Message Flashing

Good applications and user interfaces are all about feedback. If the user does not get enough feedback they will probably end up hating the application. Flask provides a really simple way to give feedback to a user with the flashing system. The flashing system basically makes it possible to record a message at the end of a request and access it next request and only next request. This is usually combined with a layout template that does this. Note that browsers and sometimes web servers enforce a limit on cookie sizes. This means that flashing messages that are too large for session cookies causes message flashing to fail silently.

Simple Flashing

So here is a full example:

```python
from flask import Flask, flash, redirect, render_template, \
     request, url_for

app = Flask(__name__)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != 'admin' or \
                request.form['password'] != 'secret':
            error = 'Invalid credentials'
        else:
            flash('You were successfully logged in')
            return redirect(url_for('index'))
    return render_template('login.html', error=error)
```

And here is the `layout.html` template which does the magic:

```html
<!doctype html>
<title>My Application</title>
{% with messages = get_flashed_messages() %}
  {% if messages %}
    <ul class=flashes>
```

```
      {% for message in messages %}
        <li>{{ message }}</li>
      {% endfor %}
      </ul>
    {% endif %}
  {% endwith %}
  {% block body %}{% endblock %}
```

Here is the `index.html` template which inherits from `layout.html`:

```
{% extends "layout.html" %}
{% block body %}
  <h1>Overview</h1>
  <p>Do you want to <a href="{{ url_for('login') }}">log in?</a>
{% endblock %}
```

And here is the `login.html` template which also inherits from `layout.html`:

```
{% extends "layout.html" %}
{% block body %}
  <h1>Login</h1>
  {% if error %}
    <p class=error><strong>Error:</strong> {{ error }}
  {% endif %}
  <form method=post>
    <dl>
      <dt>Username:
      <dd><input type=text name=username value="{{
          request.form.username }}">
      <dt>Password:
      <dd><input type=password name=password>
    </dl>
    <p><input type=submit value=Login>
  </form>
{% endblock %}
```

Flashing With Categories

▶ *Changelog*

It is also possible to provide categories when flashing a message. The default category if nothing is provided is `'message'`. Alternative categories can be used to give the user better feedback. For example error messages could be displayed with a red background.

To flash a message with a different category, just use the second argument to the **`flash()`** function:

```
flash(u'Invalid password provided', 'error')
```

Inside the template you then have to tell the **`get_flashed_messages()`** function to also return the categories. The loop looks slightly different in that situation then:

```
{% with messages = get_flashed_messages(with_categories=true) %}
  {% if messages %}
    <ul class=flashes>
    {% for category, message in messages %}
      <li class="{{ category }}">{{ message }}</li>
    {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
```

This is just one example of how to render these flashed messages. One might also use the category to add a prefix such as `<strong>Error:</strong>` to the message.

Filtering Flash Messages

▶ *Changelog*

Optionally you can pass a list of categories which filters the results of **`get_flashed_messages()`**. This is useful if you wish to render each category in a separate block.

```
{% with errors = get_flashed_messages(category_filter=["error"]) %}
{% if errors %}
<div class="alert-message block-message error">
  <a class="close" href="#">×</a>
  <ul>
    {%- for msg in errors %}
    <li>{{ msg }}</li>
    {% endfor -%}
  </ul>
</div>
{% endif %}
{% endwith %}
```