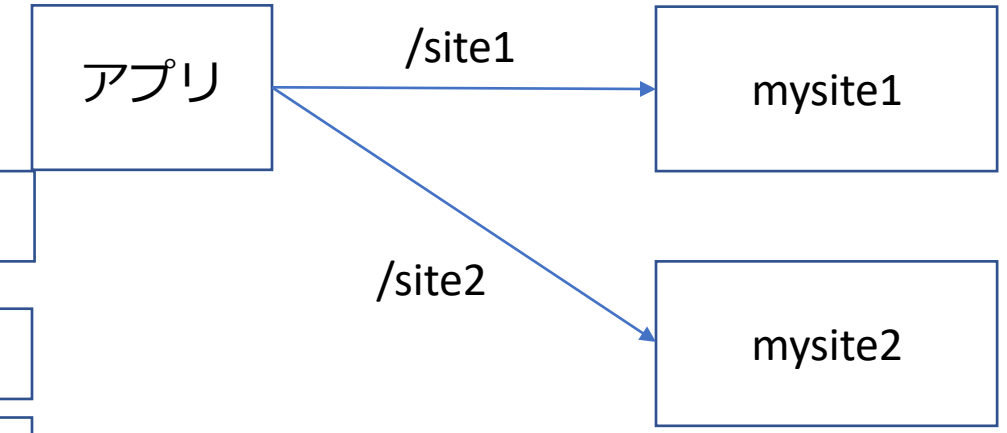
Flask(BluePrint)

View, URLを機能、アプリ単位でグループ分けしたい場合に利用する

 # BluePrintのインスタンス作成

```
mysite1_bp = Blueprint('mysite1', __name__, url_prefix('/site1'))
mysite2_bp = Blueprint('mysite2', __name__, url_prefix('/site2'))
app.register_blueprint(mysite1_bp)
```



アプリの構造

```
| ── flaskr/
|    | ── __init__.py
|    | ── mysite1/
|    |    └── views.py
|    | ── mysite2/
|    |    └── views.py
|    | ── templates/
|    |    | ── base.html
|    |    | ── mysite1/
|    |    |    └── site.html
|    |    └── mysite1/
|    |         └── site.html
| ── setup.py
```

mysite1, mysite2のBluePrintを登録

mysite1のBluePrintを作成

mysite2のBluePrintを作成

各テンプレートを配置

アプリを立ち上げる用

Flask(ログイン画面で利用するパスワードの暗号化)

パスワードの暗号化

ログインユーザを作成するには、パスワードを暗号化する必要があります。
暗号化方法には、wekzeug.securityかbcryptを利用するとよいです
(bcryptはより強固なパスワードの暗号化を行います)

pip install flask-bcrypt flask-login

from flask_bcrypt import Bcrypt
bcrypt = Bcrypt()
password = 'superpassword'
hashed_password = bcrypt.generate_password_hash(password=password) **# bcryptでパスワードを暗号化**
check = bcrypt.check_password_hash(hashed_password, 'superpassword') **# bcryptで暗号化したパスワードが正しいかチェック**

from werkzeug.security import generate_password_hash, check_password_hash

hashed_pass = generate_password_hash('mypassword') **# パスワードの暗号化**
check = check_password_hash(hashed_pass, 'wrong') **# 暗号化したパスワードが正しいかチェック**

Flask(ログイン画面を作成する)

ログイン処理をするようにライブラリ(pip install flask-login)

login_manager = LoginManager() **# Flask-Loginライブラリとアプリケーションを協調して動作させる**
login_manager. init_app(app) **# アプリケーションをログインに設定する。Flask-Loginはセッションを利用するので、SECRET_KEYを設定しなければいけない**

@login_manager.user_loader # セッションに保存されたログインしたユーザを返すために利用される。
def load_user(user_id):
    return User.query.get(user_id)

{% if current_user.is_authenticated %} **# ユーザの認証がされているか確認する**
  Hi {{ current_user.name }}!
{% endif %}

login_manager.login_view = “users.login“ **# ログインするViewの関数を設定する**
login_manager.login_message = “ログイン画面です” **# ログイン画面にリダイレクトした場合のメッセージ**

from flask_login import UserMixin **# Flask-Loginライブラリを利用するユーザが持つオブジェクトを定義**
login_user(from flask_login imort login_user) **# ユーザをログインさせる**
login_required(from flask_login imort login_required) **# ログインが必要な画面に追加するデコレータ関数**
logout_user(from flask_login imort logout_user) **# ユーザをログアウトさせる**

Flask(Ajax)

Ajaxは非同期通信を行って、インタフェース構築する技術で、画面遷移なくサーバとの情報のやり取りを行う
(参考: https://flask.palletsprojects.com/en/1.1.x/patterns/jquery/)

Jqueryの読み込み https://developers.google.com/speed/libraries#jquery

ViewにAjax実行用の関数作成
値をクライアントから取得してJson形式でデータを返却する

```
def add_numbers():
    a = request.args.get('a', 0, type=int)
    b = request.args.get('b', 0, type=int)
    return jsonify(result=a + b) # json形式で値を返す(from flask import jsonify)
```

Templateでリクエストを送信するスクリプトの作成

```
$(function() {
    $('a#calculate').bind('click', function() {
      $.getJSON('/_add_numbers', { # url,_add_numbersにリクエストを送る
        a: $('input[name="a"]').val(), # 引数
        b: $('input[name="b"]').val() # 引数
      }, function(data) {
        $("#result").text(data.result); # 返り値(data)を取得して、id=resultに値を設定する
      });
      return false;
    });
  });
```