stable-diffusion web ui api

API client for AUTOMATIC1111/stable-diffusion-webui

Supports txt2img, img2img, extra-single-image, extra-batch-images API calls.

API support have to be enabled from webui. Add --api when running webui. It's explained here.

You can use --api-auth user1:pass1,user2:pass2 option to enable authentication for api access.

(Since it's basic http authentication the password is transmitted in cleartext)

API calls are (almost) direct translation from http://127.0.0.1:7860/docs as of 2022/11/21.

【译：AUTOMATIC1111/stable-diffusion-webui 的 API 客户端

支持 txt2img、img2img、extra-single-image、extra-batch-images API 调用。

必须从 webui 启用 API 支持。运行 webui 时添加 --api。 这里对此进行了解释。

您可以使用 --api-auth user1：pass1，user2：pass2 选项为 api 访问启用身份验证。 （由于是基本的http身份验证，因此密码以明文形式传输）

截至 2022 年 11 月 21 日，API 调用（几乎）直接从 http://127.0.0.1:7860/docs 翻译而来。】

Install【安装】

```
pip install webuiapi
```

Usage【用法】

webuiapi_demo.ipynb contains example code with original images. Images are compressed as jpeg in this document.

【译：webuiapi_demo.ipynb 包含包含原始图像的示例代码。在本文档中，图像被压缩为 jpeg。】

create API client【创建api客户端】

```
import webuiapi

# create API client
api = webuiapi.WebUIApi()

# create API client with custom host, port
#api = webuiapi.WebUIApi(host='127.0.0.1', port=7860)

# create API client with custom host, port and https
#api = webuiapi.WebUIApi(host='webui.example.com', port=443, use_https=True)

# create API client with default sampler, steps.
#api = webuiapi.WebUIApi(sampler='Euler a', steps=20)

# optionally set username, password when --api-auth=username:password is set on webui.
# username, password are not protected and can be derived easily if the communication channel is not encrypted.
# you can also pass username, password to the WebUIApi constructor.
api.set_auth('username', 'password')
```

txt2img（2=to，以下函数方法名称皆为**格式转到**格式的含义）

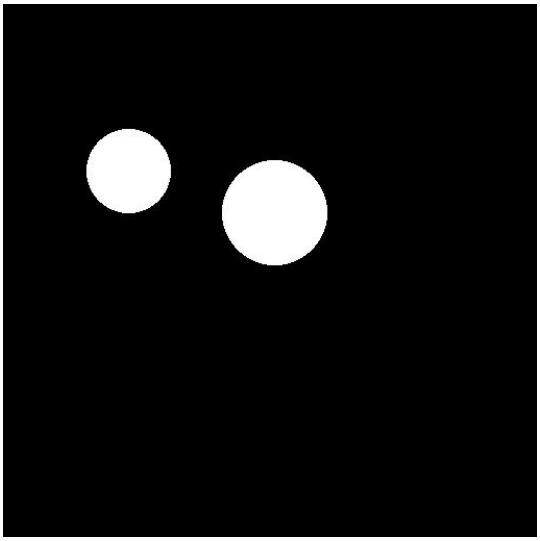
```
result1 = api.txt2img(prompt="cute squirrel",
                      negative_prompt="ugly, out of frame",
                      seed=1003,
                      styles=["anime"],
                      cfg_scale=7,
#                      sampler_index='DDIM',
#                      steps=30,
#                      enable_hr=True,
#                      hr_scale=2,
#                      hr_upscaler=webuiapi.HiResUpscaler.Latent,
#                      hr_second_pass_steps=20,
#                      hr_resize_x=1536,
#                      hr_resize_y=1024,
#                      denoising_strength=0.4,

                      )
# images contains the returned images (PIL images)
result1.images

# image is shorthand for images[0]
result1.image

# info contains text info about the api call
result1.info

# info contains paramteres of the api call
result1.parameters

result1.image
```

img2img

```
result2 = api.img2img(images=[result1.image], prompt="cute cat", seed=5555, cfg_scale=6.5, denoising_strength=0.6)
result2.image
```

img2img inpainting【译：修复】

```
from PIL import Image, ImageDraw

mask = Image.new('RGB', result2.image.size, color = 'black')
# mask = result2.image.copy()
draw = ImageDraw.Draw(mask)
draw.ellipse((210,150,310,250), fill='white')
draw.ellipse((80,120,160,120+80), fill='white')

mask
```



```
inpainting_result = api.img2img(images=[result2.image],
                                mask_image=mask,
                                inpainting_fill=1,
                                prompt="cute cat",
                                seed=104,
                                cfg_scale=5.0,
                                denoising_strength=0.7)
inpainting_result.image
```

extra-single-image【译：超单图像】

```
result3 = api.extra_single_image(image=result2.image,
                                 upscaler_1=webuiapi.Upscaler.ESRGAN_4x,
                                 upscaling_resize=1.5)
print(result3.image.size)
result3.image
```

