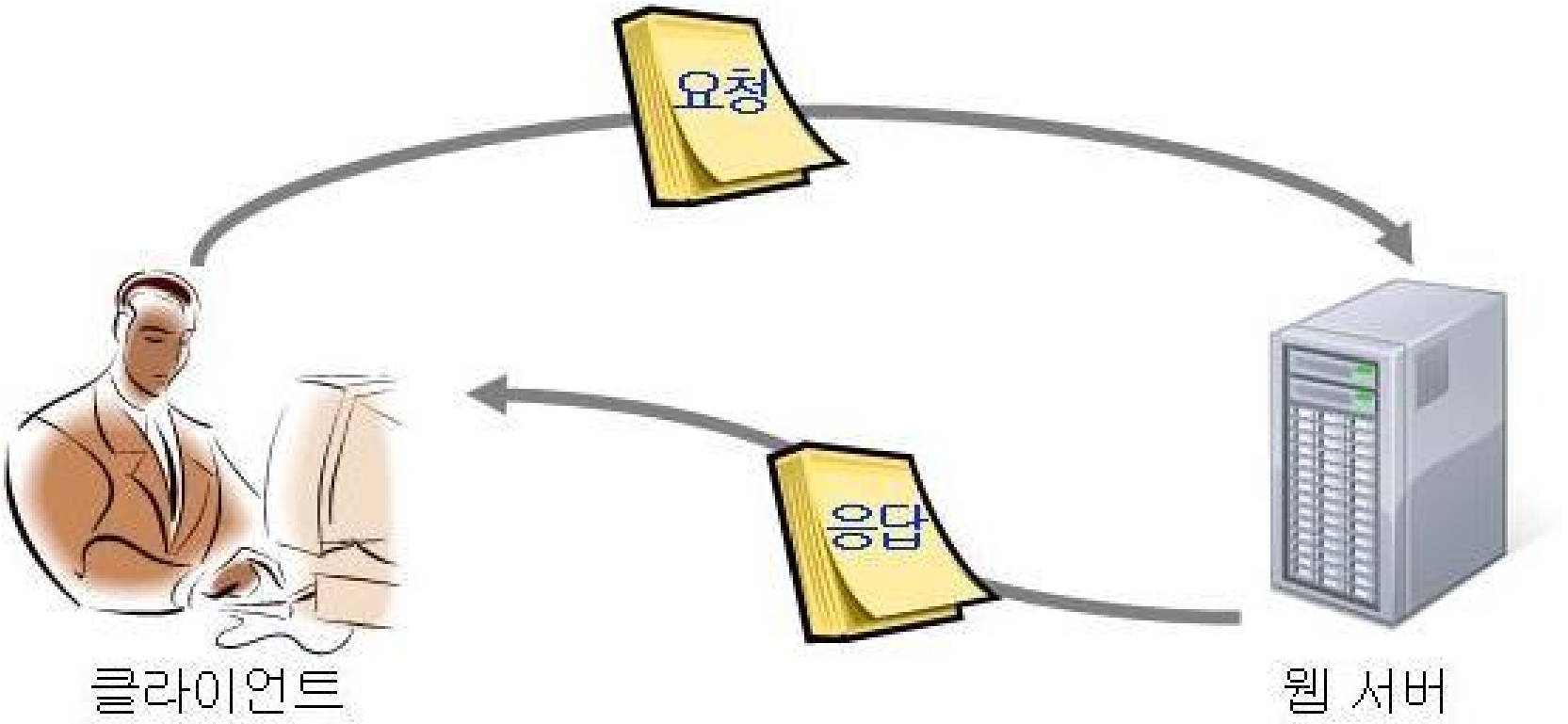
Python Web

Flask

웹 기반 애플리케이션 - WSGI(Web server Gateway Interface)



