plugemin Documentation

Release 0.1.0

iLoveTux

Apr 17, 2017

Contents

1 Plugemin **1**
- 1.1 Getting Started . . . 1
- 1.2 Contributing . . . 3
- 1.3 Versioning . . . 3
- 1.4 Authors . . . 3
- 1.5 License . . . 3
- 1.6 Acknowledgments . . . 3

CHAPTER 1

Plugemin

plugemin is a simple utility which uses the amazing jinja2 templating engine and structured data such as XML, JSON or CSV to plug values into templates many times. Think of a form letter, but endlessly useful. With this you can render plain text, HTML, XML, JSON or even Python or C. Anything is possible as long as it is plain text (UTF is supported) in the end.

plugemin was initially designed to template commands which were to be sent over SSH to target systems.

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

Prerequisites

You will need the following software installed:

- Python >= 2.7
- jinja2
- lxml

lxml is the only one which needs a compiler to install. If this is an issue, I would recomend the great Anaconda Python Distribution which includes all the prerequisites installed by default.

Installing

You can install the latest stable version with the following command:

```
$ pip install plugemin
```

or for the latest development version, you can use the following command:

```
$ pip install git+https://github.com/ilovetux/plugemin.git
```

Running the tests

In order to run the tests, you will need to clone the repository and kick off the tests with a single command. All of that can be done with the following commands:

```
$ git clone https://github.com/ilovetux/plugemin
$ cd plugemin
$ python setup.py nosetests
```

That's it, the tests should pass, if they don't please open an issue and be sure to include:

- The commands you ran to get your results
- The versions of Python, lxml and jinja2 you have installed
- What Operating system
- Any details which would cause your setup to be considered non-standard such as running an obscure version of Linux

Basic Usage

Plugemin will look for templates in a series of locations and take a structured data format as input. It will render the template with each piece of data.

Example

in C:\plugemin\templates\backup-delete.j2:

```
cp {{src}} {{dst}}
rm {{src}}
```

in C:\tmp\files.csv:

```
src,dst
/var/log/*,/tmp/.
/usr/var/log/*,/tmp/.
/var/www/*,/tmp/.
```

Then you can use the following command:

```
C:\> plugemin -t backup-delete.j2 -d C:\plugemin\files.csv
cp /var/log/* /tmp/.
rm /var/log/*
cp /usr/var/log/* /tmp/.
rm /usr/var/log/*
cp /var/www/* /tmp/.
rm /var/www/*
```

Contributing

Please read CONTRIBUTING.rst for details on our code of conduct, and the process for submitting pull requests to us.

Versioning

We use SemVer for versioning. For the versions available, see the tags on this repository.

Authors

- iLoveTux

See also the list of contributors who participated in this project.

License

This project is licensed under the GPL Version 3 or later, please see the LICENSE file for details

Acknowledgments

- Hat tip to anyone who's code was used (Jinja2, lxml and Python)
- Brian Kearney and James Brennan for the inspiration to build this utility
- Anyone listed in the contributors file
- Everyone who helps us by submitting issues and pull requests

Readme

Plugemin

plugemin is a simple utility which uses the amazing jinja2 templating engine and structured data such as XML, JSON or CSV to plug values into templates many times. Think of a form letter, but endlessly useful. With this you can render plain text, HTML, XML, JSON or even Python or C. Anything is possible as long as it is plain text (UTF is supported) in the end.

plugemin was initially designed to template commands which were to be sent over SSH to target systems.

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

Prerequisites

You will need the following software installed:

- Python >= 2.7
- jinja2
- lxml

lxml is the only one which needs a compiler to install. If this is an issue, I would recomend the great Anaconda Python Distribution which includes all the prerequisites installed by default.

Installing

You can install the latest stable version with the following command:

```
$ pip install plugemin
```

or for the latest development version, you can use the following command:

```
$ pip install git+https://github.com/ilovetux/plugemin.git
```

Running the tests

In order to run the tests, you will need to clone the repository and kick off the tests with a single command. All of that can be done with the following commands:

```
$ git clone https://github.com/ilovetux/plugemin
$ cd plugemin
$ python setup.py nosetests
```

That's it, the tests should pass, if they don't please open an issue and be sure to include:

- The commands you ran to get your results
- The versions of Python, lxml and jinja2 you have installed
- What Operating system
- Any details which would cause your setup to be considered non-standard such as running an obscure version of Linux

Basic Usage

Plugemin will look for templates in a series of locations and take a structured data format as input. It will render the template with each piece of data.

Example

in C:\plugemin\templates\backup-delete.j2:

```
cp {{src}} {{dst}}
rm {{src}}
```

in C:\tmp\files.csv:

```
src,dst
/var/log/*,/tmp/.
/usr/var/log/*,/tmp/.
/var/www/*,/tmp/.
```

Then you can use the following command:

```
C:\> plugemin -t backup-delete.j2 -d C:\plugemin\files.csv
cp /var/log/* /tmp/.
rm /var/log/*
cp /usr/var/log/* /tmp/.
rm /usr/var/log/*
cp /var/www/* /tmp/.
rm /var/www/*
```

Contributing

Please read CONTRIBUTING.rst for details on our code of conduct, and the process for submitting pull requests to us.

Versioning

We use SemVer for versioning. For the versions available, see the tags on this repository.

Authors

- iLoveTux

See also the list of contributors who participated in this project.

License

This project is licensed under the GPL Version 3 or later, please see the LICENSE file for details

Acknowledgments

- Hat tip to anyone who's code was used (Jinja2, lxml and Python)
- Brian Kearney and James Brennan for the inspiration to build this utility
- Anyone listed in the contributors file
- Everyone who helps us by submitting issues and pull requests

CLI

This document describes the Command Line Interface to plugemin. plugemin can be used to populate and render templates based on data passed in.