(768, 768)

extra-batch-images【译：超批处理映像】

```
result4 = api.extra_batch_images(images=[result1.image, inpainting_result.image],
                                 upscaler_1=webuiapi.Upscaler.ESRGAN_4x,
                                 upscaling_resize=1.5)
result4.images[0]
```

result4.images[1]

Async API support【译：异步api支持】

txt2img, img2img, extra_single_image, extra_batch_images support async api call with use_async=True parameter. You need asyncio, aiohttp packages installed.

【译：txt2img、img2img、extra_single_image extra_batch_images 支持使用 use_async=True 参数的异步 API 调用。您需要安装 asyncio、aiohttp 软件包。】

```
result = await api.txt2img(prompt="cute kitten",
                     seed=1001,
                     use_async=True
                     )
result.image
```

Scripts support【译：脚本支持】

Scripts from AUTOMATIC1111's Web UI are supported, but there aren't official models that define a script's interface.

【译：支持来自 AUTOMATIC1111 Web UI 的脚本，但没有定义脚本界面的官方模型。】

To find out the list of arguments that are accepted by a particular script look up the associated python file from AUTOMATIC1111's repo `scripts/[script_name].py` . Search for its `run(p, **args)` function and the arguments that come after 'p' is the list of accepted arguments

【译：要找出特定脚本接受的参数列表，请从以下位置查找关联的 python 文件 AUTOMATIC1111的回购scripts/[script_name].py。搜索run(p, **args)其功能和随之而来的参数 “p”后面是接受的参数列表】

Example for X/Y/Z Plot script:【译：x/y/z绘图脚本支持】

```
(scripts/xyz_grid.py file from AUTOMATIC1111's repo)

    def run(self, p, x_type, x_values, y_type, y_values, z_type, z_values, draw_legend, include_lone_images, include_sub_grids, no_fixed_seeds, margin_size):
    ...
```

List of accepted arguments:

- x_type: Index of the axis for X axis. Indexes start from [0: Nothing]
- x_values: String of comma-separated values for the X axis
- y_type: Index of the axis type for Y axis. As the X axis, indexes start from [0: Nothing]
- y_values: String of comma-separated values for the Y axis
- z_type: Index of the axis type for Z axis. As the X axis, indexes start from [0: Nothing]
- z_values: String of comma-separated values for the Z axis
- draw_legend: "True" or "False". IMPORTANT: It needs to be a string and not a Boolean value
- include_lone_images: "True" or "False". IMPORTANT: It needs to be a string and not a Boolean value
- include_sub_grids: "True" or "False". IMPORTANT: It needs to be a string and not a Boolean value
- no_fixed_seeds: "True" or "False". IMPORTANT: It needs to be a string and not a Boolean value
- margin_size: int value

【译：
接受的参数列表：

- x_type：X 轴的轴索引。索引从 [0： Nothing] 开始
- x_values：X 轴的逗号分隔值字符串
- y_type：Y 轴的轴类型的索引。作为 X 轴，索引从 [0： Nothing] 开始
- y_values：Y 轴的逗号分隔值字符串
- z_type：Z 轴的轴类型的索引。作为 X 轴，索引从 [0： Nothing] 开始
- z_values：Z 轴的逗号分隔值字符串
- draw_legend：“真”或“假”。重要提示：它必须是字符串，而不是布尔值
- include_lone_images：“真”或“假”。重要提示：它必须是字符串，而不是布尔值