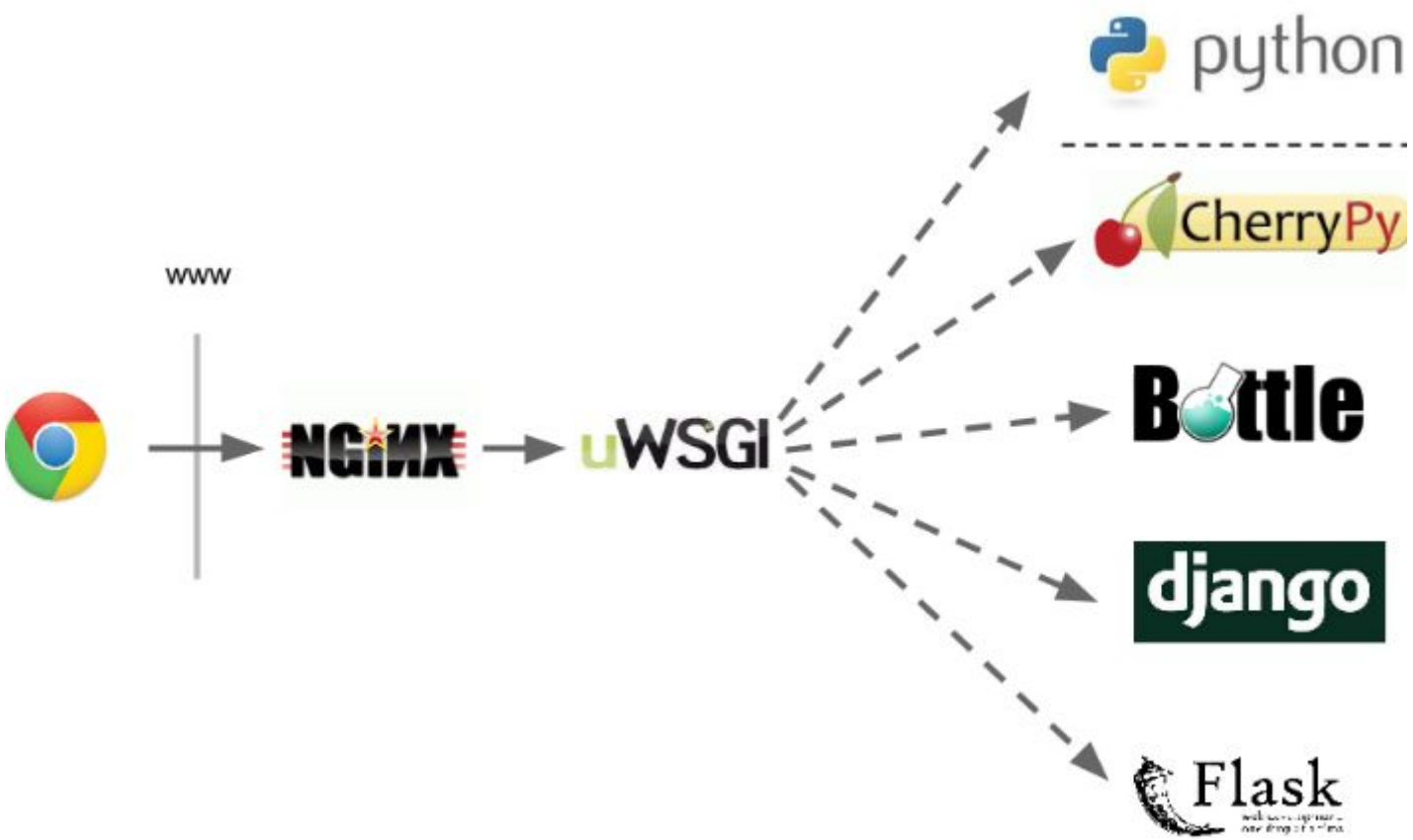
Web Framework

- Framework

  “ 소프트웨어의 구체적인 부분에 해당하는 설계와 구현을 재사용이 가능하게끔 일련의 협업화된 형태로 클래스들을 제공하는 것 - Ralph Johnson ”

  - ➢ 구체적이며 확장 가능한 기반 코드를 가지고 있으며, 설계자가 의도하는 여러 디자인 패턴 집합 구성
  - ➢ 라이브러리와 달리 애플리케이션 틀과 구조 결정, 개발된 개발자 코드 제어

- Python Web Framework



Flask

- Web Framework

  ~$ apt-get install python3-pip

  ~$ pip3 install flask

  >>> import flask

  >>> flask.__version__
- 의존성
  - ➢ Werkzeug : 라우팅, 디버깅, WSGI(Web Server Gateway Interface)
  - ➢ Jinja2 : 템플릿
- Flask Extension Registry : http://flask.pocoo.org/extensions/

  ex) flask-script

  ~$ pip3 install flask-script

  ~$ python3 -i hello.py runserver --host 0.0.0.0
- App context processor Flask

Hello World

- hello.py

```
from flask import Flask
app = Flask(__name__)  # flask 생성자
@app.route('/')                # URL 매핑
def index():                   # View 함수
    return '<h1>Hello World!</h1>'    # response

if __name__ == '__main__':
    app.run(debug=False)   # Server Startup
```

- kill Web Server

```
~$ lsof -i :port
~$ kill -9 <pid>       or   sudo kill $(sudo lsof -t -i:8000)
```

- Web Server

```
~$ python3 hello.py
```

URL Binding(Route with Decorator)

❖ 실행결과

http://localhost:5000/user/admin

http://localhost:5000/user/Yojulab

❖ 따라하기

```
from flask import Flask, redirect, url_for
app = Flask(__name__)
@app.route('/admin')
def hello_admin():
   return 'Hello Admin'
@app.route('/guest/<guest>')                    → int, float 가능.
def hello_guest(guest):
   return 'Hello ‘+guest+ ‘ as Guest'
@app.route('/user/<name>')
def hello_user(name):
   if name =='admin':
      return redirect(url_for('hello_admin'))
   else:
      return redirect(url_for('hello_guest',guest = name))
if __name__ == '__main__':
   app.run(debug = True)
```

Request

- Request Dispatch : URL Map 사용해 요청한 URL과 View 함수 매칭

  >>> from hello import app

  >>> app.url_map
- Request hooks : View 함수 작업 전후 처리, 공통 함수로 중복 줄임.
  - before_first_request : 첫 번째 처리 전 실행
  - before_request : 각 View 함수 전 실행
  - after_request : 예외 발생 없으면 각 View 함수 후 실행
  - teardown_request : 예외 발생해도 각 View 함수 후 실행
- HTTP method
  - GET : Sends data in unencrypted form to the server. Most common method.
  - HEAD : Same as GET, but without response body
  - POST : Used to send HTML form data to server. Data received by POST method is not cached by server.
  - PUT : Replaces all current representations of the target resource with the uploaded content.
  - DELETE : Removes all current representations of the target resource given by a URL

Try - HTTP method

❖ 실행결과

http://localhost:5000/httpmethod_submit.html → Post

http://localhost:5000/httpmethod?myname=Yojulab → Get

❖ 따라하기

```
// httpmethod_submit.html
    <form action = "http://localhost:5000/httpmethod" method = "post">
      <p><input type = "text" name = "myname"/></p>
       <p><input type = "submit" value = "submit" /></p>
    </form>
// httpmethod.py
     app = Flask(__name__)
     @app.route('/httpmethod_success/<myname>')
     def httpmethod_success(myname):
        return 'welcome' + myname
     @app.route('/httpmethod',methods = ['POST', 'GET'])
     def httpmethod():
        if request.method == 'POST':      user = request.form['myname']
        else: user = request.args.get('myname')
        return redirect(url_for('httpmethod_success',myname = user))
```

