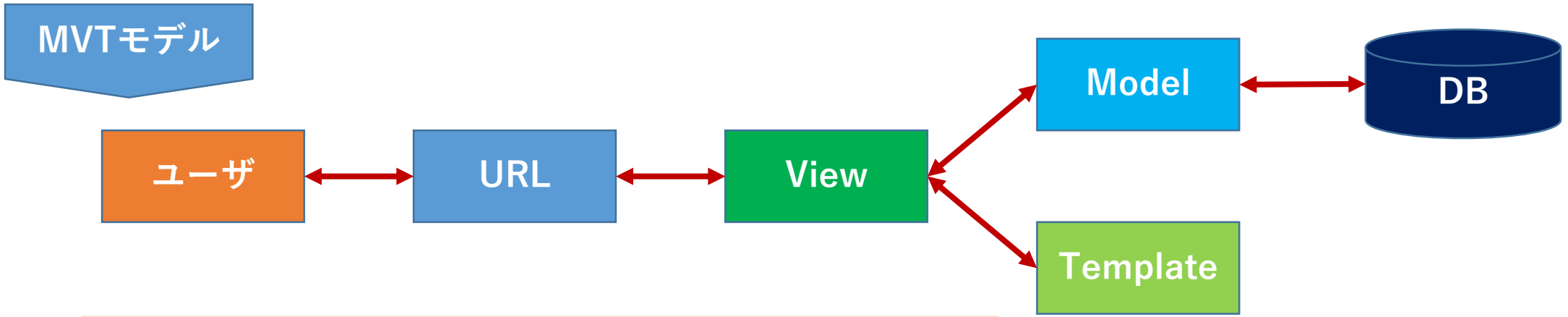
実習科目研修
クラウドコンピューティングA

AWS　AIサービスAPIを使用したAIシステムの開発

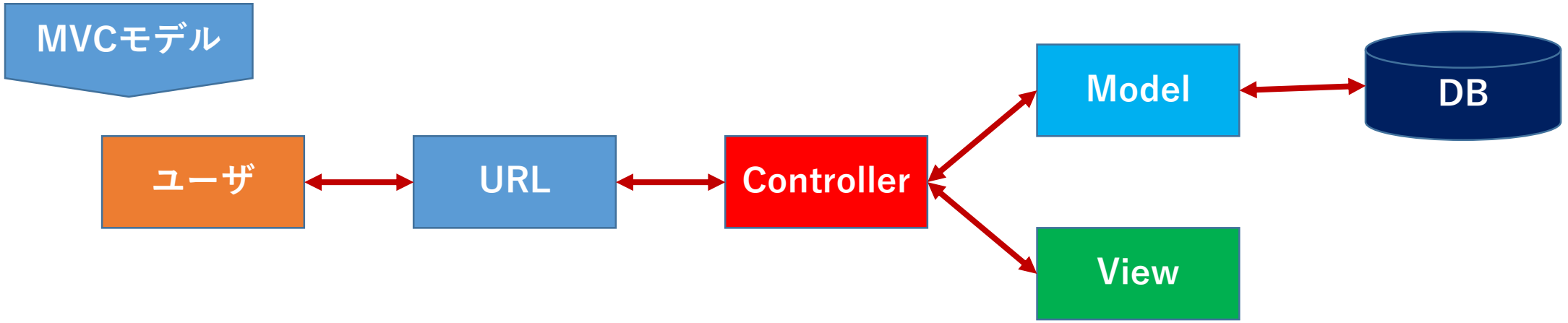
第３回

Flask入門

Flaskで採用されているアーキテクチャ　MVTモデル（Model, View, Template）



Model：ビジネスロジックを担当する
View：入力を受け取り、ModelとTemplateを制御する
Template：入出力を担当する



Flaskのインストール

```
$> pip  install  flask
```

Flaskのディレクトリ構成

```
$> tree
フォルダー パスの一覧:  ボリューム OS
ボリューム シリアル番号は A6BE-1C6F です
C:.
 └─codes                ⇐ 配下にFlaskサーバーアプリ
    ├─static            ⇐ 配下にCSSスタイルシートや画像ファイルなど
    │  ├─css
    │  └─img
    └─templates         ⇐ 配下にテンプレートファイル（HTML）

$>
```

Flaskサーバーの起動と停止

Flaskサーバーの起動

```
$> python flask_routing1.py
 * Serving Flask app "flask_routing1" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://localhost:8888/ (Press CTRL+C to quit)
```