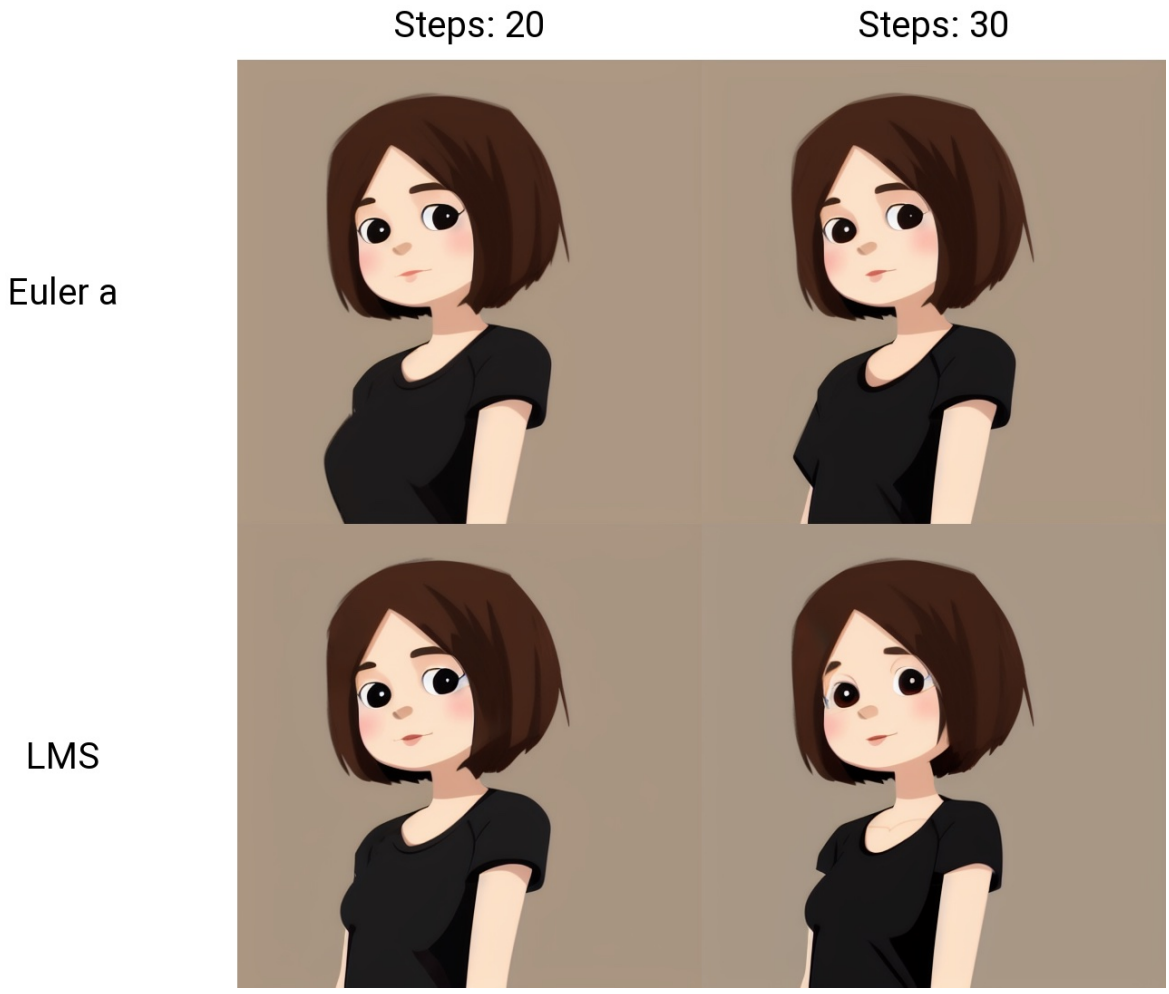
- include_sub_grids：“True”或“False”。重要提示：它必须是字符串，而不是布尔值
- no_fixed_seeds：“True”或“False”。重要提示：它必须是字符串，而不是布尔值
- margin_size：int 值

