Flaskアプリの概要

アジェンダ

1. ディレクトリ構成
2. Webアプリの仕組みについて
3. 開発の流れ

ディレクトリ構成

ディレクトリ構成について

【基本的なディレクトリ構成】

wsgi.py

application_ver2

- ├── __init__.py
- ├── __pycache__
- ├── config
- ├── **forms**
- ├── **models**
- ├── static
- ├── **templates**
- └── **views**

【開発で触れるディレクトリ】

1. Templates
2. Views
3. Forms
4. models

Viewsについて

【役割】

URLと表示されるHTMLを紐づける役割

【書き方について】(例: https://localhost:5000/homeの場合)

```python
@app.route('/home') # URLを指定
def index():
    # 処理を書く
    return render_template('index.html') # どのHTMLを表示するかを指定
```

Templatesについて

【役割】

HTMLファイルを格納する役割

【レイアウトの使用について】

1. 共通部分とメイン部分で分けて定義する。
2. 共通部分については、基本的に触れない
3. 共通部分はLayoutに格納
4. 学生,塾関連,学校関連,全体共通で分別して格納

Formsについて

【役割】

サイトで入力箇所がある場合に、**その形式をPythonのclassで定義**する。

【書き方について】(例: ログインフォームの場合)

1. **入力箇所は、emailとpasswordの2つ**
2. **EmailはStingField(文字列)でDataRequired(入力必須),Emailとして認識します**
3. **PasswordはPasswordField(入力すると**に変化)で、入力必須と認識します。**

Modelsについて

【役割】

ユーザデータ(ログイン要件以外)や組織データをデータベースとしてどのように保存するかをクラスで定義する。

【書き方について】

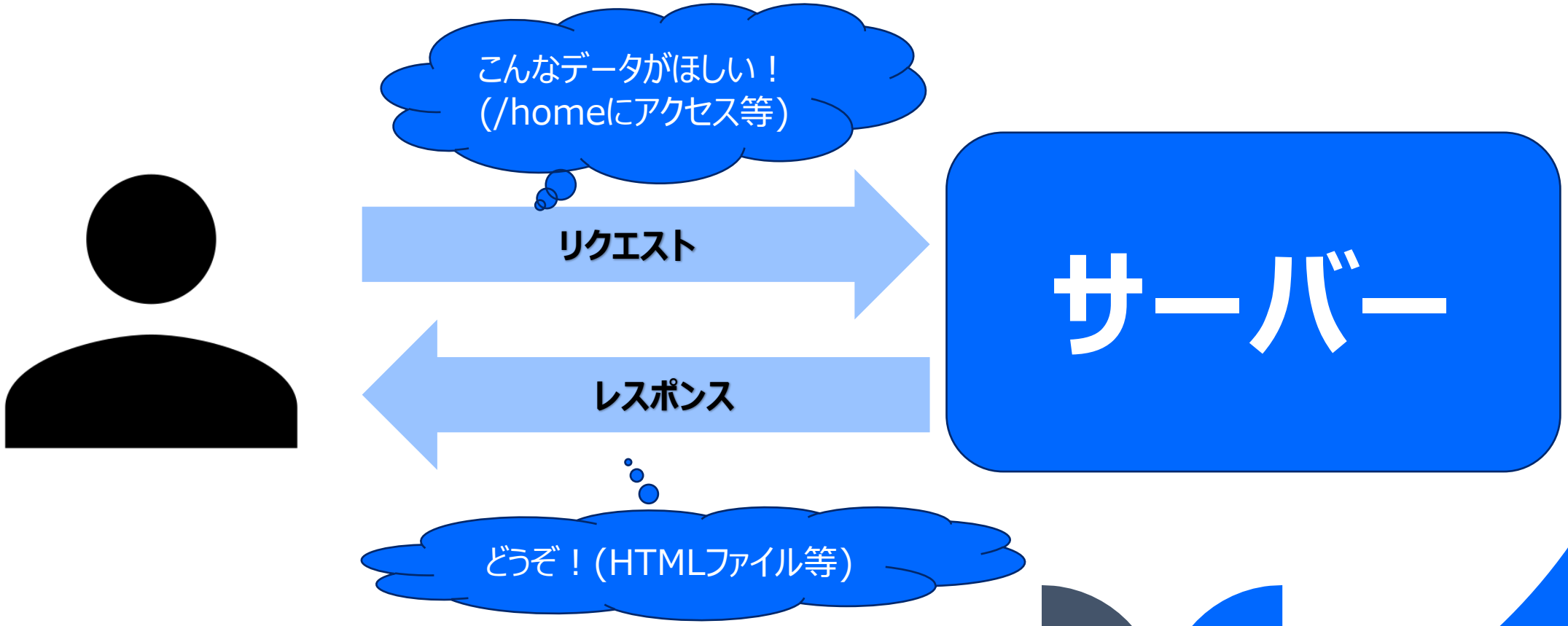
※Modelsを変更すると、データベース形式が崩れるので、影響範囲が大きいです。

