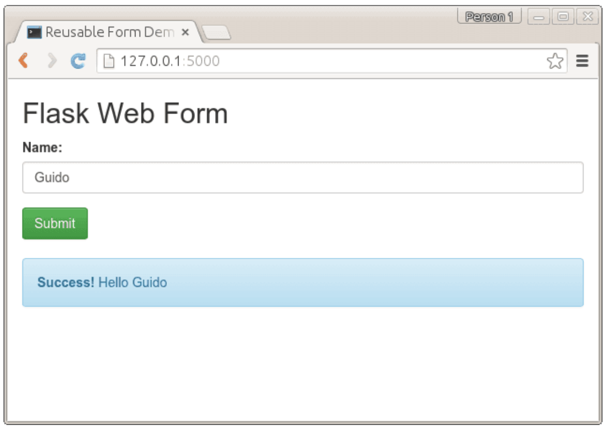
Flask web forms

tableau Storytelling with Data GET THE WHITEPAPER

In this tutorial you will learn how to do form validation with Flask. Forms play an important role in all web applications.



Flask web form

We use WTForms, a module for validation of forms. We will start with a simple form containing one field asking for a name.

Related course:
[Complete Python Web Course: Build 8 Python Web Apps]

Flask web forms

Ads help us run this site. When you use our site selected companies may access and use information on your device for various purposes including to serve relevant ads or personalised content.

ACCEPT ALL

Read more about access and use of information on your device for various purposes

```
class Reusab
    name = TextField('Name:', validators=[validators.required()


@app.route("/", methods=['GET', 'POST'])
def hello():
    form = ReusableForm(request.form)

    print form.errors
    if request.method == 'POST':
        name=request.form['name']
        print name

        if form.validate():
            # Save the comment here.
            flash('Hello ' + name)
        else:
            flash('All the form fields are required. ')

    return render_template('hello.html', form=form)

if __name__ == "__main__":
    app.run()
```

We then create the template hello.html in the /templates/ directory:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http:
<html>
    <head>
        <title>Reusable Form Demo</title>
    </head>
    <body>
        {% with messages = get_flashed_messages(with_categories
            {% if messages %}
                <ul>
                    {% for message in messages %}
                        <li>{{ message[1] }}</li>
                    {% endfor %}
                </ul>
            {% endif %}
        {% endwith %}
        <form action="" method="post">
            {{ form.csrf }}

            <div class="input text">
                {{ form.name.label }} {{ form.name }}
```

LEARN MORE

LEARN MORE

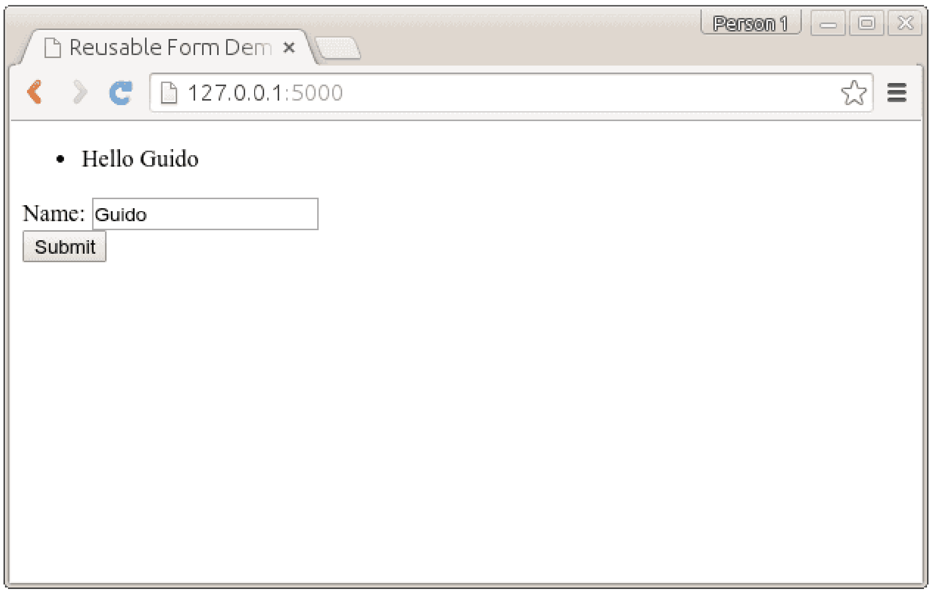
Start the application and open it in your webbrowser at http://127.0.0.1:5000.

