles pan event

Parameters:

Name	Type	Description
`event`		d3 event

<inner> handleZoom(event)

Handles the zoom event and calculates the new camera position.

Parameters:

Name	Type	Description
`event`		d3 event

<inner> hideTooltip()

Hide tooltip when nothing is hovered.

<inner> highlightObject(object, mouse)

Handles the highlighting of elements when hovering

Parameters:

Name	Type	Description
`object`		object which should be highlighted
`mouse`		mouse position

<inner> onClick(event)

Handles a click event.

Parameters:

Name	Type	Description
`event`		

<inner> onMouseMove(event)

Handles a mouse move event.

Parameters:

Name	Type	Description
`event`		

<inner> onWindowResize()

Handles window resize event

<inner> removeHighlight()

Removes the current hover highlight.

<inner> render()

Render function which executes the draw calls.

<inner> setCircleHighlight(object)

Sets the hovering Highlight as a circle objects.

Parameters:

Name	Type	Description
`object`		object which should be highlighted

Returns:

the user data of this object

Type
*

<inner> setLineHighlight(object)

Sets the hovering Highlight as a line objects.

Parameters:

Name	Type	Description
`object`		object which should be highlighted

Returns:

the user data of this object

Type
*

<inner> setPathway(p, callback)

Sets the current pathway

Parameters:

Name	Type	Description
`p`		pathway which should be set
`callback`		Callback is called when all required objects are generated

<inner> setUpCamera()

Sets up the camera.

<inner> setUpRenderer()

Sets up the renderer.

<inner> setUpScene()

Sets up the scene.

<inner> setUpZoom()

Sets up the zoom.

<inner> showChartTooltip(data)

Show the tooltip for an element in the stacked bar charts

Parameters:

Name	Type	Description
`data`		

<inner> showEntryTooltip(data)

Show the tooltip for an graph entry

Parameters:

Name	Type	Description
`data`		object containing data to show

<inner> showTooltip(object, data, mouse)

Shows the tooltip when hovering over an element.

Parameters:

Name	Type	Description
`object`		object which is hovered
`data`		data to be displayed
`mouse`		mouse position

<inner> updateResolution()

Updates the resolution of Line2 objects in the visualization.

<inner> updateScale()

Updates the linewidth of Line2 objects according to the current scale.

Class: HCLVisualization

HCLVisualization

new HCLVisualization(compare)

Creates the HCL visualization

Parameters:

Name	Type	Description
`compare`		compare type

Methods

count()

Returns the number of pathways in the visualization

Returns:

number of pathways

Type
*

generateBarChart(chartdata, numKo, colormap, showTooltip, hideTooltip)

Generates the bar chart for the HCL visualization

Parameters:

Name	Type	Description
`chartdata`		data, necessary for the stacked bar charts
`numKo`		total number of KO entries in the ko01100 map
`colormap`		Color map with organism combinations as key and the calculated color as value
`showTooltip`		callback to show the tooltip on hover
`hideTooltip`		callback to hide the tooltip on mouse leave

getColorCoding()

Returns the color coding of the organisms in this visualization. The organism names are the keys and the colors are the values

getCompareType()

Returns the type of comparison.

Returns:

Type
*

getMeshesToRaycast()

Returns the single meshes which can be used to raycast.

Returns:

array of meshes

Type
Array

getPathways()

Returns the pathways

Returns:

array of pathways

Type
*

getType()

Returns the type of the visualization

Returns:

type of visualization ('KELP' | 'HCL')

Type
string

getVisualizationObjects(callback)

Returns the meshes which should be rendered to the scene.

Parameters:

Name	Type	Description
`callback`		

Returns:

array of meshes

Type
Array

setCircleTexture(texture)

Sets the texture to render the points mesh

Parameters:

Name	Type	Description
`texture`		texture to render the points mesh

setKoEntries(entries, meshes)

Sets the KO entries of the organisms

Parameters:

Name	Type	Description
`entries`		KO entries with arrays of organisms each
`meshes`		meshes of the KO pathway (ko01100)

setOrder(o)

Sets the order of the organisms

Parameters:

Name	Type	Description
`o`		map with organism names as keys and index as values

setResolution(res)

Sets the resolution.

Parameters:

Name	Type	Description

Name	Type	Description
`res`		array with x and y entry

update(currScale)

Updates the linewidth of Line2 meshes according to the current scale

Parameters:

Name	Type	Description
`currScale`		

<inner> addLine(obj, kolist, orgmap)

Merges an object into the line geometry.

Parameters:

Name	Type	Description
`obj`		to clone and merge to the geometry
`kolist`		list of koentries of the organisms
`orgmap`		map of organisms with organsim names as keys and index as values

<inner> addPoint(obj)

Merges an object into the circle geometry.

Parameters:

Name	Type	Description
`obj`		to clone and merge to the geometry

<inner> calculateBiggestValidChroma(hue, chroma, luminance)

Calculates the biggest chroma, such that the color is inside the color space.