FlaskサーバーのURL

Flaskサーバーの
ポート番号

Flaskサーバーを停止するときは
CTRL＋C

Flaskモジュールのインポート
とFlaskインスタンスの生成

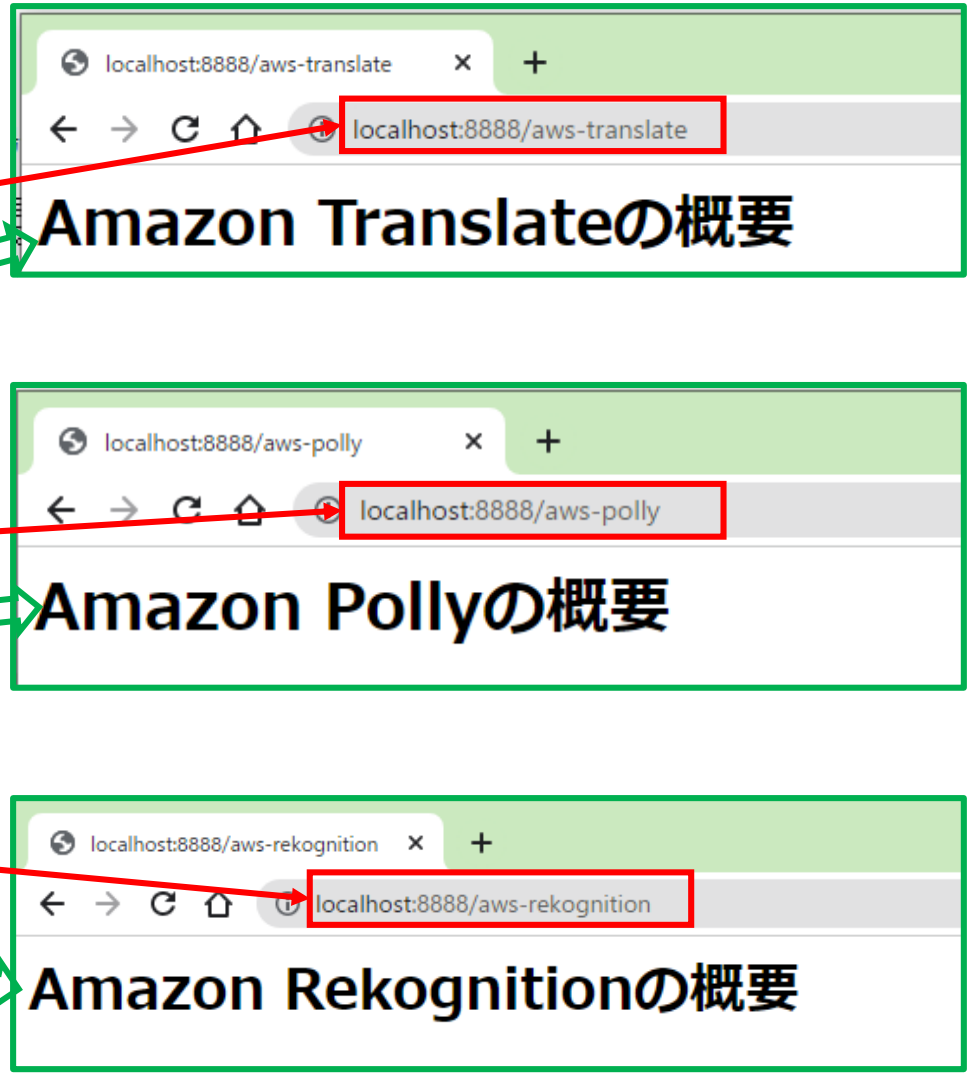
```python
# flask から Flaskモジュールをインポート
from flask import Flask

# Flaskインスタンスを生成
app = Flask(__name__)
```

