Программирование веб-приложений на стороне сервера

ЛЕКТОР: ИПАТОВА ЮЛИЯ НИКОЛАЕВНА

Введени

Е

Большинство крупных веб-сайтов используют программирование серверной части, чтобы динамично отображать различные данные при необходимости, в основном взятые из базы данных, располагающейся на сервере и отправляемые клиенту для отображения через некоторый код (например, HTML и JavaScript).

Основная польза в том, что оно позволяет формировать контент веб-сайта под конкретного пользователя. Динамические сайты могут выделять контент, который более актуален в зависимости от предпочтений и привычек пользователя, упростить использование сайтов за счёт сохранения личных предпочтений и информации.

Даёт возможность взаимодействовать с пользователем сайта, посылая уведомления и обновления по электронной почте или по другим каналам.

Клиент-сервер

Веб-браузеры взаимодействуют с веб-серверами при помощи гипертекстового транспортного протокола (HTTP). Когда вы как-либо взаимодействуете с веб-страницей, HTTP-запрос отправляется из вашего браузера на целевой сервер.

Запрос включает в себя URL, определяющий затронутый ресурс, метод, определяющий требуемое действие и может включать дополнительную информацию, закодированную в параметрах URL.

Веб-серверы ожидают сообщений с клиентскими запросами, обрабатывают их по прибытию и отвечают веб-браузеру при помощи ответного HTTP сообщения (HTTP-ответ)

Возможности серверной части

Программирование серверной части очень полезно поскольку позволяет эффективно доставлять информацию, составленную для индивидуальных пользователей и, таким образом, создавать намного лучший опыт использования.

Некоторые типичные применения и выгоды бэкенда перечислены ниже:

- Эффективное хранение и доставка информации.
- Настраиваемый пользовательский опыт взаимодействия.
- Контролируемый доступ к контенту.
- Хранение информации о сессии/состоянии.
- Уведомления и средства связи.
- Анализ данных и т.д.

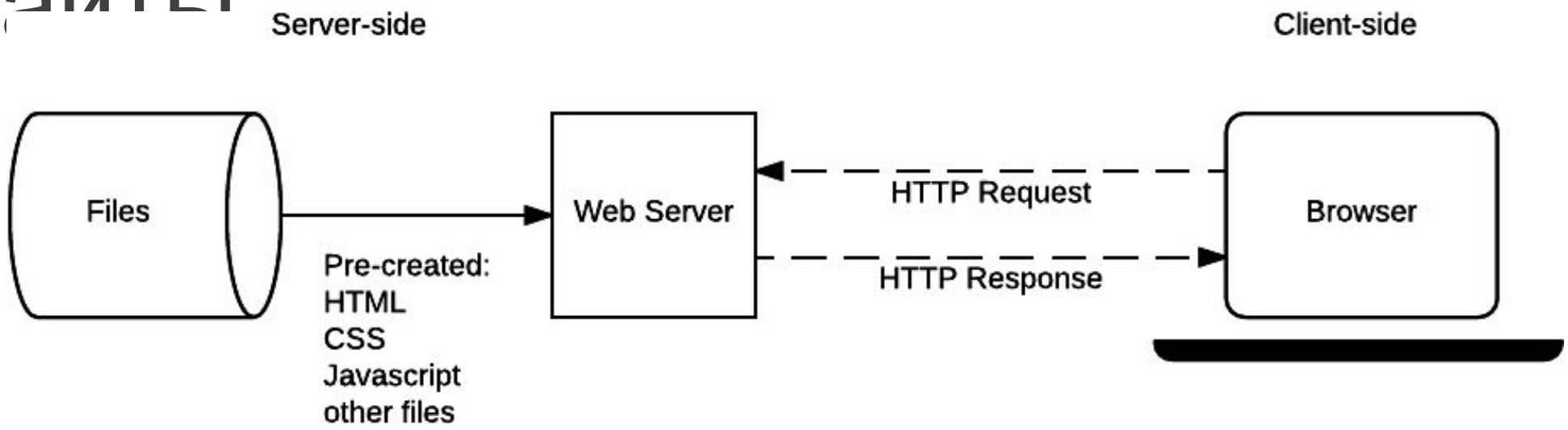
Статические сайты

Статический сайт — это тот, который возвращает тот же жёсткий кодированный контент с сервера всякий раз, когда запрашивается конкретный ресурс.

Статические сайты подходят тогда, когда количество страниц мало и необходимо отправлять один и тот же контент каждому пользователю. Однако их обслуживание может потребовать значительных затрат по мере увеличения количества страниц.

Когда пользователь хочет перейти на страницу, браузер отправляет HTTP- запрос GET с указанием URL-адреса его HTML-страницы. **Сервер** извлекает запрошенный документ из своей файловой системы и возвращает HTTP- ответ, содержащий документ и код состояния.

