内容管理平台

Table of contents

1. 概述

1.1 Using Python-Markdown on the Command Line

While Python-Markdown is primarily a python library, a command line script is included as well. While there are many oth
Markdown, you may not have them installed, or you may prefer to use Python-Markdown's various extensions.

Generally, you will want to have the Markdown library fully installed on your system to run the command line script. See t
details.

Python-Markdown's command line script takes advantage of Python's `-m` flag. Therefore, assuming the python executable
following format:

```
python -m markdown [options] [args]
```

That will run the module as a script with the options and arguments provided.

At its most basic usage, one would simply pass in a file name as the only argument:

```
python -m markdown input_file.txt
```

Piping input and output (on `STDIN` and `STDOUT`) is fully supported as well. For example:

```
echo "Some **Markdown** text." | python -m markdown > output.html
```

Use the `--help` option for a list all available options and arguments:

```
python -m markdown --help
```

If you don't want to call the python executable directly (using the `-m` flag), follow the instructions below to use a wrapper

1.1.1 Setup

Upon installation, the `markdown_py` script will have been copied to your Python "Scripts" directory. Different systems require d
files in the Python "Scripts" directory are on your system path.

- **Windows**:

  Assuming a default install of Python on Windows, your "Scripts" directory is most likely something like `C:\\Python26\Scri`
  "Scripts" directory and add it to you system path.

  Calling `markdown_py` from the command line will call the wrapper batch file `markdown_py.bat` in the `"Scripts"` directory created

- ***nix** (Linux, OSX, BSD, Unix, etc.):

  As each *nix distribution is different and we can't possibly document all of them here, we'll provide a few helpful po

  - Some systems will automatically install the script on your path. Try it and see if it works. Just run `markdown_py` fr
  - Other systems may maintain a separate "Scripts" ("bin") directory which you need to add to your path. Find it
    either add it to your path or make a symbolic link to it from your path.
  - If you are sure `markdown_py` is on your path, but it still is not being found, check the permissions of the file and m

  As an alternative, you could just `cd` into the directory which contains the source distribution, and run it from there.
  markdown text files will not likely be in that directory, so it is much more convenient to have `markdown_py` on your path.

!!!Note Python-Markdown uses `"markdown_py"` as a script name because the Perl implementation has already taken the more o
Additionally, the default Python configuration on some systems would cause a script named `"markdown.py"` to fail by importing
library. Therefore, the script has been named `"markdown_py"` as a compromise. If you prefer a different name for the script on y
create a symbolic link to `markdown_py` with your preferred name.

1.1.2 Usage

To use `markdown_py` from the command line, run it as

```
markdown_py input_file.txt
```

or

```
markdown_py input_file.txt > output_file.html
```

For a complete list of options, run

```
markdown_py --help
```

1.1.3 Using Extensions

To load a Python-Markdown extension from the command line use the `-x` (or `--extension`) option. The extension module must
Extension API for details). The extension can then be invoked by the name assigned to an entry point or using Python's do

For example, to load an extension with the assigned entry point name `myext`, run the following command:

```
python -m markdown -x myext input.txt
```

And to load an extension with Python's dot notation:

```
python -m markdown -x path.to.module:MyExtClass input.txt
```

To load multiple extensions, specify an `-x` option for each extension:

```
python -m markdown -x myext -x path.to.module:MyExtClass input.txt
```

If the extension supports configuration options (see the documentation for the extension you are using to determine what pass them in as well:

```
python -m markdown -x myext -c config.yml input.txt
```

The `-c` (or `--extension_configs`) option accepts a file name. The file must be in either the YAML or JSON format and contain YAM Python Dictionary in the format required by the `extension_configs` keyword of the `markdown.Markdown` class. Therefore, the file `config.y` might look like this:

```
myext:
    option1: 'value1'
    option2: True
```

Similarly, a JSON configuration file might look like this:

```
{
  "myext":
  {
    "option1": "value1",
    "option2": "value2"
  }
}
```

Note that while the `--extension_configs` option does specify the `myext` extension, you still need to load the extension with the `-x` extension will be ignored. Further, if an extension requires a value that cannot be parsed in JSON (for example a reference YAML configuration file.

The `--extension_configs` option will only support YAML configuration files if PyYAML is installed on your system. JSON should w The format of your configuration file is automatically detected.

1.2 生产环境升级

1.3 应急环境部署

1.4 灾备环境部署

1.5 测试环境部署

1.6 操作手册

1.7 运维手册

1.8 Windows生产环境部署

1.9 Linux独立环境部署

1.10 操作手册

1.11 运维手册

2. 第三方产品运维

2.1 GTP运维手册

2.2 Documentum运维手册

3. 常见问题

3.1 常见故障排除

3.2 自助故障排除

4. 关于

4.1 版本信息

4.2 技术支持