そのため、変更は基本的に行わないでください。

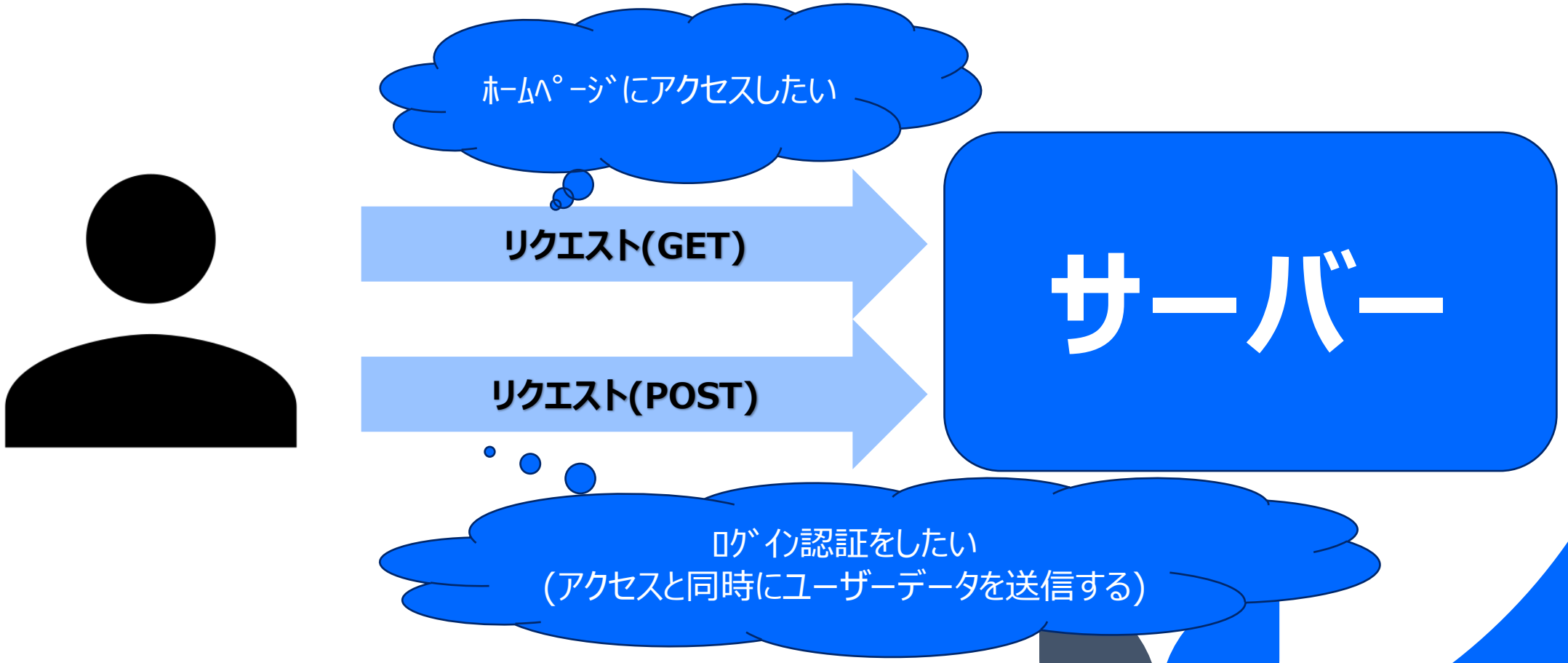
各機能で新規に定義する場合には、連絡ください。

Webアプリの仕組み

Webアプリの仕組みについて



リクエストの種類について



開発の流れ

開発の流れ

1. **Viewsで関数を定義し、HTMLファイルと接続する**

   基本的に、各機能におけるドメインは、あらかじめ決めて設定してあります。

2. **Views関数内の処理を記述する**

   GETとPOSTで別々に記述する

   TRY,EXCEPT等の例外処理をすべてに適用することで、セキュリティ脆弱性に関わるエラーを防止

3. **HTMLファイルを記述し、css等が反映されているか確認する**

   Block contentsとendblock内に記述する