Try - Template

❖ 실행결과

http://localhost:5000/template_base/Yojulab

❖ 따라하기

```
// templates/template_base.html
    <h1>Hello {{ myname }}!</h1>
// template_base.py
    @app.route('/template_base/<user>')
    def hello_name(user):
      return render_template('template_base.html', myname = user)
```

❖ static

root/static/*

ex) /static/images/tree.png

/static/js/bootstrap.js

/static/css/bootstrap.css

Jinja2(1) - Template

- Expressions : {{ ? }}
  - Complex Type 인정 : list, dictionary, object misc

    ex) value from dictionary : {{ mydict['key'] }}
     value from list : {{ mylist[3] }} {{ mylist[myintvar] }}
     value from object : {{ myobj.somemethod() }}
  - Filter 사용해 수정 가능
    - safe-이스케이프 비적용, capitalize-첫 문자 대문자
    - lower - 소문자, trim-공백삭제, striptags-HTML tag 제거

    ex) Hello, {{ mylist[3] | capitalize }}
    <h1>Hello<h1> → <h1>Hello</h1>
- Statements
  - if else

    ex) {% if user %} Hello, {{ user }}
    {% else %} Hello, Stranger!
    {% endif %}
  - for

    ex) {% for comment in comments %}
    {{ comment }}
    {% endfor %}
- Comments : {# ... #}, # ... ##

Try - Template if 구현과 이해

❖ 실행결과

http://localhost:5000/template_if/60

❖ 따라하기

// **templates**/template_if.html

```
{% if marks>50 %}
<h1> Your result is pass!</h1>
{% else %}
<h1>Your result is fail</h1>
{% endif %}
```

// template_if.py

```
@app.route('/template_if/<int:score>')
  def template_if(score):
    return render_template('template_if.html', marks = score)
```

Try - Template for 구현과 이해

❖ 실행결과

http://localhost:5000/template_for

❖ 따라하기

// **templates**/template_for.html

```
<table border = 1>
  {% for key, value in result.items() %}
    <tr>
      <th> {{ key }} </th>     <td> {{ value }} </td>
    </tr>
  {% endfor %}
</table>
```

// template_for.py

```
@app.route('/template_for')
def result():
   dict = {'phy':50,'che':60,'maths':70}
   return render_template('template_for.html', result = dict)
```

Try - Sending Form Data to Template 구현과 이해

❖ 실행결과 : http://localhost:5000/sendingform

❖ 따라하기

```
// templates/sendingform.html
    <form action = "http://localhost:5000/result" method = "POST">
        <p>Name <input type = "text" name = "name" /></p>
        <p>Birthday <input type = "text" name = "birthday" /></p>
        <p><input type = "submit" value = "submit" /></p>        </form>
// templates/result.html
    <table border = 1>
        {% for key, value in result.items() %}
          <tr><th> {{ key }} </th>   <td> {{ value }} </td></tr>
        {% endfor %}        </table>
// template_for.py
 @app.route('/sendingform')
 def sendingform():
   return render_template(sendingform.html')
 @app.route('/result',methods = ['POST', 'GET'])
 def result():
   if request.method == 'POST':
      result = request.form;     return render_template("result.html",result = result)
```

Jinja2(2) - Template

- macro : {% macro render_comment(comment) %} … {% endmacro %}
- import : {% import 'macros.html' as macros %}
- block tag : 템플릿 상속
  - base.html

```
<html>
<head>
    {% block title %}    {% endblock %}
</head>
<body>
    {% block body %}
    {% endblock %}
</body>
</html>
```

  - target.html

```
{% extends 'base.html' %}
{% block title %} Main Page {% endblock %}
…
```

Try - Deep Template 구현과 이해

```
// templates/base.html
    <html>
    <head>
        {% block head %}
        <title>{% block title %}{% endblock %} - My App</title>
        {% endblock %}
    </head>
    <body>
        {% block body %} {% endblock %}
    </body>
    </html>
// templates/index.html
    {% extends "base.html" %}
    {% block title %}Index{% endblock %}
    {% block head %}
        {{ super() }}
    {% endblock %}
    {% block body %}
    <h1>Hello World!</h1>
    {% endblock %}
```

URL link

❖ 동적 Link

ex) url_for('index') : return '/'

url_for('index', _external=True) : return http://localhost:5000

url_for('index', page=2) : return /?page=2

❖ 정적 Link : /static/<filename>

ex) // templates/base.html

{% block head %}

{{ super() }}

<link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}" type="image/x-icon">

{% endblock %}

❖ 날짜와 시간 지역화 : UTC(Coordinated Universal Time)

~$ pip3 install flask-moment

ex) from flask_moment import Moment

moment = Moment(app)

def index():

return render_template('index.html', current_time=datetime.utcnow())

// index.html

The local date is {{ moment(current_time).format('LLL') }}.