】

```
# Available Axis options (Different for txt2img and img2img!)
XYZPlotAvailableTxt2ImgScripts = [
    "Nothing",
    "Seed",
    "Var. seed",
    "Var. strength",
    "Steps",
    "Hires steps",
    "CFG Scale",
    "Prompt S/R",
    "Prompt order",
    "Sampler",
    "Checkpoint name",
    "Sigma Churn",
    "Sigma min",
    "Sigma max",
    "Sigma noise",
    "Eta",
    "Clip skip",
    "Denoising",
    "Hires upscaler",
    "VAE",
    "Styles",
]

XYZPlotAvailableImg2ImgScripts = [
    "Nothing",
    "Seed",
    "Var. seed",
    "Var. strength",
    "Steps",
    "CFG Scale",
    "Image CFG Scale",
    "Prompt S/R",
    "Prompt order",
    "Sampler",
    "Checkpoint name",
    "Sigma Churn",
```

```
    "Sigma min",
    "Sigma max",
    "Sigma noise",
    "Eta",
    "Clip skip",
    "Denoising",
    "Cond. Image Mask Weight",
    "VAE",
    "Styles",
]

# Example call
XAxisType = "Steps"
XAxisValues = "20,30"
YAxisType = "Sampler"
YAxisValues = "Euler a, LMS"
ZAxisType = "Nothing"
ZAxisValues = ""
drawLegend = "True"
includeLoneImages = "False"
includeSubGrids = "False"
noFixedSeeds = "False"
marginSize = 0


# x_type, x_values, y_type, y_values, z_type, z_values, draw_legend,
include_lone_images, include_sub_grids, no_fixed_seeds, margin_size

result = api.txt2img(
                    prompt="cute girl with short brown hair in black t-shirt in
animation style",
                    seed=1003,
                    script_name="X/Y/Z Plot",
                    script_args=[
                        XYZPlotAvailableTxt2ImgScripts.index(XAxisType),
                        XAxisValues,
                        XYZPlotAvailableTxt2ImgScripts.index(YAxisType),
                        YAxisValues,
                        XYZPlotAvailableTxt2ImgScripts.index(ZAxisType),
                        ZAxisValues,
```

```
                        drawLegend,
                        includeLoneImages,
                        includeSubGrids,
                        noFixedSeeds,
                        marginSize,                         ]
                    )

result.image
```



Configuration APIs【配置api】

```
# return map of current options
options = api.get_options()

# change sd model
options = {}
options['sd_model_checkpoint'] = 'model.ckpt [7460a6fa]'
api.set_options(options)

# when calling set_options, do not pass all options returned by get_options().
# it makes webui unusable (2022/11/21).

# get available sd models
api.get_sd_models()

# misc get apis
api.get_samplers()
api.get_cmd_flags()
api.get_hypernetworks()
api.get_face_restorers()
api.get_realesrgan_models()
api.get_prompt_styles()
api.get_artist_categories() # deprecated ?
api.get_artists() # deprecated ?
api.get_progress()
api.get_embeddings()
api.get_cmd_flags()
api.get_scripts()
api.get_memory()

# misc apis
api.interrupt()
api.skip()
```

Utility methods【译：实用方法】

```
# save current model name
old_model = api.util_get_current_model()

# get list of available models
models = api.util_get_model_names()

# refresh list of models
api.refresh_checkpoints()

# set model (use exact name)
api.util_set_model(models[0])

# set model (find closest match)
api.util_set_model('robodiffusion')

# wait for job complete
api.util_wait_for_ready()

```

LORA and alwayson_scripts example【译：lora和alwayon_scripts例子】

```
r = api.txt2img(prompt='photo of a cute girl with green hair <lora:Moxin_10:0.6> shuimobysim __juice__',
                seed=1000,
                save_images=True,
                alwayson_scripts={"Simple wildcards":[]} # wildcards extension doesn't accept more parameters.
               )
r.image
```

Extension support - Model-Keyword【译：扩展支持-model-keyword】

```
# https://github.com/mix1009/model-keyword
mki = webuiapi.ModelKeywordInterface(api)
mki.get_keywords()
```

ModelKeywordResult(keywords=['nousr robot'], model='robo-diffusion-v1.ckpt', oldhash='41fef4bd', match_source='model-keyword.txt')

Extension support - Instruct-Pix2Pix【译：扩展支持instruct-pix2pix】

```
# Instruct-Pix2Pix extension is now deprecated and is now part of webui.
# You can use normal img2img with image_cfg_scale when instruct-pix2pix model is loaded.
r = api.img2img(prompt='sunset', images=[pil_img], cfg_scale=7.5, image_cfg_scale=1.5)
r.image
```