Статические сайты



Серверу для статического сайта нужно будет только обрабатывать GET- запросы, потому что сервер не сохраняет никаких модифицируемых данных. Он также не изменяет свои ответы на основе данных HTTP-запроса (например, URL-параметров или файлов cookie).

Динамические сайты

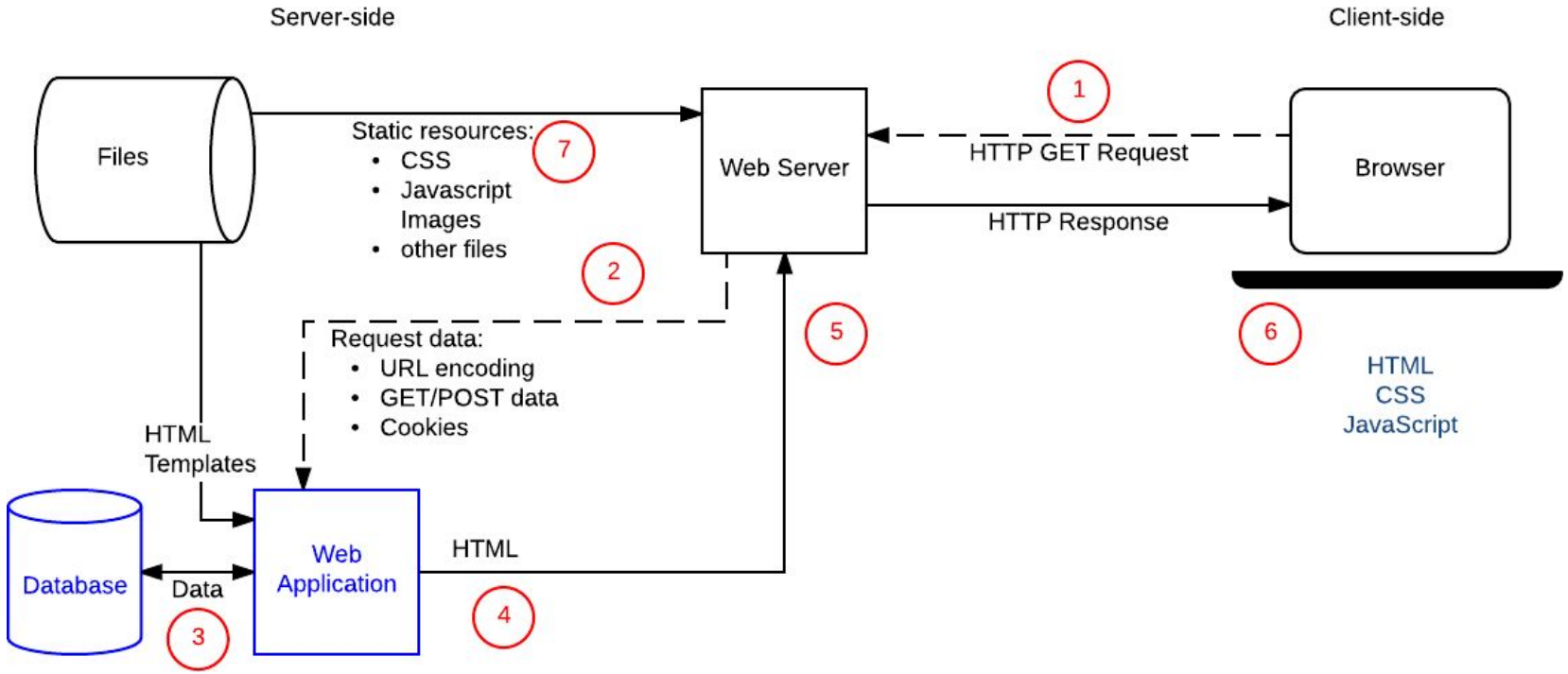
Динамический сайт — это тот, который может генерировать и возвращать контент на основе конкретного URL-адреса запроса и данных (а не всегда возвращать один и тот же жёсткий код для определённого URL-адреса).

Это имеет большие преимущества перед статическим сайтом:

- Использование базы данных позволяет эффективно хранить информацию с
  помощью легко расширяемого, изменяемого и доступного для поиска способа.
- Использование HTML-шаблонов позволяет очень легко изменить структуру HTML, потому что это нужно делать только в одном месте, в одном шаблоне, а не через потенциально тысячи статических страниц.

Динамические сайты

Частями сайта, которые делают его динамичным, являются веб-приложение, база данных и шаблоны. HTML-



Серверные веб-фреймворки

Серверные веб-фреймворки (или «фреймворки веб-приложений») — это программные среды, которые упрощают создание, масштабирование и поддержку веб-приложений. Они предоставляют инструменты и библиотеки, которые упрощают общие задачи веб-разработки, включая маршрутизацию URL-адресов для соответствующих обработчиков, взаимодействие с базами данных, поддержку сеансов и пользователей, форматирование вывода (например, HTML, JSON, XML) и авторизацию, а также улучшение защиты от веб-атак.