```
python miniapp.py
 * Running on http://127.0.0.1:5000/
 * Restarting with reloader
```

If you will submit an empty form, you will get an error. But if you enter your name, it will greet you. Form validation is handled by the wtforms module, but because we gave the argument

```
name = TextField('Name:', validators=[validators.required()])
```

the field cannot be empty. Other parameters can be given here.



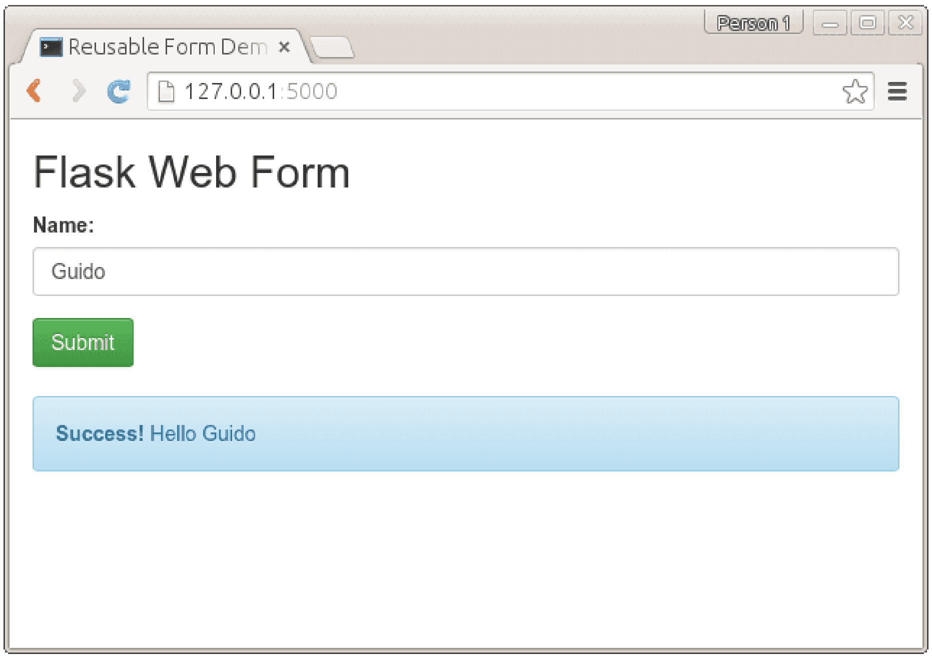
- Hello Guido

Name: Guido
Submit

Python wtforms, text field validation. The app returns the name if entered

css with Flask

We use bootstrap to style the form.Bootstrap is a popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. It makes front-end web development faster and easier. The output will be:



Flask Web Form

Name:

Guido

Submit

Success! Hello Guido

Flask wtforms

You can get the bootstrap files from http://getbootstrap.com/getting-started/#download and extract them in a new directory /static/. The code remains almost the same, but the template is changed. Code:

```
from flask import Flask, render_template, flash, request
from wtforms import Form, TextField, TextAreaField, validators,

# App config.
DEBUG = True
app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

class ReusableForm(Form):
    name = TextField('Name:', validators=[validators.required()


@app.route("/", methods=['GET', 'POST'])
def hello():
    form = ReusableForm(request.form)

    print form.errors
    if request.method == 'POST':
        name=request.form['name']
        print name

        if form.validate():
            # Save the comment here.
            flash('Hello ' + name)
        else:
            flash('Error: All the form fields are required. ')