Extension support - ControlNet【译：扩展支持-controlnet】

```
# https://github.com/Mikubill/sd-webui-controlnet

api.controlnet_model_list()
```

```
['control_v11e_sd15_ip2p [c4bb465c]',
 'control_v11e_sd15_shuffle [526bfdae]',
 'control_v11f1p_sd15_depth [cfd03158]',
 'control_v11p_sd15_canny [d14c016b]',
 'control_v11p_sd15_inpaint [ebff9138]',
 'control_v11p_sd15_lineart [43d4be0d]',
 'control_v11p_sd15_mlsd [aca30ff0]',
 'control_v11p_sd15_normalbae [316696f1]',
 'control_v11p_sd15_openpose [cab727d4]',
 'control_v11p_sd15_scribble [d4ba51ff]',
 'control_v11p_sd15_seg [e1f51eb9]',
 'control_v11p_sd15_softedge [a8575a2a]',
 'control_v11p_sd15s2_lineart_anime [3825e83e]',
 'control_v11u_sd15_tile [1f041471]']
```

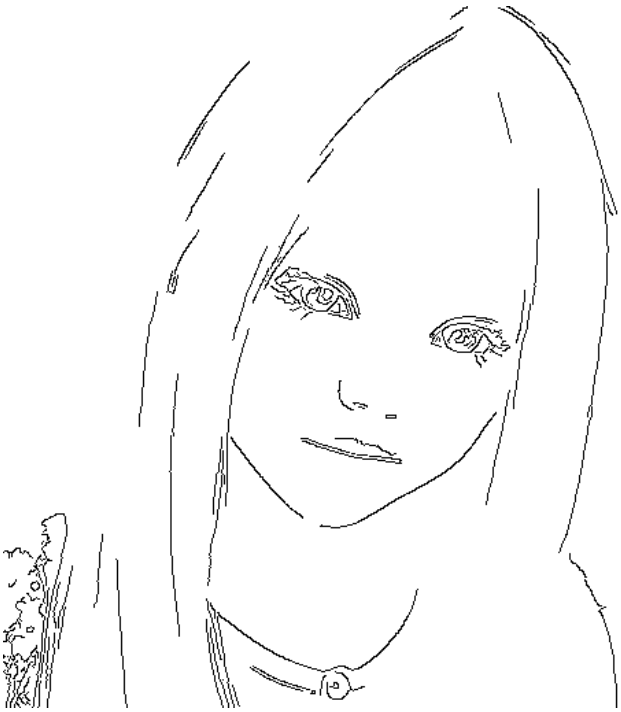
```
api.controlnet_version()
api.controlnet_module_list()
```

```
# normal txt2img
r = api.txt2img(prompt="photo of a beautiful girl with blonde hair", height=512, seed=100)
img = r.image
img
```

```
# txt2img with ControlNet (used 1.0 but also supports 1.1)
unit1 = webuiapi.ControlNetUnit(input_image=img, module='canny', model='control_canny-fp16 [e3fe7712]')

r = api.txt2img(prompt="photo of a beautiful girl", controlnet_units=[unit1])
r.image
```

```
# img2img with multiple ControlNets (used 1.0 but also supports 1.1)
unit1 = webuiapi.ControlNetUnit(input_image=img, module='canny', model='control_canny-fp16 [e3fe7712]')
unit2 = webuiapi.ControlNetUnit(input_image=img, module='depth', model='control_depth-fp16 [400750f6]', weight=0.5)

r2 = api.img2img(prompt="girl",
             images=[img],
             width=512,
             height=512,
             controlnet_units=[unit1, unit2],
             sampler_name="Euler a",
             cfg_scale=7,
            )
r2.image
```

```
r2.images[1]
```



```
r2.images[2]
```

```
r = api.controlnet_detect(images=[img], module='canny')
r.image
```

Extension support - RemBG (contributed by webcoderz)

```
# https://github.com/AUTOMATIC1111/stable-diffusion-webui-rembg
rembg = webuiapi.RemBGInterface(api)
r = rembg.rembg(input_image=img, model='u2net', return_mask=False)
r.image
```

Extension support - SegmentAnything (contributed by TimNekk)

```
# https://github.com/continue-revolution/sd-webui-segment-anything

segment = webuiapi.SegmentAnythingInterface(api)

# Perform a segmentation prediction using the SAM model using points
sam_result = segment.sam_predict(
    image=img,
    sam_positive_points=[(0.5, 0.25), (0.75, 0.75)],
    # add other parameters as needed
)

# Perform a segmentation prediction using the SAM model using GroundingDINO
sam_result2 = segment.sam_predict(
    image=img,
    dino_enabled=True,
    dino_text_prompt="A text prompt for GroundingDINO",
    # add other parameters as needed
)

# Example of dilating a mask
dilation_result = segment.dilate_mask(
    image=img,
    mask=sam_result.masks[0],  # using the first mask from the SAM prediction
    dilate_amount=30
)

# Example of generating semantic segmentation with category IDs
semantic_seg_result = segment.sam_and_semantic_seg_with_cat_id(
    image=img,
    category="1+2+3",  # Category IDs separated by '+'
    # add other parameters as needed
)
```