Parameters:

Name	Type	Description
`hue`		hue of the color to calculate

Name	Type	Description
`chroma`		chroma to start calculation
`luminance`		luminance of the color to calculate

Returns:

HCL color within HCL color space

Type
*

<inner> calculateColors(keys, orgmap)

Parameters:

Name	Type	Description
`keys`		
`orgmap`		

<inner> calculateLuminance(numOrganisms)

Calculates the luminance for a given number of organisms

Parameters:

Name	Type	Description
`numOrganisms`		number of organisms, which share an entry

Returns:

luminance dependent on the number of organisms

Type
number

<inner> generateBarChart(chartdata, numKo, colormap, showTooltip, hideTooltip)

Generates the bar chart for the HCL visualization

Parameters:

Name	Type	Description
`chartdata`		data, necessary for the stacked bar charts
`numKo`		total number of KO entries in the ko01100 map

Name	Type	Description
`colormap`		Color map with organism combinations as key and the calculated color as value
`showTooltip`		callback to show the tooltip on hover
`hideTooltip`		callback to hide the tooltip on mouse leave

<inner> generateVisualization(kolist, meshes)

Generates the visualization meshes

Parameters:

Name	Type	Description
`kolist`		list of ko entries of the organisms
`meshes`		meshes from the ko map (ko01100)

<inner> getOrganismColorForIndex(index)

Returns the color of an organism for the organism index given.

Parameters:

Name	Type	Description
`index`		

Returns:

Type
THREE.Color

<inner> getOrganismColorForName(name)

Returns the color of an organisms for the organism name given.

Parameters:

Name	Type	Description
`name`		name of the organism

Returns:

Type
THREE.Color

<inner> interpolateColorForOrganisms(orgs)

Calculates the color according to the organism indices specified.

Parameters:

Name	Type	Description
`orgs`		array of organism indices

Returns:

Type
THREE.Color

<inner> setOrder(o)

Sets the order of the organisms in the visualization.

Parameters:

Name	Type	Description
`o`		map with organism names as keys and index as values

<inner> visualizeObject(objectColor, organism, id)

Determines if an object should by visualized or not.

Parameters:

Name	Type	Description
`objectColor`		color of the object specified in the pathway
`organism`		the organism, the object belongs to
`id`		id of the object

Returns:

Type
boolean

Class: KelpVisualization

KelpVisualization

new KelpVisualization(compare)

Creates the kelp visualization

Parameters:

Name	Type	Description
`compare`		compare type

Methods

getChartData(entries, meshes)

Gets the chartdata of this visualization.

Parameters:

Name	Type	Description
`entries`		ko entries
`meshes`		mehses of the ko map

getColorCoding()

Returns the color coding of the organisms in the visualization. The organism names as keys and colors as values.

getCompareType()

Returns the type of comparison.

Returns:

type of comparison

Type
*

getMeshesToRaycast()

Returns an array of single meshes which can be used to raycast

Returns:

Type
Array

getType()

Returns the visualization type

Returns:

visualization type ('KELP' | 'HCL')

Type
string

getVisualizationObjects(callback)

Returns the object which are rendered to the scene.

Parameters:

Name	Type	Description
`callback`		

Returns:

array of meshes to render

Type
Array

setCircleTexture(texture)

Sets the circle texture to render the point geometry.

Parameters:

Name	Type	Description
`texture`		texture to render the points of the circle geometry

setKoEntries(entries, meshes)

Sets the KO entries of the organisms of this visualization.

Parameters:

Name	Type	Description

Name	Type	Description
`entries`		ko entries
`meshes`		mehses of the ko map

setOrder(o)

Sets the order of the organisms of the visualization

Parameters:

Name	Type	Description
`o`		map with organism names as keys and index as values

setResolution(res)

Sets the resolution of the visualization.

Parameters:

Name	Type	Description
`res`		array with x and y resolution

update(currScale)

Updates the linewidth of Line2 meshes according to the current scale.

Parameters:

Name	Type	Description
`currScale`		current scale

<inner> addLine(obj, koentries, entries, orgVertices, sharedColors)

Merges a line to the line mesh.

Parameters:

Name	Type	Description
`obj`		object to clone and merge to the line mesh
`koentries`		KO entries

Name	Type	Description
`entries`		
`orgVertices`		line Vertices per organism
`sharedColors`		colors for shared layer

<inner> addPoint(obj)

Merges a point to the circle mesh.

Parameters:

Name	Type	Description
`obj`		to clone and merge to circle mesh

<inner> generateVisualization(entries, meshes)

Generates the visualization from the KO entries and meshes of the ko01100 map.

Parameters:

Name	Type	Description
`entries`		KO entries with arrays of the organisms
`meshes`		meshes of the ko01100 map

<inner> getColorForIndex(index)

Returns the organism color for a given index

Parameters:

Name	Type	Description
`index`		of the organism

Returns:

Type
THREE.Color

<inner> getWidth(name, type, scale)

Calculates the width for an element according to its order and the current scale.

Parameters:

Name	Type	Description
`name`		name of the organism
`type`		type of the mesh
`scale`		current scale

Returns:

width of the object

Type
number

<inner> setOrder(o)

Sets the order of the organisms.

Parameters:

Name	Type	Description
`o`		map with organisms as key and index as values

<inner> updateVisualization()

Update the visualization objects according to the current scale.