Try - static 구현과 이해

```
// static/static.js
      function sayHello() {
         alert("Hello World")
      }
// templates/static.html
   </head>
      <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}">
      <script type = "text/javascript" src = "{{ url_for('static', filename = 'static.js') }}">
      </script>
   </head>
   <body>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
   </body>
// static.py
from flask import Flask, render_template
      @app.route("/static")
      def index():
         return render_template("static.html")
```

Try - Request Object(Form) 구현과 이해

```
// templates/request_form_submit.html
      <form action = "/request_form_result" method = "POST">
        <p>Name <input type = "text" name = "Name" /></p>    ...
        <p>Chemistry <input type = "text" name = "chemistry" /></p>
        <p><input type = "submit" value = "submit" /></p>
      </form>
// templates/request_form_result.html
      <table border = 1>
        {% for key, value in result.items() %}
          <tr> <th> {{ key }} </th> <td> {{ value }} </td> </tr>
        {% endfor %}
      </table>
// request_form.py
      @app.route('/request_form_submit')
      def request_form_submit():
          return render_template('request_form_submit.html')
      @app.route('/request_form_result',methods = ['POST', 'GET'])
      def request_form_result():
        if request.method == 'POST':
          result = request.form
          return render_template("/request_form_result.html",result = result)
```

Try - Request Object(Cookies) 구현과 이해

❖ 따라해보기 : inspect(on Chrome) > Application > Cookies

```
// templates/request_cookies_submit.html
    <form action = "/request_cookies_set" method = "POST">
        <p><input type = 'text' name = 'myname'/></p>
       <p><input type = 'submit' value = 'Login'/></p>
    </form>
// request_cookies.py
    @app.route('/request_cookies_submit')
    def request_cookies_submit():
        return render_template('request_cookies_submit.html')
    @app.route('/request_cookies_set',methods = ['POST', 'GET'])
    def request_cookies_set():
        if request.method == 'POST':
            user = request.form['myname']
            resp = make_response(render_template('request_cookies_read.html'))
            resp.set_cookie('userID', user)
        return resp
    @app.route('/request_cookies_get')
    def request_cookies_get():
      myname = request.cookies.get('userID')
      return '<h1>welcome '+myname+'</h1>'
```

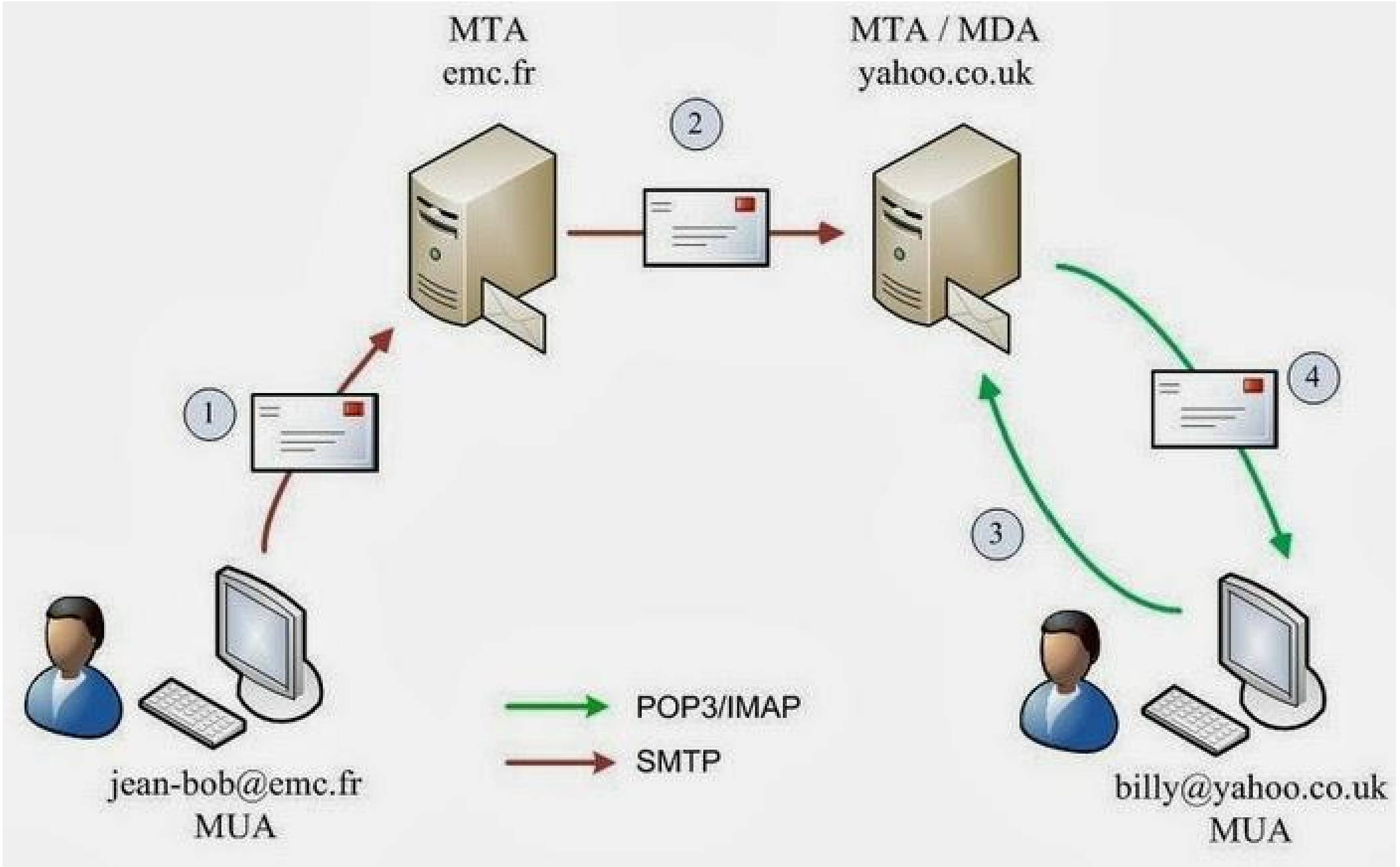
Try - Request Object(Sessions) 구현과 이해

```
// templates/request_sessions_submit.html
  <form action = "request_sessions_in" method = "post">
    <p><input type = text name = username /></p>
    <p><input type = submit value = Login /></p>
  </form>
 // request_sessions.py
      app.secret_key = 'any random string'
      @app.route('/request_sessions_submit')
      def request_sessions_submit():
         if 'username' in session:    username = session['username']
            return 'Logged in as ' + username + '<br>' + \
             "<b><a href = '/request_sessions_out'>click here to log out</a></b>"
         return "You are not logged in <br><a href = '/request_sessions_in'></b>" + \
           "click here to log in</b></a>"
      @app.route('/request_sessions_in', methods = ['GET', 'POST'])
      def request_sessions_in():
         if request.method == 'POST':    session['username'] = request.form['username']
            return redirect(url_for('request_sessions_submit'))
         return render_template('request_sessions_submit.html')
      @app.route('/request_sessions_out')
      def request_sessions_out():
         session.pop('username', None)
         return redirect(url_for('request_sessions_submit'))
```