Flaskサーバーの起動

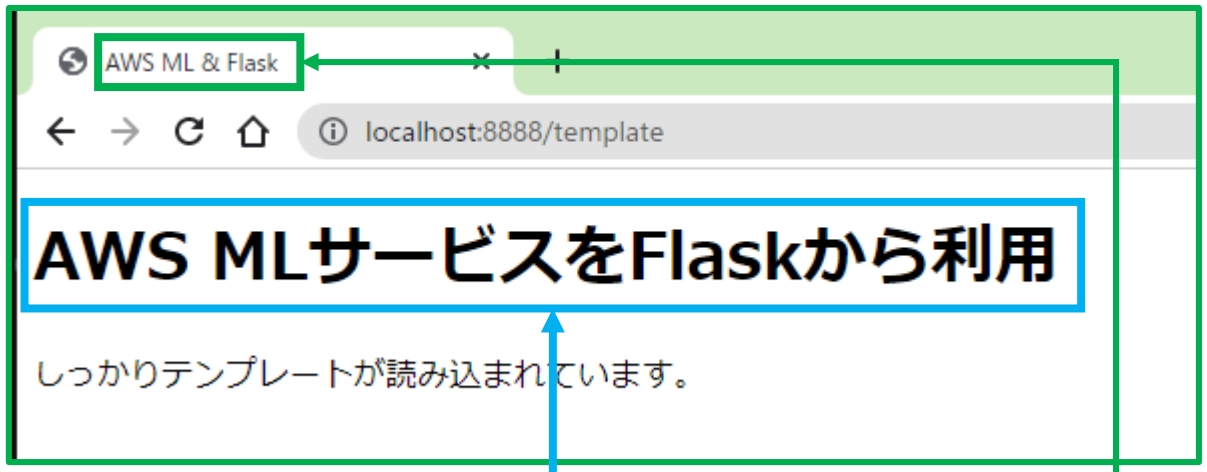
```python
if __name__ == '__main__':
  app.run(host='localhost', port=8888)
```

Flaskのルーティング（flask_routing1.py）

```
# http://XXX/aws-translateをルーティング
# 実行した場合に translate関数が実行される
@app.route('/aws-translate')
def translate():
  return '<h1>Amazon Translateの概要</h1>'

# http://XXX/aws-pollyをルーティング
# 実行した場合にこの関数が実行される
@app.route('/aws-polly')
def polly():
  return '<h1>Amazon Pollyの概要</h1>'

# http://XXX/aws-rekognitionを実行した場合にこの関数が実行される
@app.route('/aws-rekognition')
def rekognition():
  return '<h1>Amazon Rekognitionの概要</h1>'
```



Flaskのテンプレート（flask_template1.py template1.html）




```python
# flask から Flask, render_templateモジュールをインポート
from flask import Flask, render_template

# Flaskインスタンスを生成
app = Flask(__name__)

# http://XXX/templateをルーティング
# 実行した場合に template関数が実行される
@app.route('/template')
def template():
  title = 'AWS ML & Flask'
  message = 'AWS MLサービスをFlaskから利用'
  return render_template('template1.html',
                         title = title,
                         message = message)

if __name__ == '__main__':
  app.run(host='localhost', port=8888)
```

```html
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-
width,initial-scale=1">
  <title>{{ title }}</title>
</head>
<body>
  <h1>{{ message }}</h1>
  <p>しっかりテンプレートが読み込まれています。</p>
</body>
</html>
```

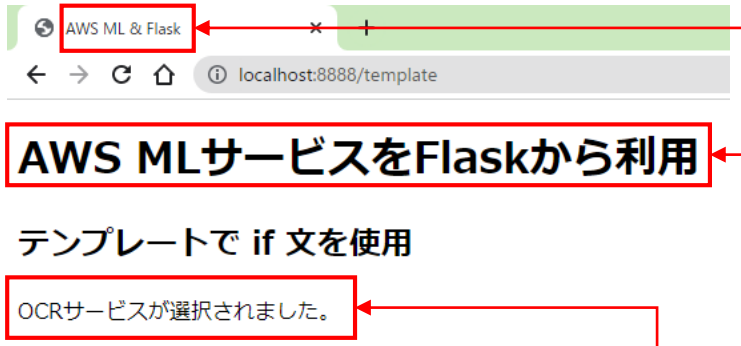
Flaskのテンプレート for文（flask_template_for.py、template_for.html）


```
<head>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width,initial-
scale=1">
  <title>{{title}}</title>
</head>
<body>
  <h1>{{message}}</h1>

  <h2>要素にアクセスしてリストを表示</h2>
  <p>{{ml_services[0]}}サービス</p>
  <p>{{ml_services[1]}}サービス</p>
  <p>{{ml_services[2]}}サービス</p>
  <p>{{ml_services[3]}}サービス</p>
  <hr>

  <h2>for文でリストを表示</h2>
  {% for service in ml_services %}
    <p>{{ service }}サービス</p>
  {% endfor %}
  <hr>

  <h2>for文で辞書を表示</h2>
  {% for k, v in ml_service_dict.items() %}
    <p>{{k}}サービスの価格は１日{{v}}円</p>
  {% endfor %}
</body>
</html>
```

```html
<head>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width,initial-
scale=1">
  <title>{{title}}</title>
</head>
<body>
  <h1>{{message}}</h1>

  <h2>要素にアクセスしてリストを表示</h2>
  <p>{{ml_services[0]}}サービス</p>
  <p>{{ml_services[1]}}サービス</p>
  <p>{{ml_services[2]}}サービス</p>
  <p>{{ml_services[3]}}サービス</p>
  <hr>

  <h2>for文でリストを表示</h2>
  {% for service in ml_services %}
    <p>{{ service }}サービス</p>
  {% endfor %}
  <hr>

  <h2>for文で辞書を表示</h2>
  {% for k, v in ml_service_dict.items() %}
    <p>{{k}}サービスの価格は１日{{v}}円</p>
  {% endfor %}
</body>
</html>
```

```python
def template():
  # テンプレートに渡す各種変数を初期化
  title = 'AWS ML & Flask'
  message = 'AWS MLサービスをFlaskから利用'
  # リスト
  ml_services =
      ['Translate', 'Polly', 'Textract', 'Rekognition']
  # 辞書
  ml_service_dict = {'Translate' : 120,
                     'Polly' : 160,
                     'Textract' : 200,
                     'Rekognition' : 220}
  # テンプレートに各種変数を渡してレンダリング
  return render_template('template_for.html',
                         title = title,
                         message = message,
                         ml_services = ml_services,
                         ml_service_dict = ml_service_dict)
```

