Unicode in Flask

Flask, like Jinja2 and Werkzeug, is totally Unicode based when it comes to text. Not only these libraries, also the majority of web related Python libraries that deal with text. If you don't know Unicode so far, you should probably read The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets. This part of the documentation just tries to cover the very basics so that you have a pleasant experience with Unicode related things.

Automatic Conversion

Flask has a few assumptions about your application (which you can change of course) that give you basic and painless Unicode support:

- the encoding for text on your website is UTF-8
- internally you will always use Unicode exclusively for text except for literal strings with only ASCII character points.
- encoding and decoding happens whenever you are talking over a protocol that requires bytes to be transmitted.

So what does this mean to you?

HTTP is based on bytes. Not only the protocol, also the system used to address documents on servers (so called URIs or URLs). However HTML which is usually transmitted on top of HTTP supports a large variety of character sets and which ones are used, are transmitted in an HTTP header. To not make this too complex Flask just assumes that if you are sending Unicode out you want it to be UTF-8 encoded. Flask will do the encoding and setting of the appropriate headers for you.

The same is true if you are talking to databases with the help of SQLAlchemy or a similar ORM system. Some databases have a protocol that already transmits Unicode and if they do not, SQLAlchemy or your other ORM should take care of that.

The Golden Rule

So the rule of thumb: if you are not dealing with binary data, work with Unicode. What does working with Unicode in Python 2.x mean?

- as long as you are using ASCII code points only (basically numbers, some special characters of Latin letters without umlauts or anything fancy) you can use regul literals (`'Hello World'`).

- if you need anything else than ASCII in a string you have to mark this string as Unicode string by prefixing it with a lowercase *u*. (like `u'Hänsel und Gretel'`)
- if you are using non-Unicode characters in your Python files you have to tell Python which encoding your file uses. Again, I recommend UTF-8 for this purpose. To tell the interpreter your encoding you can put the `# -*- coding: utf-8 -*-` into the first or second line of your Python source file.
- Jinja is configured to decode the template files from UTF-8. So make sure to tell your editor to save the file as UTF-8 there as well.

Encoding and Decoding Yourself

If you are talking with a filesystem or something that is not really based on Unicode you will have to ensure that you decode properly when working with Unicode interface. So for example if you want to load a file on the filesystem and embed it into a Jinja2 template you will have to decode it from the encoding of that file. Here the old problem that text files do not specify their encoding comes into play. So do yourself a favour and limit yourself to UTF-8 for text files as well.

Anyways. To load such a file with Unicode you can use the built-in **`str.decode()`** method:

```
def read_file(filename, charset='utf-8'):
    with open(filename, 'r') as f:
        return f.read().decode(charset)
```

To go from Unicode into a specific charset such as UTF-8 you can use the **`unicode.encode()`** method:

```
def write_file(filename, contents, charset='utf-8'):
    with open(filename, 'w') as f:
        f.write(contents.encode(charset))
```

Configuring Editors

Most editors save as UTF-8 by default nowadays but in case your editor is not configured to do this you have to change it. Here some common ways to set your editor to store as UTF-8:

- Vim: put `set enc=utf-8` to your `.vimrc` file.
- Emacs: either use an encoding cookie or put this into your `.emacs` file:

```
(prefer-coding-system 'utf-8)
(setq default-buffer-file-coding-system 'utf-8)
```

- Notepad++:

  1. Go to *Settings -> Preferences ...*
  2. Select the “New Document/Default Directory” tab
  3. Select “UTF-8 without BOM” as encoding

  It is also recommended to use the Unix newline format, you can select it in the same panel but this is not a requirement.