Try - Request Object(Files) 구현과 이해

```
// templates/request_files_submit.html
      <form action = "request_files_upload" method = "POST"
         enctype = "multipart/form-data">
         <input type = "file" name = "file01" /> <input type = "submit"/>
      </form>
 // request_files.py
from werkzeug import secure_filename;          import os
app = Flask(__name__)
app.config [ 'UPLOAD_FOLDER'] = '/home/yojulab/Downloads'
app.config [ 'MAX_CONTENT_PATH'] = 5 * 1024 * 1024
@app.route('/request_files_submit')
def request_files_submit():
    return render_template('request_files_submit.html')
@app.route('/request_files_upload', methods = ['GET', 'POST'])
def request_files_upload():
    if request.method == 'POST':
        file01 = request.files['file01']
        filename01 = secure_filename(file01.filename) → Cross-Site Scripting (XSS)
        file01.save(os.path.join(app.config['UPLOAD_FOLDER'], filename01))
        return 'file uploaded successfully'
```

Email


MTA
emc.fr
MTA / MDA
yahoo.co.uk
1
2
3
4
POP3/IMAP
SMTP
jean-bob@emc.fr
MUA
billy@yahoo.co.uk
MUA

Email

~$ pip3 install flask-mail

- SMTP 서버 설정값 : MAIL_HOSTNAME, MAIL_PORT, MAIL_USE_TLS, MAIL_SSL, MAIL_USERNAME, MAIL_PASSWORD

```
ex) app.config['MAIL_HOSTNAME'] = 'smtp.googlemail.com'
    app.config['MAIL_PORT'] = 587
    app.config['MAIL_USE_TLS'] = True
    app.config['MAIL_USERNAME'] = os.environ.get(['MAIL_USER'])
```

- 비동기화 : 대용량 전송 경우 태스크 큐(Celery etc) 사용

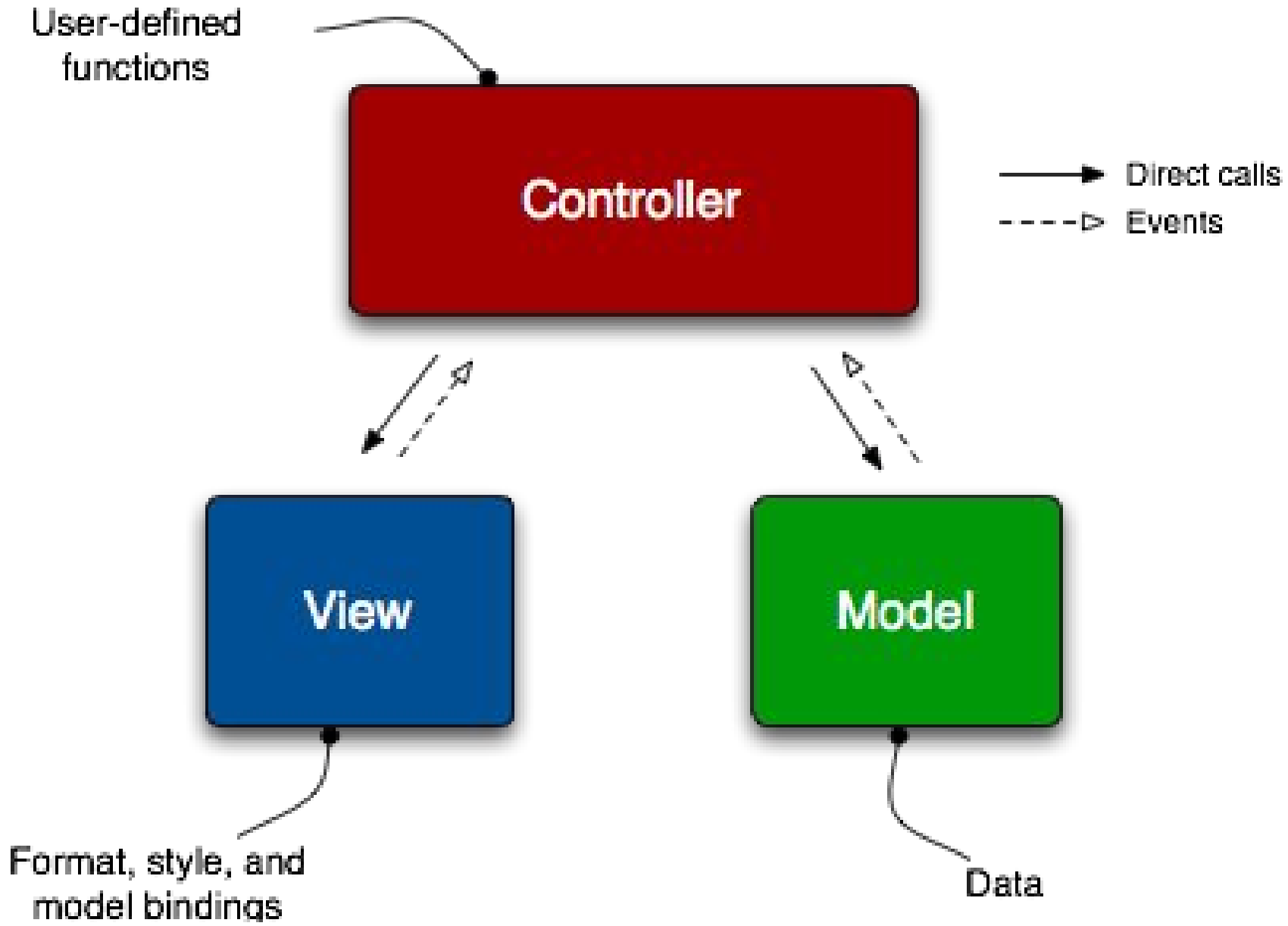
```
ex) from threading import Thread
    def send_sync_email(app, msg):
        with app.app_context():
            mail.send(msg)

    thr = Tread(target=send_async_email, args=[app, msg])
    thr.start()
```