Flaskのテンプレート if文（flask_template_if.py、template_if.html）

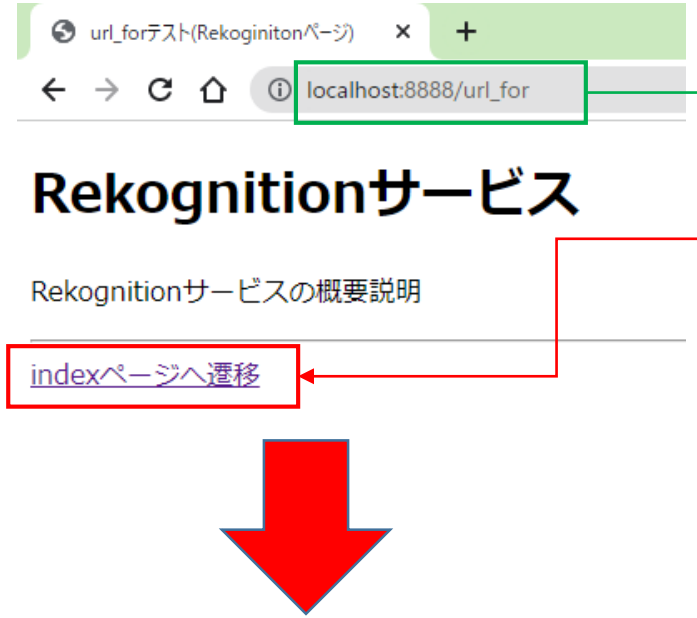



```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-
width,initial-scale=1">
  <title>{{title}}</title>
</head>
<body>
  <h1>{{message}}</h1>

  <h2>テンプレートで if 文を使用</h2>
  {% if ml_srvice == 'Translate' %}
    <p>翻訳サービスが選択されました。</p>
  {% elif ml_service == 'Polly' %}
    <p>音声合成サービスが選択されました。</p>
  {% elif ml_service == 'Textract' %}
    <p>OCRサービスが選択されました。</p>
  {% elif ml_service == 'Rekognition' %}
    <p>画像認識サービスが選択されました。</p>
  {% else %}
    <p>その他のサービスが選択されました。</p>
  {% endif %}

</body>
</html>
```

```
def template():
  # テンプレートに渡す各種変数を初期化
  title = 'AWS ML & Flask'
  message = 'AWS MLサービスをFlaskから利用'
  # リスト
  ml_services = ['Translate', 'Polly',
                 'Textract', 'Rekognition']
  # テンプレートに各種変数を渡してレンダリング
  return render_template('template_if.html',
                         title = title,
                         message = message,
                         ml_service = ml_services[2])
```

式	意味
==	左辺と右辺が等しい
!=	左辺と右辺が等しくない
<	左辺が右辺より小さい
<=	左辺が右辺以下
>	左辺が右辺より大きい
>=	左辺が右辺以上
in	左辺がリスト、タプル、辞書などで左辺の値が含まれる
not in	左辺がリスト、タプル、辞書などで左辺の値が含まれない