Они делают написание кода для обработки большого количества различных операций намного проще.

Возможности веб-фреймворков

Работа с HTTP-запросами и ответами напрямую

Веб-фреймворки позволяют писать упрощённый синтаксис, который будет генерировать серверный код для работы с запросами и ответами. Это означает, что будет легче работать, взаимодействуя с более простым кодом более высокого уровня.

```
# Django view function
from django.http import HttpResponse

def index(request):
    # Get an HttpRequest (request)
    # perform operations using information from the request.
    # Return HttpResponse
    return HttpResponse('Output string to return')
```

Возможности веб-фреймворков

Сопоставление маршрута к соответствующему обработчику

Фреймворки предоставляют простые механизмы для сопоставления шаблонов URL-адресов с конкретными функциями обработчика. Этот подход имеет преимущества с точки зрения обслуживания - можно изменить URL-адрес, используемый для доставки определённой

```
@app.route("/")
def hello():
    return "Hello World!"
```

```
urlpatterns = [
    url(r'^$', views.index),
    # example: /best/myteamname/5/
    url(r'^best/(?P<team_name>\w.+?)/(?P<team_number>[0-9]+)/$', views.best),
]
```

Возможности веб-фреймворков

Упрощенный доступ к данным в запросе

Данные могут быть закодированы в HTTP-запросе разными способами. HTTP-запрос GET может кодировать, какие данные требуются в URL-параметрах или в структуре URL. HTTP-запрос POST будет включать обновлённую информацию внутри тела запроса. HTTP-запрос может также включать информацию о текущей сессии или пользователе в cookie со стороны клиента.

Веб-фреймворки предоставляют соответствующие языку программирования механизмы доступа к этой информации.

Возможности веб-фреймворков

Упрощенный доступ к базе данных

Веб-фреймворки часто предоставляют слой базы данных, который абстрагирует операции чтения, записи, запроса и удаления базы данных. Этот уровень абстракции называется Object-Relational Mapper (ORM).

Это дает следующие преимущества:

- Вы можете заменить лежащую в основе базу данных без необходимости изменять код, который её использует.
- Может быть реализована проверка данных. Это позволяет легче и безопаснее проверить, что данные имеют правильный формат (например, адрес электронной почты) и не являются вредоносными.

Возможности веб-фреймворков

Упрощенный доступ к базе данных

```
from django.db import models

class Team(models.Model):
    team_name = models.CharField(max_length=40)

    TEAM_LEVELS = (
        ('U09', 'Under 09s'),
        ('U10', 'Under 10s'),
        ('U11, 'Under 11s'),
        ...  #list our other teams
    )
    team_level = models.CharField(max_length=3,choices=TEAM_LEVELS,default='U11')
```

Возможности веб-фреймворков

Упрощенный доступ к базе данных

```
from django.shortcuts import render
from .models import Team

def youngest(request):
    list_teams = Team.objects.filter(team_level__exact="U09")
    context = {'youngest_teams': list_teams}
    return render(request, 'best/index.html', context)
```

Возможности веб-фреймворков

Визуализация данных

Веб-фреймворки часто предоставляют системы шаблонов. Они позволяют вам указать структуру выходного документа, используя заполнители для данных, которые будут добавлены позже. Шаблоны часто используются для создания HTML, но могут также создавать другие типы документов.

Веб-фреймворки часто предоставляют механизм, позволяющий легко создавать другие форматы из хранимых данных, включая JSON и XML.

```
<!DOCTYPE html>
<html lang="en">
<body>

 {% if youngest_teams %}
    <ul>
    {% for team in youngest_teams %}
        <li>{{ team.team_name }}</li>
    {% endfor %}
    </ul>
{% else %}
    <p>No teams are available.</p>
{% endif %}

</body>
</html>
```

Выбор веб-фреймворка

Существует множество веб-фреймворков для различных языков программирования. При таком большом количестве вариантов может оказаться затруднительным определить, какой фреймворк обеспечивает лучшую отправную точку для вашего нового веб-приложения.

Вот некоторые из факторов, на которые стоит обращать внимание:

- Усилия для изучения
- Производительность
- Производительность фреймворка / языка программирования
- Поддержка кеширования
- Масштабируемость
- Веб-безопасность

Популярные веб-фреймворки

- Django (Python)
- Flask (Python)
- Express (Node.js/JavaScript)
- Ruby on Rails (Ruby)
- ASP.NET
- Mojolicious (Perl)
- Spring Boot (Java)
- Laravel (PHP) и т.д.