Try - Email 구현과 이해(https://myaccount.google.com/lesssecureapps)

```
from flask import Flask
from flask_mail import Mail, Message
app =Flask(__name__)
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'yourId@gmail.com'
app.config['MAIL_PASSWORD'] = '*****'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail=Mail(app)
@app.route("/mail_submit")
def mail_submit():
   msg = Message('Hello', sender = 'yourId@gmail.com', recipients =
['otter35@naver.com', ‘otter.oh@gmail.com’])          → Try : add 3 Email
   msg.body = "Hello Flask message sent from Flask-Mail"
   mail.send(msg)
   return "Sent"
if __name__ == '__main__':
   app.run(debug = True)
```

웹 기반 애플리케이션 - MVC



WTForm

~$ pip3 install flask-wtf

- CSRF 기본 보호 : 크로스-사이트 리퀘스트

  ex) app.config['SECRET_KEY'] = 'hard to guess string'

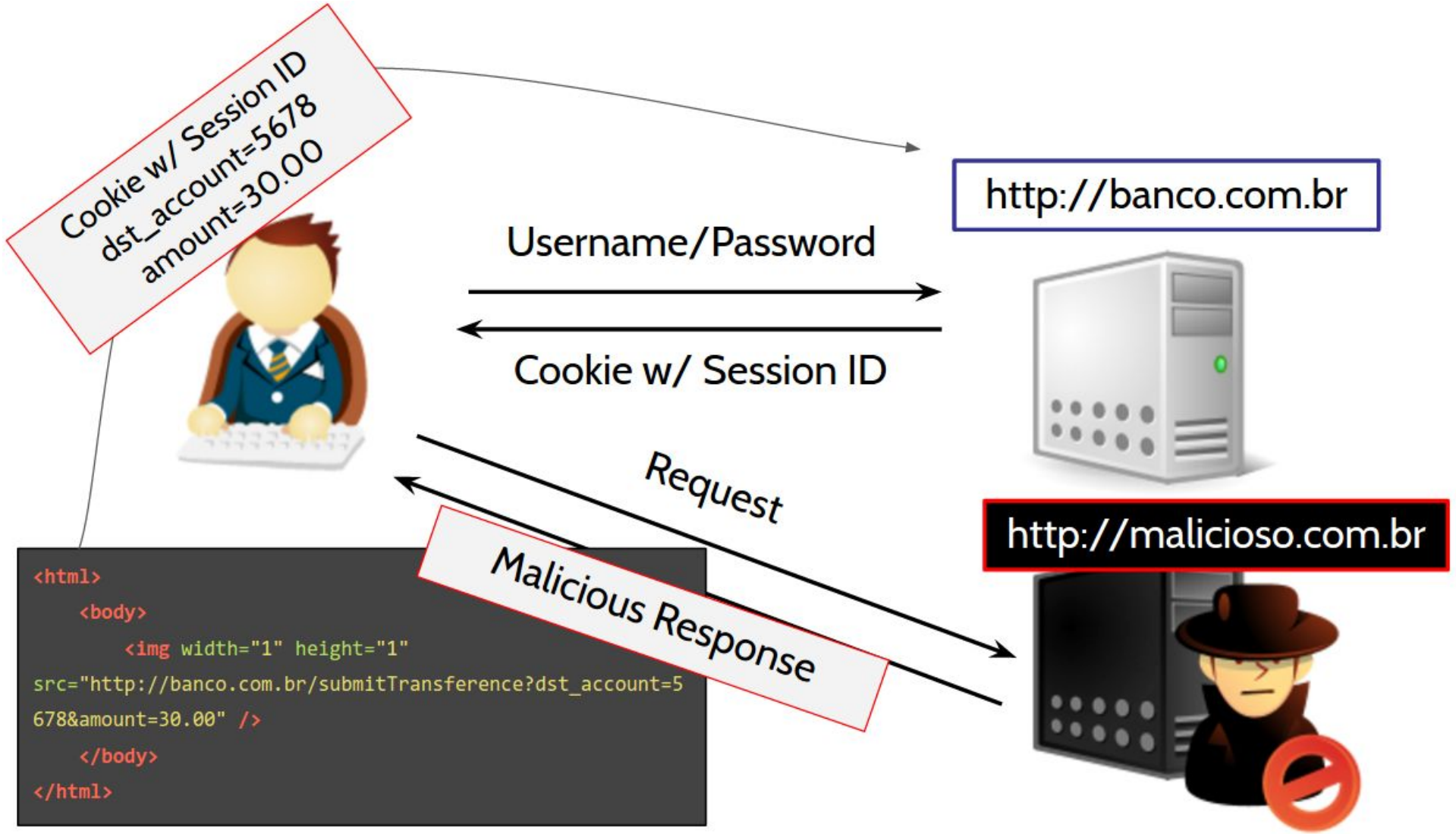
- WTForms HTML Rendering Field
  - StringField, TextAreaField : 한줄과 다중 라인 텍스트
  - PasswordField, HiddenField : 패스워드와 숨겨진 텍스트
  - DateField, DateTimeField : 날짜 시간 포맷 텍스트
  - IntegerField, DecimalField, FloatField : 숫자 받는 텍스트
  - BooleanField, RadioField : 체크박스와 라디오 버튼
  - SelectField, SelectMultipleField, FieldList : 선택 가능 드롭-다운 리스트
  - FileField, SubmitField, FormField : 파일과 서브미션 버튼
- WTForms Validator
  - Email, EqualTo, IPAddress
  - Length, NumberRange, Optional, Required
  - Regexp, URL, AnyOf, NoneOf