Flaskのテンプレート url_for（url_for_views.py、rekognition.html、index.html）



recognition.html

```
<body>
  <h1>Rekognitionサービス</h1>
  <p>Rekognitionサービスの概要説明</p>
  <hr>
  <a href="{{ url_for('index') }}">
    indexページへ遷移
  </a>
</body>
```

index.html

```
<body>
  <h1>Amazon MLサービス</h1>
  <p>Amazon MLサービスの概要説明</p>
</body>
```

```
# http://XXX/をルーティング
# 実行した場合に index関数が実行される
@app.route('/')
def index():
    return render_template('index.html')

# http://XXX/url_forをルーティング
# 実行した場合に url_for関数が実行される
@app.route('/url_for')
def url_for():
    return
render_template('rekognition.html')
```

Flaskのテンプレートの継承

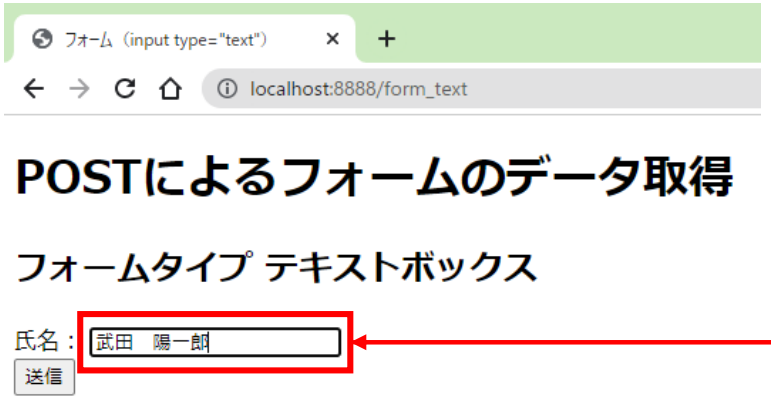
form_base.html

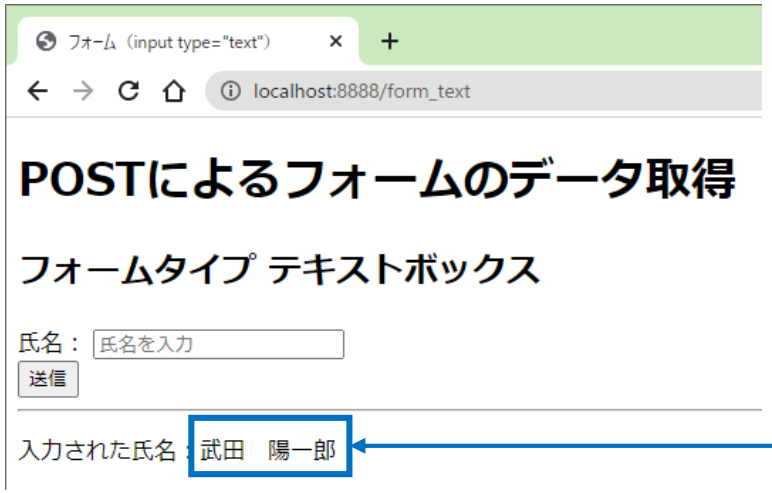
```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>
    {% block title %}{{ title }}{% endblock %}
  </title>
</head>
<body>
  <h1>POSTによるフォームのデータ取得</h1>
  <div>{% block content %}{% endblock %}</div>
</body>
</html>
```

form_text.html

```
{% extends "form_base.html" %}
{% block title %}{{super()}}{% endblock %}
{% block content %}
<div>
  <h2>フォームタイプ {{ form_type }}</h2>
  <form action="/form_text" method="POST" enctype="multipart/form-data">
    <div>
      <label for="name">氏名：</label>
      <input type="text" id="name" name="name" placeholder="氏名を入力">
    </div>
    <div>
      <input type="submit" value="送信">
    </div>
  </form>
  <hr>
  <div><p>入力された氏名：{{ file_name }}</p></div>
</div>
{% endblock %}
```