Basically, it takes a template as an argument with an optional argument for the data (which defaults to stdin). The template will be rendered for each line of data.

The input format is parsed using plugins. The following plugins are included by default with plugemin:

- CsvInput: Reads a CSV, assumes that first row is a header
- JsonInput: Reads JSON, assumes that each line is a JSON document
- XmlInput: Reads XML, assumes that each line is an XML document and that there is just one level of children for the root node

Examples:

```
$ cat input.csv | plugemin -t template.j2
$ cat input.json | plugemin -t template.j2 -p JsonInput
$ cat input.xml | plugemin -t template.j2 -p XmlInput

$ plugemin -t template.j2 -d input.csv
$ plugemin -t template.j2 -d input.json -p JsonInput
$ plugemin -t template.j2 -d input.xml -p XmlInput
```

plugins

This document will describe how to use and create plugins for plugemin.

plugemin plugins are used to parse input data. The following are the plugins which come bundled with plugemin by default:

- CsvInput
- JsonInput
- XmlInput

CsvInput

The `CsvInput` parses a CSV file assuming the first line is a header.

Below is a simplified version of the CsvInput implementation:

```
class CsvInput(object):
    def __init__(self, data):
        if os.path.isfile(data):
            with open(data, "r") as fp:
                data = fp.readlines()
        self.parser = csv.DictReader(data)

    def __iter__(self):
        for row in self.parser:
            yield row
```

JsonInput

The `JsonInput` parses input where each line is a JSON document.

Below is a simplified version of the `JsonInput` plugin:

```
class JsonInput(object):
    def __init__(self, data):
        if os.path.isfile(data):
            with open(data, "r") as fp:
                data = fp.readlines()
        self.data = data

    def __iter__(self):
        for document in self.data:
            yield json.loads(document)
```

Creating your own plugins

In order to create a plugin for parsing data, you will simply make a `class` with an `__init__` method takes an argument called `data`. `data` is meant to be an iterator yielding lines for processing. The plugin should then yield a `dict` representing relevant values from each line.

We will make a plugin which takes lines of three space delimited values representing name, age and gender. We will then make two templates one which outputs a html page describing each person and another which outputs a friendly letter to each person. This demo is supposed to showcase some of the more useful features of plugemin and to help you get started using it.

Please consider the following directory structure:

```
.
- custom_input
|    - custom_input.py
|    - __init__.py
- test
|    - test_custom_input.py
- setup.py
```

Please consider the following code from `custom_input.py`:

```
class CustomInput(object):
    def __init__(self, data):
        self.data = data

    def __iter__(self):
        name, age, gender = line.split()
        for line in data:
            yield {"name": name, "age": int(age), "gender": gender}
```

That is all that is required for this plugin. With this, you can now ingest your custom data type and use the data to fill in the blanks on some templates.

Note: You would probably in real-world use, perform some error handling and validation. It would probably help to also define a couple of custom exceptions.

Next, we need to make a setup.py which will register our plugin with plugemin (the important piece is in the entry_points):

```
import sys
from setuptools import setup

tests_require = ["nose>=1.0"]
if sys.version_info < (3,0):
```

```
    tests_require = ["nose>=1.0", "mock"]

setup(
    name="plugemin-CustomPlugin",
    version="0.1.0",
    author="anon",
    author_email="anon@email.com",
    description="A plugemin plugin for parsing my custom data type",
    license="GPLv3",
    keywords="template reports",
    url="http://github.com/anon/plugemin-CustomPlugin",
    packages=['custom_input'],
    install_requires=["plugemin"],
    entry_points={
        "plugemin.InputPlugin": [
            "CustomInput=custom_plugin:CustomInput",
        ]
    },
    test_suite="nose.collector",
    tests_require=tests_require,
    classifiers=[
        "Development Status :: 4 - Beta",
        "Topic :: Utilities",
        "License :: OSI Approved :: GNU General Public License v3 (GPLv3)",
    ],
)
```

In order to make use of our plugin we will have to craft a template, we will draft two templates (described above).

First consider our sample data:

```
alice 23 female
bob 32 male
```

Please consider the html template below, we will call this `profile.html.j2`:

```
<div class="person" id="person-{{name}}">
    <h1>{{name}}</h1>
    <p>A {{gender}} of {{age}} years of age.</p>
</div>
```

Now, please consider the output of the following command:

```
$ plugemin -t profile.html.j2 -d data.txt

<div class="person" id="person-alice">
    <h1>alice</h1>
    <p>A female of 23 years of age.</p>
</div>

<div class="person" id="person-bob">
    <h1>bob</h1>
    <p>A male of 32 years of age.</p>
</div>
```

Let's try one more example template because it would be a shame if we could only do one thing with our data.

Please consider `hello-letter.j2`:

```
Dear {{name}}:

We appreciate you taking the time to complete our survey.

We have recorded that your name is {{name}}, your age is {{age}} and
your gender is {{gender}}.

If any of this is incorrect, please notify us at our email.
```

Now consider the following command:

```
$ plugemin -t hello-letter.j2 -d data.txt

Dear alice:

We appreciate you taking the time to complete our survey.

We have recorded that your name is alice, your age is 23 and
your gender is female.

If any of this is incorrect, please notify us at our email.

Dear bob:

We appreciate you taking the time to complete our survey.

We have recorded that your name is bob, your age is 32 and
your gender is male.

If any of this is incorrect, please notify us at our email.
```

Now for this to be useful, we would probably pipe the output which might send off the emails.

Thank you for taking the time to read our documentation, please check back as we will try to keep this documentation up to date.

API