  ex) name = StringField('What is your name?', validators=[Required()])

  →

  <input id = "csrf_token" name = "csrf_token" type = "hidden" />
  <label for = "name">What is your name?</label>
  <input id = "name" name = "name" type = "text" request value = "" />

CSRF(Cross-Site Request Forgery)



Try - Flask-WTF 구현과 이해

```
// templates/wtf.html
<form action = "/wtf" method = post>
      {{ form.name.label }}<br>     {{ form.name }}<br>       {{ form.Gender.label }}
      {{ form.Gender }}<br>         {{ form.language.label }}{{ form.language }}<br>
      {{ form.submit }}
</form>
 // wtf.py
    from flask_wtf import Form;          from wtforms import *
    app.secret_key = 'development key'
class ContactForm(Form):
   name = TextField("Name Of Student",[validators.Required("Enter your name.")])
   Gender = RadioField('Gender', choices = [('M','Male'),('F','Female')])
   language = SelectField('Languages', choices = [('cpp', 'C++'), ('py', 'Python')])
   submit = SubmitField("Send")
def wtf():
    form = ContactForm()
    if request.method == 'POST':
        if form.validate() == False:  return render_template('wtf.html', form = form)
        else: return 'success'
    elif request.method == 'GET':
        return render_template('wtf.html', form = form)
```

Try - Message Flashing 구현과 이해

```
// flashing.html
{% with messages = get_flashed_messages() %}
      {% if messages %}
            <ul>{% for message in messages %}
                        <li<{{ message }}</li>
                  {% endfor %}    </ul>
      {% endif %}
{% endwith %}
<form action = "/login" method = post>
<input type = text name = username value = "{{request.form.username }}">
<input type = password name = password>
<input type = submit value = Login></form>
// flashing.py
 @app.route('/login', methods = ['GET', 'POST'])
 def login():
    if request.method == 'POST':
       if request.form['username'] != 'admin' or request.form['password'] != 'admin':
          flash('Invalid username or password. Please try again!')
         return render_template('login.html')
       else:    return redirect(url_for('index'))
```

Try - Error Handler 구현과 이해

```
// 404.html
        {% extends "base.html" %}
        {% block title %}Flasky - Page Not Found{% endblock %}
        {% block page_content %}
        <div class="page-header">   <h1>Not Found</h1>   </div>
        {% endblock %}
// 500.html
        {% block page_content %}
        <div class="page-header">   <h1>Internal Server Error</h1>     </div>
         {% endblock %}
// errorhandler.py          # add flashing.py
        @app.errorhandler(404)
        def page_not_found(e):
            return render_template('404.html'), 404
        @app.errorhandler(500)
        def internal_server_error(e):
            return render_template('500.html'), 500
```

Try(1) - SQL 구현과 이해

❖ 실행결과

http://localhost:5000/enternew http://localhost:5000/list

❖ 따라하기

```
// sqlite3_db.py
import sqlite3
conn = sqlite3.connect('sqlite3_database.db')
conn.execute('CREATE TABLE students (name TEXT, addr TEXT, city TEXT)')
conn.close()
// sqlite3_submit.html
    <form action = "{{ url_for('addrecord') }}" method = "POST">
      Name<br>
      <input type = "text" name = "nm" /></br>
      Address<br>
      <textarea name = "add" ></textarea><br>
      City<br>
      <input type = "text" name = "city" /><br>
      <input type = "submit" value = "submit" /><br>
    </form>
```

Try(2) - SQL 구현과 이해

```
// sqlite3_db.py
@app.route('/enternew')
def enternew():
   return render_template('sqlite3_submit.html')
@app.route('/addrecord',methods = ['POST', 'GET'])
def addrecord():
   if request.method == 'POST':
      try:
         nm = request.form['nm'];addr = request.form['add'];      city = request.form['city']
         with sql.connect("sqlite3_database.db") as con:
            cur = con.cursor()
            cur.execute("INSERT INTO students (name,addr,city) \
                              VALUES (?,?,?)",(nm,addr,city) );
            con.commit(); msg = "Record success"
      except:          con.rollback();     msg = "error in insert operation"
      finally:
         return render_template("sqlite3_result.html",msg = msg);   con.close()
@app.route('/list')
def list():
   con = sql.connect("sqlite3_database.db");      con.row_factory = sql.Row
   cur = con.cursor();  cur.execute("select * from students")
   rows = cur.fetchall();
   return render_template("sqlite3_list.html",rows = rows)
```