Flaskとフォーム（テキストボックス）form_text.py、form_text.html

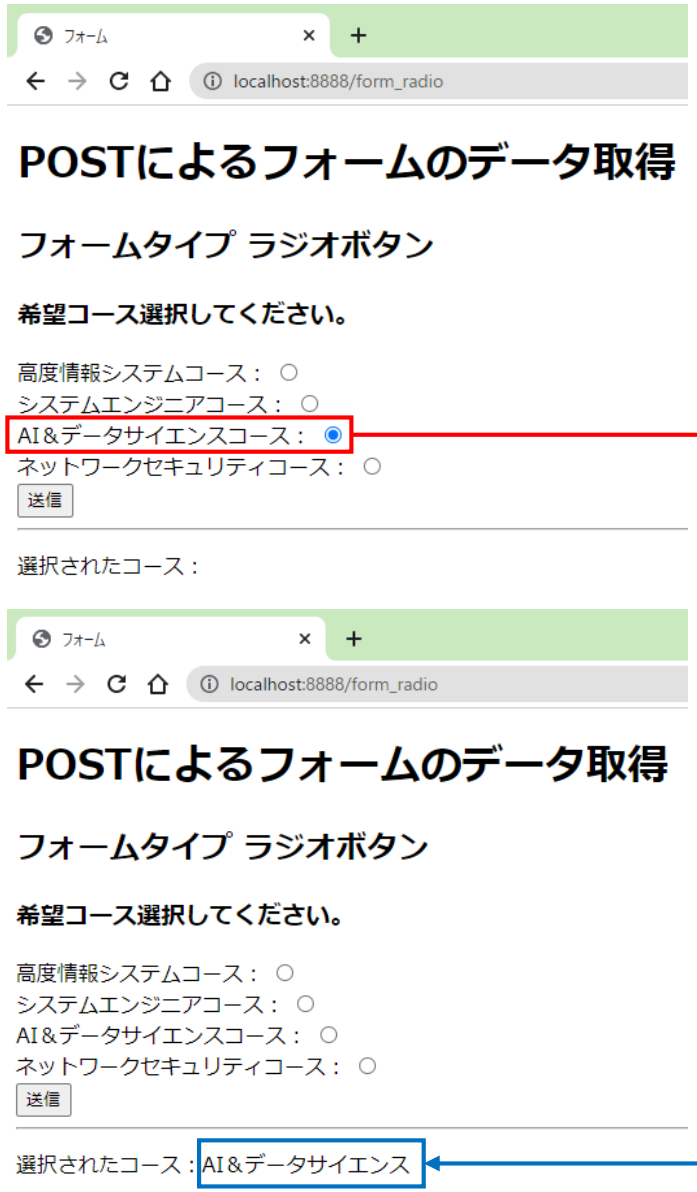





```
<div>
  <h2>フォームタイプ {{ form_type }}</h2>
  <form action="/form_text" method="POST" enctype="multipart/form-data">
    <div>
      <label for="name">氏名：</label>
      <input type="text" id="name" name="name" placeholder="氏名を入力">
    </div>
    <div>
      <input type="submit" value="送信">
    </div>
  </form>
  <hr>
  <div><p>入力された氏名：{{ file_name }}</p></div>
</div>
```

```
# http://XXX/form_textをルーティング
# 実行した場合に form_text関数が実行される
@app.route('/form_text', methods=['GET', 'POST'])
def form_text_post():
  title = 'フォーム（input type="text"）'
  # method が POST か否かを判断
  if request.method == 'POST':
    # POST のとき
    file_name = request.form.get('name')
  else:
    # GET のとき
    file_name = ''
  form_type = 'テキストボックス'
  return render_template('form_text.html',
                         title = title,
                         file_name = file_name,
                         form_type = form_type)
```

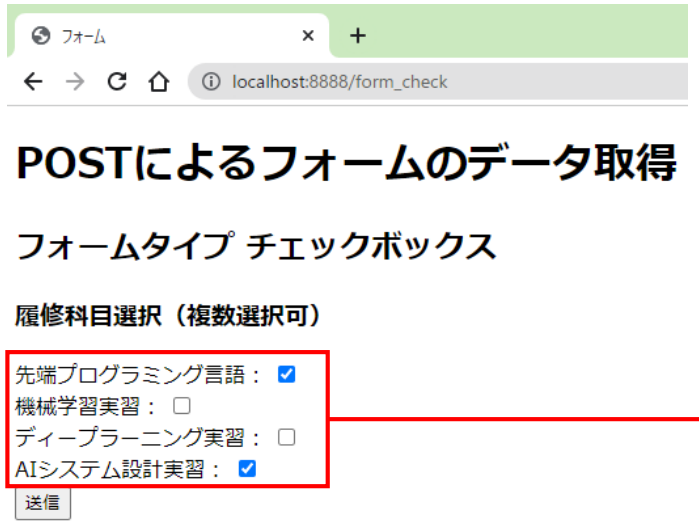
Flaskとフォーム（ラジオボタン）form_radio.py、form_radio.html