    return render_template('hello.html', form=form)
```

```
if __name__ == "__main__":
    app.run()
```

We added bootstrap to the template hello.html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http:
<html>
    <head>
        <title>Reusable Form Demo</title>
       <link rel="stylesheet" media="screen" href ="static/boot
       <link rel="stylesheet" href="static/bootstrap-theme.min.
       <meta name="viewport" content = "width=device-width, ini

    </head>
    <body>


<div class="container">


  <h2>Flask Web Form</h2>
  <form  action="" method="post" role="form">
    {{ form.csrf }}
    <div class="form-group">
      <label for="name">Name:</label>
      <input type="text" class="form-control" id="name" name="n
    </div>
    <button type="submit" class="btn btn-success">Submit</butto
  </form>

  <br>
        {% with messages = get_flashed_messages(with_categories
            {% if messages %}

        {% for message in messages %}
            {% if "Error" not in message[1]: %}
                <div class="alert alert-info">
                <strong>Success! </strong> {{ message[1] }}
                </div>
            {% endif %}

            {% if "Error" in message[1]: %}
                <div class="alert alert-warning">
                {{ message[1] }}
                </div>
            {% endif %}
        {% endfor %}
            {% endif %}
        {% endwith %}

</div>
<br>
</div>
</div>
</body>
</html>
```

Flask registration form

We use the same principle to create a registration form asking for name, email and password. We update the Form class:

```
class ReusableForm(Form):
    name = TextField('Name:', validators=[validators.required()
    email = TextField('Email:', validators=[validators.required
    password = TextField('Password:', validators=[validators.re

    def reset(self):
        blankData = MultiDict([ ('csrf', self.reset_csrf() ) ])
        self.process(blankData)
```

And we can get the variables passed using:

```
        name=request.form['name']
        password=request.form['password']
        email=request.form['email']
```

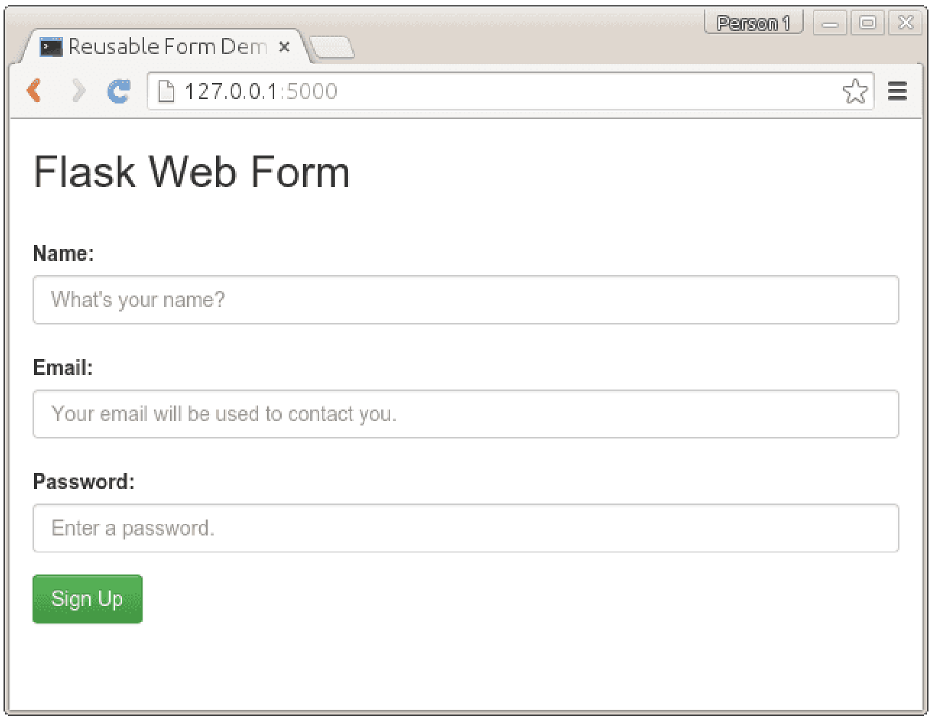
Full code:

```
from flask import Flask, render_template, flash, request
from wtforms import Form, TextField, TextAreaField, validators,

# App config.
DEBUG = True
app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

class ReusableForm(Form):
    name = TextField('Name:', validators=[validators.required()
    email = TextField('Email:', validators=[validators.required
    password = TextField('Password:', validators=[validators.re


@app.route("/", methods=['GET', 'POST'])
def hello():
    form = ReusableForm(request.form)

    print form.errors
    if request.method == 'POST':
        name=request.form['name']
        password=request.form['password']
        email=request.form['email']
        print name, " ", email, " ", password

        if form.validate():
            # Save the comment here.
            flash('Thanks for registration ' + name)
        else:
            flash('Error: All the form fields are required. ')

    return render_template('hello.html', form=form)

if __name__ == "__main__":
    app.run()
```

