Разработка серверного ПО. Лекция 1. Сервисориентированная архитектура

Подход при котором, сервер может служить клиентом для другого сервера, используемый для декомпозиции прложений на сервисы по областям функциональности, называется `сервисориентированной архитектурой` (SOA, микросервисная архитектура — переоткрытие SOA).

В сервисориентированной архитектуре важно следить за совместимостью форматов кодирования данных (совместимостью разных API).

`API` — интерфейс прикладного программирования. Описывает слой абстракции над реализацией, спецификацию и семантику (поведение). Делятся на два типа: в процессе ОС (вызов метода) и вне процесса (например, сетевой запрос).

`Веб-сервис` — сервис для связи с которым используется протокол `HTTP`.

`Подходы к проектированию веб-сервисов:`

- REST. Акцент на простых форматах данных, применение URL и возможностей HTTP. RESTful — API спроектированный в соответствии с принципами REST.
- SOAP — основан на XML; имеет множество стандартов; независим от HTTP; языки WSDL не предназначены для чтения людьми; WSDL генерируется вызовом методов библиотек. (Пример WSDL 1.1).
- RPC(удалённый вызов процедур) — обращение к серверу выглядит как вызов функции или метода на обычном языке программирования. За счёт введения дополнительного слоя абстракции не зависит от расположения (location transparency). Но имеет фундаментальные отличия от локального вызова функции или метода:
    - Локальный вызов предсказуем — завершается успешно (возврат результата) или нет (генерация исключения).
    - Локальный вызов предсказуем по времени.
    - Нет проблем с представлениями данных (в разных языках программирования даже представление примитивных типов может отличаться).

REST используется для реализации общедоступных API. RPC используется для реализации взаимодействия внутри одной организации (даже ЦОДа).

Протокол HTTP

Так как рассматриваем веб-сервисы, то необходимо рассмотреть протокол HTTP как протокол по которому происходит взаимодействие сервисов друг с другом.

Для указания пути к сервису используется URI (uniform resource identifier), который описывает URL(uniform resource location) — имеет информацию о механизме доступа к ресурсу, и URN(uniform resource name) — не включает в себя всю информацию о нахождении и способе обращения. Объект к которому идёт обращение по URI называется `ресурсом`.

URI схема описана в RFC3986 раздел 3

```
scheme "://" authority "/" path ["?" query] ["#" fragment]
```

URI правила:

- терминальный "/" не должен включаться в URI.
- "-" можно использовать для повышения читабельности.
- "_" нельзя использовать в URI.
- Предпочтителен нижний регистр.
- Не включать расширение файла в URI.

Архетипы ресурсов:

1. Документ — единичный ресурс.
2. Коллекция — множество ресурсов.
3. Хранилище — управляемое пользователем множество ресурсов.
4. Контроллер — процедура/исполняемая функция. Не имеет дочерних узлов.

Модель пути в URI:

