NOTES

Bachelor of Science (Second Year)
CSS104- Web Application Development Using Flask
Title of the Unit : Flask Extensions
Module Name : Flask Extensions, Installing flask-wtf extension, Flask WTF
Module Number : 21

Flask – Extensions

Flask is often referred to as a micro framework, because a core functionality includes WSGI and routing based on **Werkzeug** and template engine based on **Jinja2**.
In addition, Flask framework has support for cookie and sessions as well as web helpers like **JSON**, static files etc.
Obviously, this is not enough for the development of a full-fledged web application. This is where the Flask extensions come in picture. Flask extensions give extensibility to Flask framework.
There are a large number of Flask extensions available. A Flask extension is a Python module, which adds specific type of support to the Flask application. The required extension can be downloaded by **pip** utility.

Following are the Flask extensions :−

- **Flask WTF** − adds rendering and validation of WTForms
- **Flask SQLAlchemy** − adds SQLAlchemy support to Flask application

Installing flask-wtf extension

First, Flask-WTF extension needs to be installed.
pip install flask-WTF
The installed package contains a **Form** class, which has to be used as a parent for user-defined form.

Flask – WTF

Disadvantages of HTML <form>

- The Server side script has to recreate the form elements from http request data. So in effect, form elements have to be defined twice – once in HTML and again in the server side script.
- Another disadvantage of using HTML form is that it is difficult (if not impossible) to render the form elements dynamically. HTML itself provides no way to validate a user's input.

This is where WTForms, a flexible form, rendering and validation library comes handy. Flask-WTF extension provides a simple interface with this WTForms library.

Using **Flask-WTF**, we can define the form fields in our Python script and render them using an HTML template. It is also possible to apply validation to the **WTF** field.

WTforms package contains definitions of various form fields. **Some Standard form fields are listed below.**

Sr.No	Standard Form Fields & Description
1	**TextField** Represents <input type = 'text'> HTML form element
2	**BooleanField** Represents <input type = 'checkbox'> HTML form element
3	**DecimalField** Textfield for displaying number with decimals
4	**IntegerField** TextField for displaying integer
5	**RadioField** Represents <input type = 'radio'> HTML form element
6	**SelectField** Represents select form element
7	**TextAreaField** Represents <testarea> html form element
8	**PasswordField** Represents <input type = 'password'> HTML form element
9	**SubmitField** Represents <input type = 'submit'> form element

For example, a form containing a text field can be designed as below −

```
from flask_wtf import Form
from wtforms import TextField

class ContactForm(Form):
   name = TextField("Name Of Student")
```

In addition to the **'name'** field, a hidden field for CSRF token is created automatically. This is to prevent **Cross Site Request Forgery** attack.

When rendered, this will result into an equivalent HTML script as shown below.

```
<input id = "csrf_token" name = "csrf_token" type = "hidden" />
<label for = "name">Name Of Student</label><br>
<input id = "name" name = "name" type = "text" value = "" />
```

A user-defined form class is used in a Flask application and the form is rendered using a template.

```
from flask import Flask, render_template
```

```
from forms import ContactForm
app = Flask(__name__)
app.secret_key = 'development key'

@app.route('/contact')
def contact():
   form = ContactForm()
   return render_template('contact.html', form = form)

if __name__ == '__main__':
   app.run(debug = True)
```

WTForms package also contains validator class. It is useful in applying validation to form fields.

Following list shows commonly used validators.

Sr.No	Validators Class & Description
1	**DataRequired** Checks whether input field is empty
2	**Email** Checks whether text in the field follows email ID conventions
3	**IPAddress** Validates IP address in input field
4	**Length** Verifies if length of string in input field is in given range
5	**NumberRange** Validates a number in input field within given range
6	**URL** Validates URL entered in input field

We shall now apply **'DataRequired'** validation rule for the **name** field in contact form.

```
name = TextField("Name Of Student",[validators.Required("Please enter your name.")])
```

The **validate()** function of form object validates the form data and throws the validation errors if validation fails. The **Error** messages are sent to the template. In the HTML template, error messages are rendered dynamically.

```
{% for message in form.name.errors %}
   {{ message }}
{% endfor %}
```