SQLAlchemy

- 관계형 데이터 프레임워크.
  ~$ pip3 install flask-sqlalchemy
- Column Type
  - Integer, Text, Numeric, Boolean, DateTime, LargeBinary misc
    ex) id = db.Column(db.Integer, primary_key=True)
  - 열 옵션 : primary_key, unique, index, nullable misc
  - 관계 옵션 : backref, primaryjoin, order_by
    ex) users = db.relationship('User', backref='role')
    role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))
- Query Filters
  - filter(), filter_by(), limit(), order_by(), group_by()
    ex) User.query.filter_by(role=user_role))
- Query Executors
  - all(), first(), get(), count(), paginate()
    ex) user_role.users.order_by(User.username).all()

Try - SQLAlchemy 구현과 이해

```
// sqlalchemy_db.py              → make sqlalchemy_list.html, sqlalchemy_list.html
from flask_sqlalchemy import SQLAlchemy
class students(db.Model):
    id = db.Column('student_id', db.Integer, primary_key = True)
    name = db.Column(db.String(100));          city = db.Column(db.String(150))
    def __init__(self, name, city, addr,pin):      self.name = name;   self.city = city
@app.route('/sqlalchemy_list')
def sqlalchemy_list():
    return render_template('sqlalchemy_list.html', students = students.query.all() )
@app.route('/sqlalchemy_submit', methods = ['GET', 'POST'])
def sqlalchemy_submit():
   if request.method == 'POST':
      if not request.form['name'] or not request.form['city']:
         flash('Please enter all the fields', 'error')
      else:
         student = students(request.form['name'], request.form['city'])
         db.session.add(student);    db.session.commit()
         return redirect(url_for('sqlalchemy_list'))
   return render_template('sqlalchemy_submit.html')
if __name__ == '__main__':
   db.create_all();   app.run(debug = True)
```

REST(Representational State Transger)

- RIA(Rich Internet Application) 대두 : 서버 역할 축소
  - XML-RPC, SOAP
  - REST : WWW 인기
- 특징 : 리소스가 전부
  - 클라이언트 - 서버 분리
  - 스테이트리스
  - 유니폼 인터페이스
  - 계층화된 시스템
  - 코드-온-디맨드

  $ pip3 install flask-httpauth
- Using PostMan in Chrome : Tool(https://chrome.google.com/webstore)
  - GET
    - type in address in chrome : chrome://apps/
    - insert url value in address (ex.http://www.yojulab.com/api/v1.0/tasks)
  - POST
    - click ‘header’ > insert Content-Type : application/json
    - click ‘body’ tap > select ‘raw’ > insert {key:value}
    - insert URL value in address > Click Button ‘Send’
    - Click ‘Code’

Try - REST GET 구현과 이해

❖ 실행결과 : http://localhost:5000/api/v1.0/tasks/2

❖ 따라하기

```
// restful_get.py
from flask import Flask, jsonify
app = Flask(__name__)
tasks = [ {     'id': 1,
                'title': u'Buy groceries',
                'description': u'Milk, Cheese, Pizza, Fruit, Tylenol',
                'done': False
            },
            {   'id': 2,
                'title': u'Learn Python',
                'description': u'Need to find a good Python tutorial on the web',
                'done': False
            } ]

@app.route('/api/v1.0/tasks', methods=['GET'])
def get_tasks():
    return jsonify({'tasks': tasks})
```

Try - REST GET 구현과 이해

❖ 실행결과

http://localhost:5000/api/v1.0/tasks/2

http://localhost:5000/api/v1.0/tasks/5 → need error handler

❖ 따라하기

```
// restful_get.py      # add restful_get.py
from flask import abort
@app.route('/api/v1.0/tasks/<int:task_id>', methods=['GET'])
def get_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    return jsonify({'task': task[0]})

from flask import make_response
@app.errorhandler(404)    # 404 error handler
def not_found(error):
    return make_response(jsonify({'error': 'Not found'}), 404)
```

Try - REST POST 구현과 이해

❖ 실행결과

~$ curl -i -H "Content-Type: application/json" -X POST -d '{"title":"Read a book"}' http://localhost:5000/api/v1.0/tasks

❖ 따라하기

```
// restful_post.py          # copy restful_get.py and add code
from flask import request
@app.route('/api/v1.0/tasks', methods=['POST'])
def create_task():
    if not request.json or not 'title' in request.json:
        abort(400)
    task = {
        'id': tasks[-1]['id'] + 1,
        'title': request.json['title'],
        'description': request.json.get('description', ""),
        'done': False
    }
    tasks.append(task)
```

Try - REST External 구현과 이해

❖ 실행결과

~$ curl -i http://192.43.2.49/api/v1.0/tasks/external

❖ 따라하기

```
// restful_post.py          # add restful_post.py
from flask import url_for
def make_public_task(task):
    new_task = {}
    for field in task:
        if field == 'id':
            new_task['uri'] = url_for('get_task', task_id=task['id'], _external=True)
        else:
            new_task[field] = task[field]
    return new_task
@app.route('/api/v1.0/tasks/external', methods=['GET'])
def get_externaltasks():
    return jsonify({'tasks': [make_public_task(task) for task in tasks]})
```

Try - REST httpauth 구현과 이해

❖ 실행결과 : postMan > authorization > id:password

~$ curl -i yojulab:passwordvalue http://localhost:5000/api/v1.0/tasks/httpauth

❖ 따라하기

```
// restful_httpauth.py          # add restful_post.py
from flask_httpauth import HTTPBasicAuth
auth = HTTPBasicAuth()
@auth.get_password
def get_password(username):
    if username == 'yojulab':
        return 'passwordvalue'          # password
    return None
@auth.error_handler
def unauthorized():
    return make_response(jsonify({'error': 'Unauthorized access'}), 401)
@app.route('/api/v1.0/tasks/httpauth', methods=['GET'])
@auth.login_required
def get_httpauthtasks():
    return jsonify({'tasks': tasks})
```


수고하셨습니다.