Знакомство с REST

JSP & Servlets
9 уровень, 7 лекция

ОТКРЫТА

8.1 Подход Remote API

Все программисты делают одну и ту же ошибку при построении архитектуры клиент-сервер. Они начинают воспринимать запросы к серверу, как вызов методов.

Ты хочешь запустить процесс генерации отчета на сервере, почему бы не отправить ему запрос вида:

```
http://server.com/startDocumentGeneration?params
```

А как загрузить отчет после того, как он завершится? Для этого напишем еще один метод:

```
http://server.com/getDocument
```

Сервер в `HttpSession` хранит информацию по нашему документу, и как только документ будет сгенерирован, сервер его отдаст.

Отличный подход. Или нет?

Подход на самом деле ужасный. Все дело в том, что сервер должен запомнить номер документа. Другими словами, чтобы правильно обрабатывать новые вызовы методов, сервер должен запомнить результаты вызова предыдущих методов.

В вебе это большая проблема. Интернет может пропасть, браузер закрыться. Страницу можно перегрузить или случайно кликнуть по ссылке и так далее. А сервер будет продолжать хранить мегабайты данных из предыдущих запросов пользователя…

Для комфортной работы с сервером вы не можете рассчитывать, что у вас всегда под рукой будут данные предыдущих запросов к серверу.

Как же тогда вызывать методы сервера? Правильный ответ будет ужасен: а никак!

8.2 Подход REST

Программисты вернулись к истокам и вспомнили, что изначально запрос содержал путь к файлу на сервере:

```
http://server.com/path?params
```

И решили использовать этот подход по максимуму.

Теперь сервер рассматривается как хранилище данных, которые видны наружу в виде некоторого дерева.

Хотите получить список всех пользователей, вызывайте запрос:

```
http://server.com/users
```

Хотите получить данные по пользователю 113, выполняете запрос:

```
http://server.com/users/113
```

И так далее, все в том же ключе.

Еще раз, сервер рассматривается как хранилище данных, которые видны наружу в виде некоторого дерева.

Данные можно получать – запросы **GET**, изменять – запрос **POST** и удалять – запрос **DELETE**.

8.3 Отсутствие состояния

REST-протокол взаимодействия между клиентом и сервером требует соблюдения следующего условия: в период между запросами от клиента никакая информация о состоянии клиента на сервере не хранится.

Все запросы от клиента должны быть составлены так, чтобы сервер каждый раз получил **всю необходимую информацию для выполнения запроса**. Состояние сессии при этом сохраняется на стороне клиента.

Во время обработки клиентских запросов считается, что клиент находится в переходном состоянии. Каждое отдельное состояние приложения представлено связями, которые могут быть задействованы при следующем обращении клиента.

8.4 Единообразие интерфейса

Все пути, по которым отдаются объекты с сервера, стандартизируются. Это очень удобно, особенно если ты получаешь данные из чужих REST-серверов.

Все интерфейсы объектов должны соблюдать три условия:

Идентификация ресурсов

Все ресурсы идентифицируются в запросах с использованием URI. Ресурсы внутри сервера отделены от представлений, которые возвращаются клиентам. Например, сервер может отсылать данные из базы данных в виде HTML, XML или JSON, ни один из которых не является типом хранения внутри сервера.

Манипуляция ресурсами через представление

Если клиент хранит представление ресурса, включая метаданные, то он обладает достаточной информацией для модификации или удаления ресурса на сервере.

"Самоописываемые" сообщения

Каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать. Например, если вам нужна информация о пользователе, то сервер вернет вам JSON объект, где будут поля имя, адрес.

Не должно быть ситуации, когда клиент должен знать, что первое число в ответе — это возраст, а второе – это дата рождения.

8.5 Кэширование

Подход REST предполагает, что запросы к данным идет посредством HTTP протокола. Поэтому объекты получаются посредством вызова GET-запроса. А значит на них, как и на все ресурсы, полученные посредством GET-запроса, распространяются все правила кэширования HTTP-ресурсов.

То есть данные, полученные через REST API, кэшируются так же, как и любые статические ресурсы на веб-серверах. Красота :)

+11

Комментарии (2) популярные новые старые

JavaCoder

Введите текст комментария

Oleg Khilko Уровень 51 14 августа, 10:21

"Данные можно получать – запросы GET, изменять – запрос POST и удалять – запрос DELETE."

Про GET и DELETE говорить нечего - скучно и идемпотентно. Правда на повторном вызове DELETE будет не 200, а 404, но допустим это ок.

Если про POST вы говорите что это "изменять", то как вы сравните POST vs PUT vs PATCH?

По сути про них всех можно сказать что они "изменяют", но, как в анекдоте про Петьку и Василия Ивановича, - "есть нюанс".

Давайте будем изначально людям закладывать верный вектор движения, чтобы они не считали что раз POST/PUT/PATCH изменяют что-то, то они равны между собой, что в корне неверно.

В конце концов не зря же люди целых три HTTP метода придумали, а не один.

Ответить +7

wf Уровень 33 9 августа, 17:47

На странице есть заглушка для какой-то картинки
"<sp an> image </sp an>"
Так и должно быть?

Ответить +2

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store