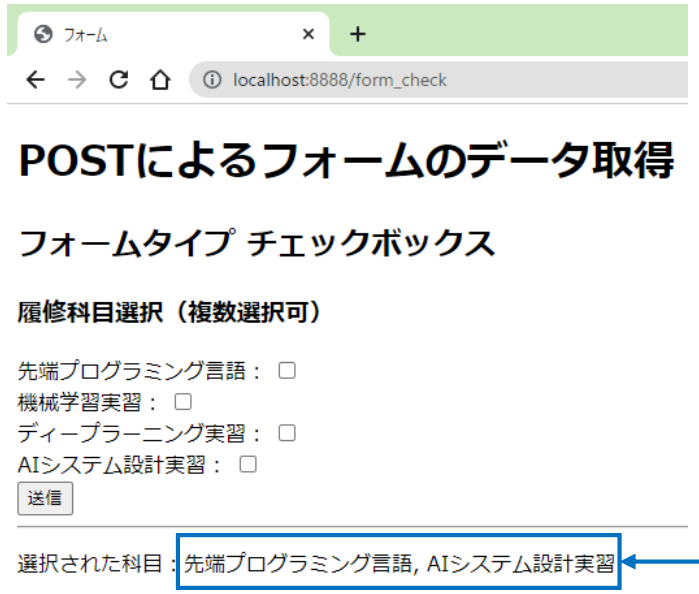



```python
# http://XXX/form_radioをルーティング
# 実行した場合に form_radio関数が実行される
@app.route('/form_radio', methods=['GET', 'POST'])
def form_radio():
  title = 'フォーム'
  # method が POST か否かを判断
  if request.method == 'POST':
    # POST のとき
    cource_name = request.form.get('radio')
  else:
    # GET のとき
    cource_name = ''
  form_type = 'ラジオボタン'
  return render_template('form_radio.html',
                         title = title,
                         cource_name = cource_name,
                         form_type = form_type)
```

```
  </form>
  <hr>
  <div><p>選択されたコース：{{ cource_name }}</p></div>
</div>
```

Flaskとフォーム（チェックボックス）form_check.py、form_check.html

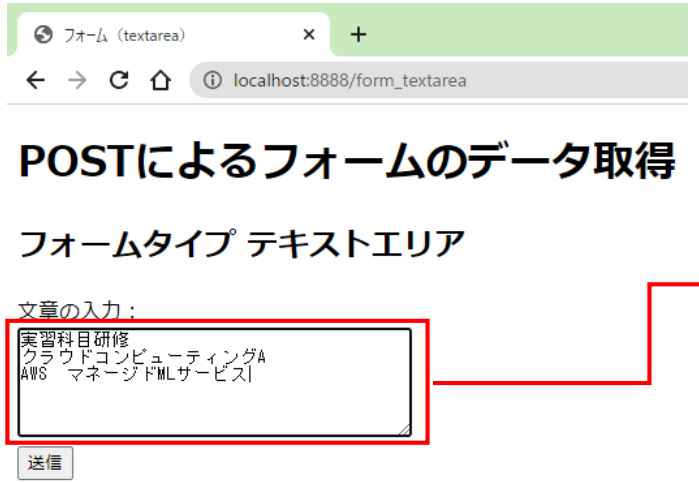




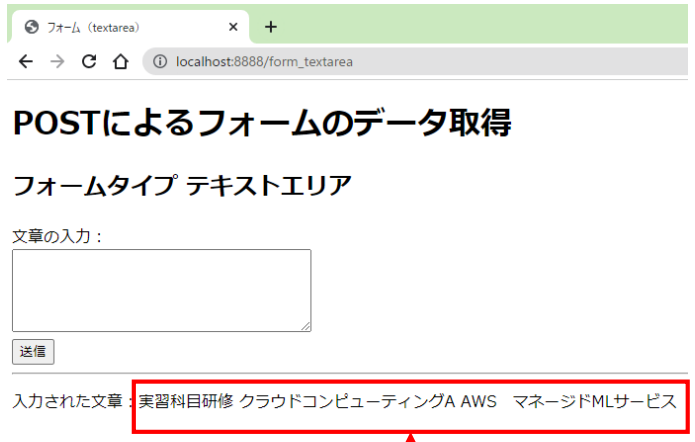

```python
# http://XXX/form_checkをルーティング
# 実行した場合に form_check関数が実行される
@app.route('/form_check', methods=['GET', 'POST'])
def form_check():
  title = 'フォーム'
  # method が POST か否かを判断
  if request.method == 'POST':
    # POST のとき
    selected_list = request.form.getlist('checkbox')
    subject_name = ', '.join(selected_list)
  else:
    # GET のとき
    subject_name = ''
  form_type = 'チェックボックス'
  return render_template('form_check.html',
                         title = title,
                         subject_name = subject_name,
                         form_type = form_type)
```

```
</form>
  <hr>
  <div><p>選択された科目：{{ subject_name }}</p></div>
</div>
```

