Forms and View Logic

Reindert-Jan Ekker
nl.linkedin.com/in/rjekker/
@rjekker

pluralsight
hardcore dev and IT training

In This Module

Forms

Extension: Flask-WTF

Form templates

Validating input

Handling logic

Views

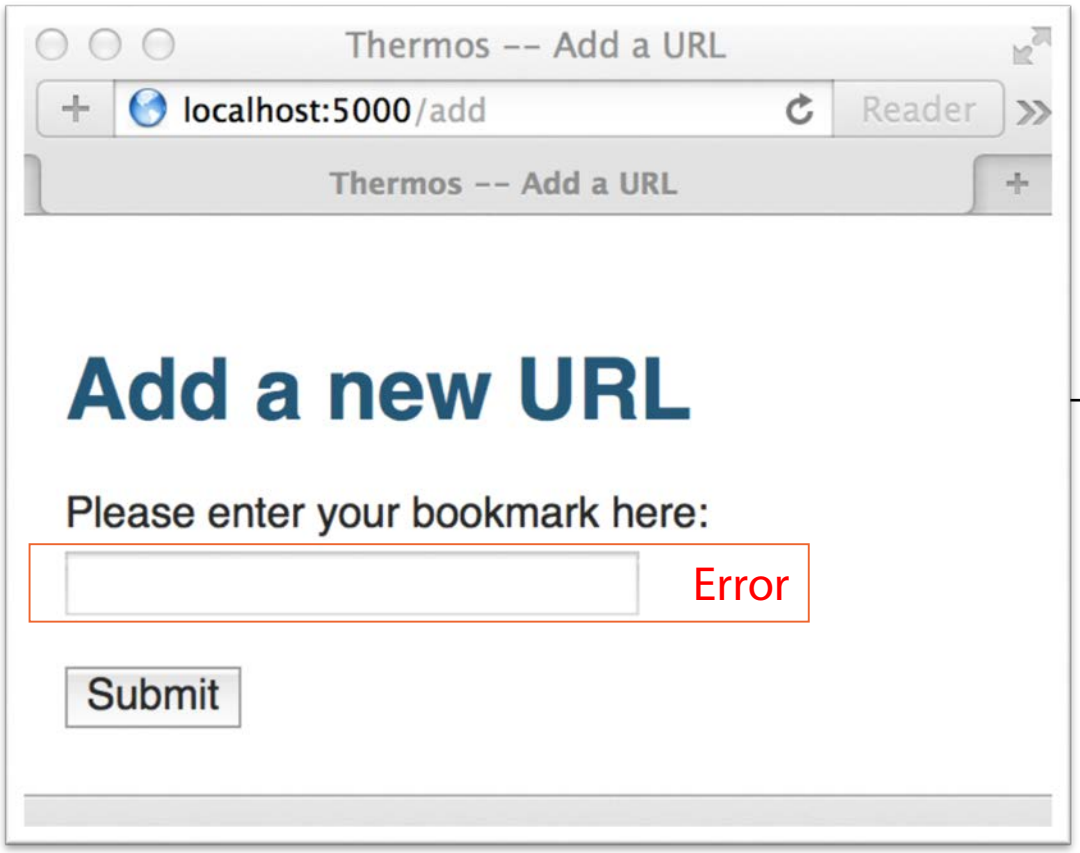
POST method

Redirection

Message flashing

Post-Redirect-Get

Form page view (**GET**)



`render_template()`

Yes

Submit (**POST**)

Validation error?

No

`redirect()`

Views and Forms

```
@app.route('/add', methods=['GET', 'POST'])
def add():
    if request.method == "POST":
        url = request.form['url']
        store_bookmark(url)
        return redirect(url_for('index'))
    return render_template('add.html')
```

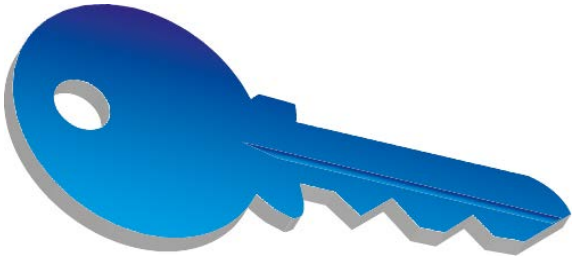
The request object

- **Globally available**
  - But temporarily bound to the *current* request
  - Don't use it when no request is active (outside view functions)
- **Some attributes of the request object:**

Name	Description
form	Form data from POST or PUT requests
args	Contents of the query string. (The part in the URL after the question mark)
cookies	Cookies transmitted with the request
headers	The incoming request headers as a dictionary like object
files	Files uploaded as part of a POST or PUT request
method	The current request method (POST, GET etc.)

Session and Flashes

- **Session object**
  - Remember data between requests
  - Works by setting a cookie
  - Data associatedwith the user's **HTTP Session**
  - Is a Flask context global, like `request`
- **Using the session**
  - Need to set `Flask.secret_key` for creating cookies
  - Store values in it like a dict
- **Flashing messages**
  - Use `flash()`
  - Available in template through `get_flashed_messages()`



Jinja: If, For, With

```
{% if expression %} … {% else %} … {% endif %}
```

Can also use `{% elif %}`

```
{% for var in expression %} … {% endfor %}
```

Jinja provides a `loop` variable inside `for`

```
{% with var = expression %} … {% endwith %}
```

- Flask extension
- WTForms integration
- Render forms
- Validate forms

Flask-WTF Form Classes

- **Simple Python class inheriting from `flask_wtf.Form`**

- **Fields**
  - Many field classes for different kinds of input
  - StringField, DateField, BooleanField, SelectField, HiddenField, etc.
  - Note: flask_wtf includes HTML5 forms in `flask.ext.wtf.html5`

- **Pass a list of validators to the field**
  - `url=URLField('url',validators=[DataRequired(), url()])`

WTForms in Views

- **Create a form instance**
  - `form = BookmarkForm()`
  - This will be filled with any data from the request, if any
- **Check: was the form submitted and does it validate?**
  - `form.validate_on_submit()`
- **Get data from fields**
  - `url = form.url.data`
- **Make sure to pass the form to the template**
  - `render_template('add.html', form=form)`

WTForms in Templates

- **Don't forget CSRF protection!**
  - `{{ form.hidden_tag() }}`
  - Need a secret key
- **Render fields**
  - `{{ form.url }}`
  - Keyword arguments will become HTML attributes
  - `{{ form.url(class="fancy") }}`
- **Errors**
  - `{% for error in form.url.errors %}`

Resources

- http://jinja.pocoo.org/docs/dev/templates/#list-of-control-structures (http://goo.gl/UFmbSr)
- https://flask-wtf.readthedocs.org (http://goo.gl/EW1hdn)
- http://wtforms.readthedocs.org/en/latest/index.html
  - (http://goo.gl/ulTg3Z)
- https://en.wikipedia.org/wiki/Post/Redirect/Get
  - (http://goo.gl/yjhW2l)

Summary

- **Forms with Flask-WTF**
- **Views**
  - Post/Redirect/Get
  - Message Flashing
  - Secret Key
- **Templates**
  - If, For, With
  - Rendering form fields and errors
  - CSRF protection
  - Macros
- **Custom Validation**