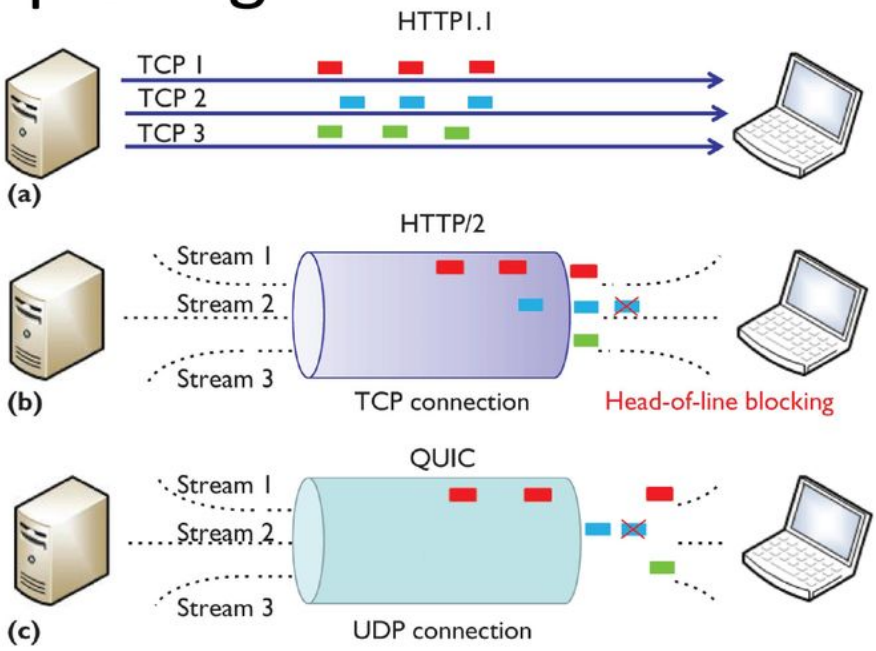
```
collection/store/document
```

HTTP — клиент-серверный протокол для взаимодействия отдельными сообщениями. Клиент отправляет запросы, сервер отправляет ответы.

HTTP-поток:

1. Открытие TCP соединения.
2. Отправка HTTP сообщения.
3. Чтение ответа.
4. Закрытие или переиспользование TCP соединения.

Multiplexing



`Особенности:`

- Сообщения человекочитаемы.
- Протокол расширяем за счёт заголовков.
- Не имеет состояния, но можно реализовать сессию (через куки).

`Наиболее распространённые возможности HTTP:`

- Кеширование как на стороне клиента, так и на стороне сервера.
- Ослабление ограничений источника.
- Аутентификация (заголовок WWW-Authenticate).
- Прокси и туннелирование.
- Сессии.

Сообщения http

Методы http запроса

Коды ответов http

`Когда какие методы использовать?`

- `GET` для получения представления ресурса.
- `HEAD` для получения заголовков ресурса.
- `PUT` для добавления нового ресурса или модификации существующего.
- `POST` для добавления нового ресурса в хранилище или для вызова контроллеров.
- `DELETE` для удаления ресурса из родителя.
- `OPTIONS` для отправки метаданных, описывающих способы взаимодействия с ресурсом. Например, клиент отправляет запрос с заголовком `Allow` и может получить детальную информацию о способах взаимодействия с ресурсом.

REST (REpresentation State Transfer)

Архитектурные ограничения REST:

- Клиент-серверная архитектура.
- Отсутствие состояния — вся нужная информация передаётся вместе с запросом.
- Кешируемость — запросы, возвращающие всегда одинаковые ответы, должны кешироваться на стороне клиента. Информация о возможности кеширования должна содержаться в ответе.
- Многослойность системы — клиент не должен знать о сложности системы.
- Код по запросу — сервер может передавать исполняемый код (например js) клиенту (необязательное условие).
- Единообразие интерфейса — все взаимодействия основаны на концепции идентифицированных ресурсов, которыми манипулируют стандартными методами. Взаимодействия должны предоставлять все метаданные, которые необходимы для интерпретации ресурса и, что с ним можно сделать.

HATEOAS (Hypermedia as the engine of application state). При запросе клиентом ресурса, в ответ включается список связанных ссылок. Если клиент запрашивает детали заказа, то в ответе будут ссылки на отмену и оплату.

На практике HATEOAS редко реализуется:

- Информация есть в документации, которая богаче и лучше структурирована.
- Не всегда понятно какие ссылки возвращать, например, у пользователей могут быть разные права.
- В зависимости от состояния ресурса, могут быть недоступны некоторые действия и ресурсы.
- Список ссылок может быть слишком большим, что будет сильно увеличивать размер ответа.

Модель зрелости Ричардсона

Уровень 0. Веб-API в стиле RPC. HTTP используется как транспортный протокол; API предоставляет единый URI для запросов, которые выполняются одним методом, а детали запроса передаются через тело запроса.

Уровень 1. Введение понятия ресурса. Вместо общего эндпоинта (URI) сервер предоставляет URI-адреса, соответствующие ресурсам.

Уровень 2. Использование методов и статус-кодов. Методы HTTP указывают, что нужно сделать с ресурсом, а статус-коды информируют о результатах обработки запроса.

Уровень 3. Средства управления гипермедиа. Реализован HATEOAS. Уровень используется редко.

JSON (JavaScript Object Notation)

XML

```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40></age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

JSON

```
{  "empinfo"  :
      {
            "employees" :  [
            {
                  "name" : "James Kirk",
                  "age" : 40,
            },
            {
                  "name" : "Jean-Luc Picard",
                  "age" : 45,
            },
            {
                  "name" : "Wesley Crusher",
                  "age" : 27,
            }
                              ]
      }
   }
```