Домашнее задание 2

Обработка HTTP запросов

Целью домашнего задания является:

- создание основных view сайта (отображающих “рыбу”);
- настройка маршрутизации URL;
- использование шаблонизатора для генерации HTML;
- постраничное отображение (пагинация).

1. Создание директории проекта ask_pupkin

Создаем корневую директорию проекта и папки для статики (стилей, javascript, картинок и файлов пользователя в **/uploads/**)

```
mkdir -p ~/projects/askme
cd  ~/projects/askme
mkdir -p {static,uploads,templates}
mkdir -p static/{css,js,img}
```

Далее рекомендуется создать в каталоге git репозиторий - шпаргалка. Обязательно создайте корректный файл .gitignore в котором нужно будет исключить **venv**, ***.pyc** и прочие временные и не относящиеся к проекту папки и файлы, например **.idea/**.

2. Создание django проекта и приложения

Разворачиваем в директории проекта python virtualenv ставим django. С помощью утилиты django-admin (появляется после установки django) нужно создать следующую структуру:

```
askme                - директория проекта
    |--- app           - директория приложения (создается командой startproject)
    |--- askme         - библиотеки проекта (будут созданы django-admin.py)
    |--- manage.py     - скрипт управления (будет создан django-admin.py)
    |--- templates     - шаблоны
    |--- static        - статические файлы (JS, CSS, картинки)
    └--- uploads       - файлы загруженные юзером
```

При этом директории templates, static, uploads — придется создать самостоятельно.

3. Отображение данных

Разработать view для отображения следующих страниц:

- список новых вопросов (главная страница) (URL = **/**)
- список “лучших” вопросов (URL = **/hot/**)
- список вопросов по тэгу (URL = **/tag/blablabla/**)
- страница одного вопроса со списком ответов (URL = **/question/35/**)
- форма логина (URL = **/login/**)

- форма регистрации (URL = **/signup/**)
- форма создания вопроса (URL = **/ask/**)

На данном этапе view не обязаны выбирать данные из базы, их задача - подготовить данные-заглушки (“рыбу”) и передать на отрисовку в соответствующий шаблон. Например таким образом:

```
questions = []
for i in xrange(1,30):
  questions.append({
    ‘title’: ‘title ‘ + str(i),
    ‘id’: i,
    ‘text’: ‘text’ + str(i)
  })
```

Далее список **questions** можно будет легко заменить на выборку из базы данных. Views отображающие формы (например форму логина) - могут просто вызывать шаблон.

4. Маршрутизация URL

Для всех вышеперечисленных view нужно прописать маршруты в **urls.py**. Маршруты должны быть именованными.

5. Шаблонизация

Для каждой из вышеперечисленных view нужно создать отдельный шаблон, отрисовывающий HTML данной страницы. В шаблонах **не должно быть** дублирования HTML - для этого следует использовать наследование шаблонов и подшаблоны (include). В шаблонах должны быть прописаны гиперссылки для переходов на другие страницы. URL **не хардкодить**, вместо этого использовать тэг **url**.

6. Функция пагинации

Общим функционалом для этих страниц является пагинация. Поэтому функционал пагинации следует вынести в отдельную функцию, которая будет принимать список объектов (или **query-set**), объект запроса (**request**) и будет возвращать нужную страницу и данные для шаблона пагинатора. Некорректные параметры пагинации нужно заменять на значения по-умолчанию. Примерная сигнатура функции такова:

```
def paginate(objects_list, request, per_page=10):
    # do smth with Paginator, etc…
    return page
```

7. Полезные ссылки

- Туториал по Django Views;
- Документация по шаблонам;
- Наследование шаблонов;
- Постраничный вывод (пагинация).