Extension support - ADetailer (contributed by tomj2ee and davidmartinrius)

txt2img with ADetailer

```python
# https://github.com/Bing-su/adetailer

import webuiapi

api = webuiapi.WebUIApi()

ads = webuiapi.ADetailer(ad_model="face_yolov8n.pt")

result1 = api.txt2img(prompt="cute squirrel",
                      negative_prompt="ugly, out of frame",
                      seed=-1,
                      styles=["anime"],
                      cfg_scale=7,
                      adetailer=[ads],
                      steps=30,
                      enable_hr=True,
                      denoising_strength=0.5
             )


img = result1.image
img

# OR

file_path = "output_image.png"
result1.image.save(file_path)
```

img2img with ADetailer

```python
import webuiapi
from PIL import Image

img = Image.open("/path/to/your/image.jpg")

ads = webuiapi.ADetailer(ad_model="face_yolov8n.pt")

api = webuiapi.WebUIApi()

result1 = api.img2img(
    images=[img],
    prompt="a cute squirrel",
    steps=25,
    seed=-1,
    cfg_scale=7,
    denoising_strength=0.5,
    resize_mode=2,
    width=512,
    height=512,
    adetailer=[ads],
)

file_path = "img2img_output_image.png"
result1.image.save(file_path)
```

Support for interrogate with "deepdanbooru / deepbooru" (contributed by davidmartinrius)【译：支持使用“deepdanbooru / deepbooru”进行询问（由 davidmartinrius 提供）】

```python
import webuiapi
from PIL import Image

api = webuiapi.WebUIApi()

img = Image.open("/path/to/your/image.jpg")

interrogate_result = api.interrogate(image=img, model="deepdanbooru")
# also you can use clip. clip is set by default
#interrogate_result = api.interrogate(image=img, model="clip")
#interrogate_result = api.interrogate(image=img)

prompt = interrogate_result.info
prompt

# OR
print(prompt)
```

Support for ReActor, for face swapping (contributed by davidmartinrius)

【译：支持 ReActor，用于换脸（由 davidmartinrius 提供）】

```
import webuiapi
from PIL import Image

img = Image.open("/path/to/your/image.jpg")

api = webuiapi.WebUIApi()

your_desired_face = Image.open("/path/to/your/desired/face.jpeg")

reactor = webuiapi.ReActor(
    img=your_desired_face,
    enable=True
)

result1 = api.img2img(
    images=[img],
    prompt="a cute squirrel",
    steps=25,
    seed=-1,
    cfg_scale=7,
    denoising_strength=0.5,
    resize_mode=2,
    width=512,
    height=512,
    reactor=reactor
)

file_path = "face_swapped_image.png"
result1.image.save(file_path)
```