Flaskとフォーム（テキストエリア）form_check.py、form_check.html




```python
# http://XXX/form_textareaをルーティング
# 実行した場合に form_text_area関数が実行される
@app.route('/form_textarea', methods=['GET', 'POST'])
def form_text_area():
  title = 'フォーム（textarea）'
  # method が POST か否かを判断
  if request.method == 'POST':
    # POST のとき
    texts = request.form.get('texts')
  else:
    # GET のとき
    texts = ''
  form_type = 'テキストエリア'
  return render_template('form_textarea.html',
                         title = title,
                         texts = texts,
                         form_type = form_type)
```




```html
<div>
  <h2>フォームタイプ {{ form_type }}</h2>
  <form action="/form_textarea" method="POST" enctype="multipart/form-data">
    <div>
      <label for="name">文章の入力：</label>
    </div>
    <div>
      <textarea name="texts" rows="6" cols="40"></textarea>
    </div>
    <div>
      <input type="submit" value="送信">
    </div>
  </form>
  <hr>
  <div><p>入力された文章：{{ texts }}</p></div>
</div>
```

Flaskとフォーム（画像の読み込みと表示）form_check.py、form_check.html

フォーム

localhost:8888/form_file_image

POSTによるフォームのデータ取得

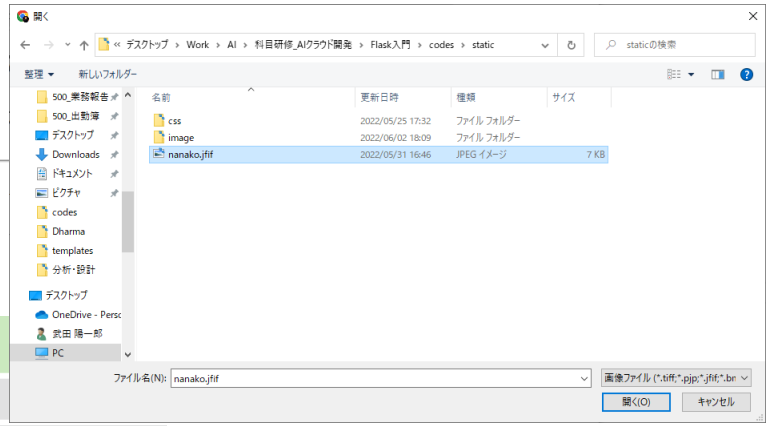
フォームタイプ ファイル選択

画像ファイル

ファイルを選択 選択されていません

送信

選択された画像：



フォーム

localhost:8888/form_file_image

POSTによるフォームのデータ取得

フォームタイプ ファイル選択

画像ファイル

ファイルを選択 nanako.jfif

送信

選択された画像：

フォーム

localhost:8888/form_file_image

POSTによるフォームのデータ取得

フォームタイプ ファイル選択

画像ファイル

ファイルを選択 選択されていません

送信

選択された画像：nanako.jfif


```
<h2>フォームタイプ {{ form_type }}</h2>
<form action="/form_file_image" method="POST" enctype="multipart/form-data">
  <div><h3>画像ファイル</h3></div>
  <div>
    <input type="file" name="image" accept="image/*" required>
  </div>
  <div>
    <input type="submit" value="送信">
  </div>
</form>
<hr>
<div>
  <p>選択された画像：{{ filename }}</p>
  <img src="{{ image_url }}" alt="{{ image_name }}">
</div>
```

```
# http://XXX/form_file_imageをルーティング
# 実行した場合に form_file_image関数が実行される
@app.route('/form_file_image', methods=['GET', 'POST'])
def form_file_image():
  title = 'フォーム'
  # method が POST か否かを判断
  if request.method == 'POST':
    # POST のとき
    # ファイルのリクエストパラメータを取得
    f = request.files.get('image')
    # ファイル名を取得
    filename = secure_filename(f.filename)
    # ファイルを保存するディレクトリを指定
    filepath = 'static/image/' + filename
    # ファイルを保存する
    f.save(filepath)
  else:
    # GET のとき
    filename = ''
    filepath = ''
  form_type = 'ファイル選択'
  return render_template('form_file_image.html',
                         title = title,
                         filename = filename,
                         form_type = form_type,
                         image_name = filename,
                         image_url = filepath)
```