RadiPOP_API

Generated by Doxygen 1.9.2

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 run_app Namespace Reference

Functions

- def correctPartition ()

  Reveives index to slice/mask + coordinates--> returns partion corrected mask as PNG to client.
- def drawOnMask ()

  Reveives index to slice/mask + x,y coordinates --> returns drawn on mask as PNG to client.
- def extendThresholds ()

  Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.
- def getMask ()

  Reveives index to slice/mask --> returns mask stored on flask server as PNG to client.
- def highlightOrgan ()

  Reveives index of slice + x,y coordinates --> returns highlighted mask as PNG to client.
- def initialize ()

  Receive Paths to ordered slices, caches slices.
- def labelOrgan ()

  Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.
- def postPickleGetMask ()

  Receives path to pickle file --> returns mask as PNG to client.
- def saveMasks ()

  Reveives path, saves all stored masks as pickle files to path --> returns output path.
- def updateMask ()

  Receives index of slice + slider values --> returns updated mask as PNG to client.

Variables

- app = Flask(__name__)
- int FLASK_PORT = 4041
- string FLAST_HOST = '0.0.0.0'
- host
- dictionary patients = {}

  Dictionary which will hold for each patientID a RadiPopGUI object.
- port

3.1.1 Function Documentation

3.1.1.1 correctPartition()

```
def run_app.correctPartition ( )
```

Reveives index to slice/mask + coordinates--> returns partion corrected mask as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated
coordinates	array of coordinates of the form [x0,y0,x1,y1,...,xn,yn]

Returns

mask as transparent PNG as byte stream

Note: The coordinates array will be used to generate a line that cuts/divides the segmented organs.

Example handling of return image stream in js:

```
bytestring = data['status']
img = bytestring.split('\'')[1]
target.src ="" + img;
```

3.1.1.2 drawOnMask()

```
def run_app.drawOnMask ( )
```

Reveives index to slice/mask + x,y coordinates --> returns drawn on mask as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated
coordinates	array of coordinates of the form [x0,y0,x1,y1,...,xn,yn]

Returns

mask as transparent PNG as byte stream

Note: The coordinates array will be used to draw a line on the mask.

Example handling of return image stream in js:

```
bytestring = data['status']
img = bytestring.split('\'')[1]
target.src ="" + img;
```

3.1.1.3 extendThresholds()

```
def run_app.extendThresholds ( )
```

Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated
left	Extend labeling up to index-label
right	Extend labeling up to index+label

Returns

json data containing left_most_idx and right_most_idx

Note: The left_most_idx and right_most_idx correspond to the indices of the slices up to which the labeling has been extended. After the the function has finished use the function API's function /getMask to update the masks in your GUI. Example in js:

```
for (let index=parseInt(data["left_most_idx"]); index<parseInt(data["right_most_idx"])+1; index++) {
    $.post(FLASK_SERVER+"/getMask", {
        javascript_data: JSON.stringify({patienID: id, index: idx})
    })
}
```

3.1.1.4 getMask()

```
def run_app.getMask ( )
```

Reveives index to slice/mask --> returns mask stored on flask server as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated

Returns

mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
bytestring = data['status']
img = bytestring.split('\'')[1]
target.src ="" + img;
```

3.1.1.5 highlightOrgan()

```
def run_app.highlightOrgan ( )
```

Reveives index of slice + x,y coordinates --> returns highlighted mask as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated
x	relative x coordinates (0<=x<=1)
y	relative y coordinates (0<=y<=1)

Returns

mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
bytestring = data['status']
img = bytestring.split('\'')[1]
target.src ="" + img;
```

3.1.1.6 initialize()

```
def run_app.initialize ( )
```

Receive Paths to ordered slices, caches slices.

Parameters

patientID	The ID of the patient
paths	An array with the paths to the slices

Returns

200,OK

Note: Paths to slices !!!MUST BE ORDERED!!! 0,1,..,n

3.1.1.7 labelOrgan()

```
def run_app.labelOrgan ( )
```

Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated
label	Label of organ (1 for liver, 2 for spleen, 0 nothing, >2 other organ)

Returns

mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
bytestring = data['status']
img = bytestring.split('\'')[1]
target.src ="" + img;
```

3.1.1.8 postPickleGetMask()

```
def run_app.postPickleGetMask ( )
```

Receives path to pickle file --> returns mask as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice the mask refers to
path	The path to the mask file

Returns

mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
bytestring = data['status']
img = bytestring.split('\'')[1]
target.src ="" + img;
```

3.1.1.9 saveMasks()

```
def run_app.saveMasks ( )
```

Reveives path, saves all stored masks as pickle files to path --> returns output path.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated

Returns

path/directory to which the pickle files were written

3.1.1.10 updateMask()

```
def run_app.updateMask ( )
```

Receives index of slice + slider values --> returns updated mask as PNG to client.

Parameters

patientID	The ID of the patient
index	The index of the slice for which to mask should be updated
liver-intensity-slider	Slider value for liver intesity
bone-intensity-slider	Slider value for bone intesity
blood-vessel-intensity-slider	Slider value for blood-vessel intesity

Returns

mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
bytestring = data['status']
img = bytestring.split('\'')[1]
target.src ="" + img;
```

3.1.2 Variable Documentation

3.1.2.1 app

```
app = Flask(__name__)
```

3.1.2.2 FLASK_PORT

```
int FLASK_PORT = 4041
```

3.1.2.3 FLAST_HOST

```
FLAST_HOST = '0.0.0.0'
```

3.1.2.4 host

```
host
```

3.1.2.5 patients

```
dictionary patients = {}
```

Dictionary which will hold for each patientID a RadiPopGUI object.

Patients are added by the API's /initialize function

3.1.2.6 port

```
port
```

Chapter 4

File Documentation

4.1 /Users/lorenz/Desktop/RadiPOP-standalone/web_app/run_app.py File Reference

Namespaces

- namespace run_app

Functions

- def correctPartition ()

  Reveives index to slice/mask + coordinates--> returns partion corrected mask as PNG to client.
- def drawOnMask ()

  Reveives index to slice/mask + x,y coordinates --> returns drawn on mask as PNG to client.
- def extendThresholds ()

  Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.
- def getMask ()

  Reveives index to slice/mask --> returns mask stored on flask server as PNG to client.
- def highlightOrgan ()

  Reveives index of slice + x,y coordinates --> returns highlighted mask as PNG to client.
- def initialize ()

  Receive Paths to ordered slices, caches slices.
- def labelOrgan ()

  Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.
- def postPickleGetMask ()

  Receives path to pickle file --> returns mask as PNG to client.
- def saveMasks ()

  Reveives path, saves all stored masks as pickle files to path --> returns output path.
- def updateMask ()

  Receives index of slice + slider values --> returns updated mask as PNG to client.

Variables

- app = Flask(__name__)
- int FLASK_PORT = 4041
- string FLAST_HOST = '0.0.0.0'
- host
- dictionary patients = {}

  Dictionary which will hold for each patientID a RadiPopGUI object.
- port

Index