Support for Self Attention Guidance (contributed by yano)【译：支持自我注意力指导（由yano提供）】

https://github.com/ashen-sensored/sd_webui_SAG

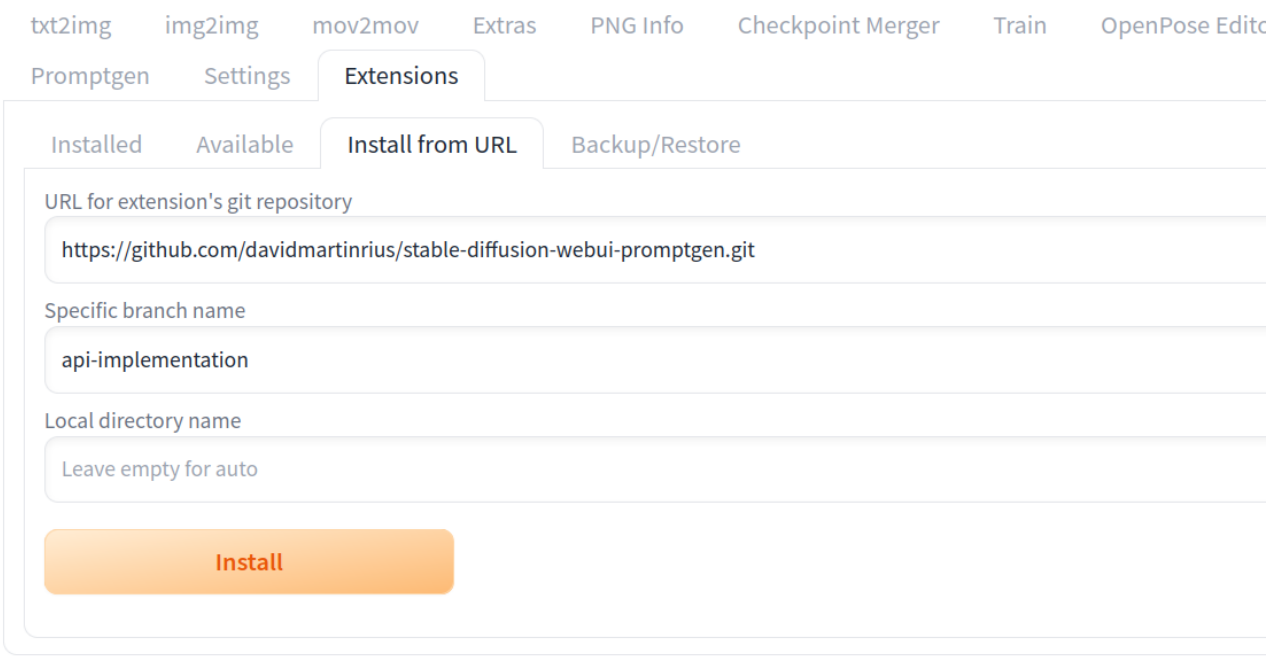
```
import webuiapi
from PIL import Image

img = Image.open("/path/to/your/image.jpg")

api = webuiapi.WebUIApi()

your_desired_face = Image.open("/path/to/your/desired/face.jpeg")

sag = webuiapi.Sag(
    enable=True,
    scale=0.75,
    mask_threshold=1.00
)

result1 = api.img2img(
    images=[img],
    prompt="a cute squirrel",
    steps=25,
    seed=-1,
    cfg_scale=7,
    denoising_strength=0.5,
    resize_mode=2,
    width=512,
    height=512,
    sag=sag
)

file_path = "face_swapped_image.png"
result1.image.save(file_path)
```

Prompt generator API by David Martin Rius:【译：David Martin Rius 的提示生成器 API：】

This is an unofficial implementation to use the api of promptgen. Before installing the extension you have to check if you already have an extension called Promptgen. If so, you need to uninstall it. Once uninstalled you can install it in two ways:

【译：这是使用 promptgen api 的非官方实现。 在安装扩展程序之前，您必须检查是否已经有一个名为 Promptgen 的扩展程序。如果是这样，您需要卸载它。 卸载后，您可以通过两种方式进行安装】

1. From the user interface【译：从用户界面】



2. From the command line【译：从命令行】

cd stable-diffusion-webui/extensions

git clone -b api-implementation https://github.com/davidmartinrius/stable-diffusion-webui-promptgen.git

Once installed:

```
api = webuiapi.WebUIApi()

result = api.list_prompt_gen_models()
print("list of models")
print(result)
# you will get something like this:
#['AUTOMATIC/promptgen-lexart', 'AUTOMATIC/promptgen-majinai-safe',
'AUTOMATIC/promptgen-majinai-unsafe']

text = "a box"

To create a prompt from a text:
# by default model_name is "AUTOMATIC/promptgen-lexart"
result = api.prompt_gen(text=text)

# Using a different model
result = api.prompt_gen(text=text, model_name="AUTOMATIC/promptgen-majinai-unsafe")

#Complete usage
result =  api.prompt_gen(
        text=text,
        model_name="AUTOMATIC/promptgen-majinai-unsafe",
        batch_count= 1,
        batch_size=10,
        min_length=20,
        max_length=150,
        num_beams=1,
        temperature=1,
        repetition_penalty=1,
        length_preference=1,
        sampling_mode="Top K",
        top_k=12,
        top_p=0.15
    )

# result is a list of prompts. You can iterate the list or just get the first result
like this: result[0]

```