Update the template hello.html with this code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http:
<html>
    <head>
        <title>Reusable Form Demo</title>
       <link rel="stylesheet" media="screen" href ="static/boot
       <link rel="stylesheet" href="static/bootstrap-theme.min.
       <meta name="viewport" content = "width=device-width, ini

    </head>
    <body>


<div class="container">


  <h2>Flask Web Form</h2>
  <form  action="" method="post" role="form">
    {{ form.csrf }}
    <div class="form-group">
      <label for="name">Name:</label>
      <input type="text" class="form-control" id="name" name="n
      <br>
      <label for="email">Email:</label>
      <input type="text" class="form-control" id="email" name="
      <br>
      <label for="password">Password:</label>
      <input type="password" class="form-control" id="password"


    </div>
    <button type="submit" class="btn btn-success">Sign Up</butt
  </form>

  <br>
        {% with messages = get_flashed_messages(with_categories
            {% if messages %}

        {% for message in messages %}
            {% if "Error" not in message[1]: %}
                <div class="alert alert-info">
                <strong>Success! </strong> {{ message[1] }}
                </div>
            {% endif %}

            {% if "Error" in message[1]: %}
                <div class="alert alert-warning">
                {{ message[1] }}
                </div>
            {% endif %}
        {% endfor %}
            {% endif %}
        {% endwith %}

</div>
<br>
</div>
</div>
</body>
</html>
```

Output:



Flask Web Form

Name:
What's your name?

Email:
Your email will be used to contact you.

Password:
Enter a password.

flask form bootstrap

Download Flask Examples

WTForms can validate email, password, numbers and many more. For a list of validators see: http://wtforms.readthedocs.org/en/latest/validators.html

« Back

Next »

Posted in Flask

bootstrap flask form wtforms

5 THOUGHTS ON “FLASK WEB FORMS”

Log in to Reply

Connor - September 7, 2015

In the first code example, where are you getting the request object from? Also, what are you doing when you are passing “methods=[‘GET’, ‘POST’]” to app.route? Im a newbie to flask so please, explain it like im 5. 😁 thanks for the awesome tutorials!

Log in to Reply

Frank - September 7, 2015

The request object is imported from flask on top.

Your computer communicates with a special computer called a server. This server sends you data such as this website. Both computers need to ‘speak’ a language to communicate, this language is called the HTTP protocol. The language is very simple, there are only a few words such as GET and POST.

A client (webbrowser) can send the “GET” word to the server, it will return the file you request. To give data to the server, you use the “POST” command.
When you opened this page, your web browser did the GET command. When you posted this comment, your browser did a POST command.

Flask is server side software. Python needs a way to connect the incoming HTTP commands (URLs) to functions, in this case "/", but may as well be "/connor". This process is called routing.

In the first example we define a route @app.route("/"), and map it to a function for both GET and POST commands. If a GET is send, the function returns a screen with the text input. If a POST is send, another output is shown.

Log in to Reply

Connor - September 8, 2015

Thanks!

Log in to Reply

Duckpearl - July 19, 2015

Could you explain a bit more about the following line of code?

def reset(self):
blankData = MultiDict([('csrf', self.reset_csrf())])
self.process(blankData)

I can't see where it is used and I have never come across it in wtforms comments before.
Thanks

Log in to Reply

Frank - July 19, 2015

Hi, thanks for the comment! This method is not used now, you can remove it. For CSRF protection see: http://flask.pocoo.org/snippets/3/

LEAVE A REPLY

You must be logged in to post a comment.

Copyright © 2017 Pythonspot | Cookie policy | Terms of use | Privacy policy