Лекция 13

ПОИТ-3, ПСКП

**REST**

1. **REST**: **Representational State Transfer** - передача состояния представления.
2. **REST**: ресурс (имеет URI), состояние, представление, управление состоянием ресурса с помощью глаголов (GET, POST, PUT, DELETE).
3. **REST**: архитектурный стиль взаимодействия компонентов распределенного приложения.
4. **REST**: автор **Roy Fielding**, описан в диссертации «Архитектурные стили и дизайн сетевых программных архитектур», 2000 г.
5. **REST**: альтернатива RPC.
6. **RESTful**: описание web-службы, поддерживающей REST-интерфейс в полном объеме (со строгим соблюдением ограничений).
7. **REST**: нет официального стандарта, но REST использует стандарты HTTP, URL/URI, XML, JSON.
8. **REST**: шесть обязательных ограничений:

* модель клиент-сервер;
* отсутствие состояния на стороне сервера, сохранение состояния допускается на стороне клиента, допускается сохранение состояния в другом сервисе (например, в БД);
* кэширование на стороне клиента, сервер явно управляет кэшированием;
* единообразие интерфейсов (идентификация ресурсов, манипуляция ресурсами через представления, самодостаточные сообщения, HATEOAS);
* для клиента сервер должен представляться конечным;
* код по требованию: допускается (необязательно) выгрузка на клиент апплетов или сценариев для расширения его функциональности.

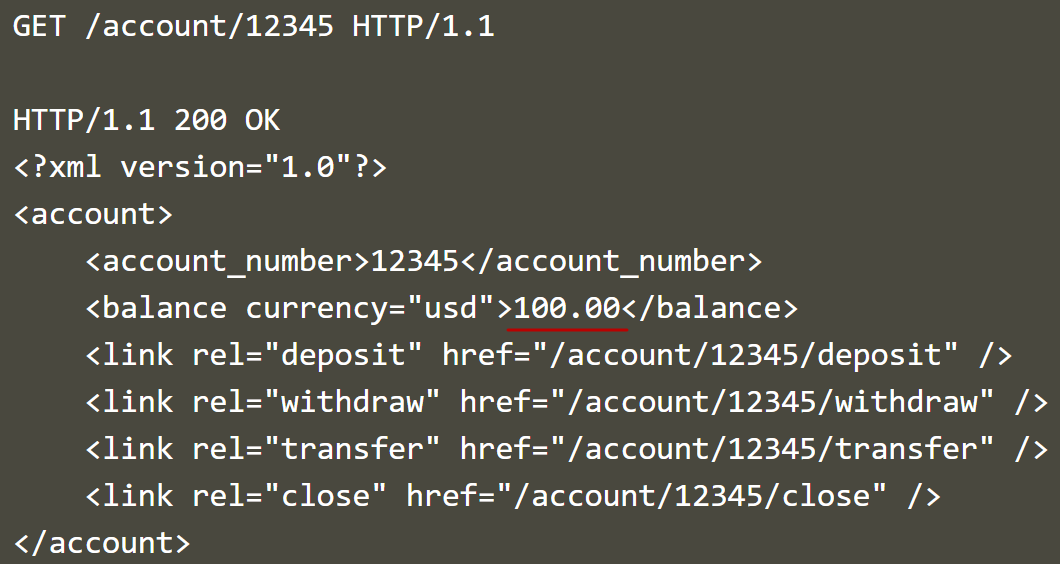
1. **REST:** GET (select), POST (insert), PUT (update), DELETE (delete).

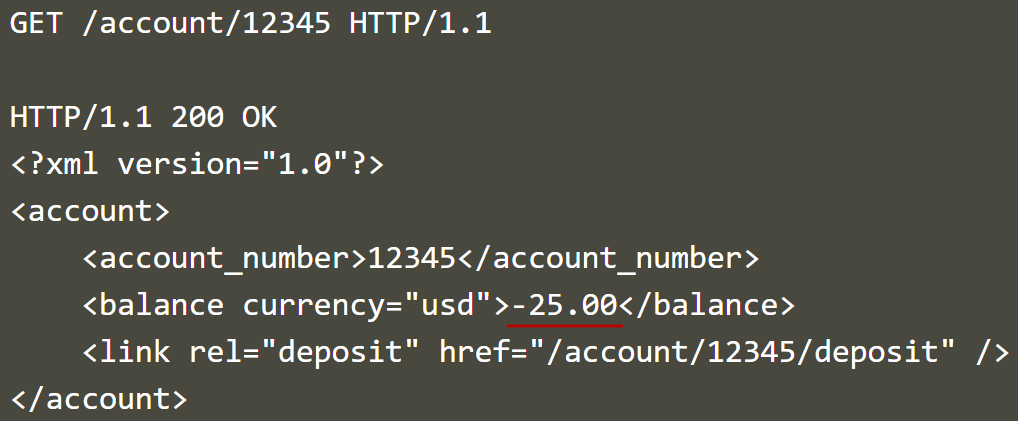


1. **HATEOAS: Hypermedia As The Engine Of Application State** – гипермедиа в качестве управления состоянием.
2. **Гипермедиа**: технология обработки, структурирования информации и произвольного доступа к ее элементам с помощью гиперсвязей (Тед Нильсон, 1965), WWW – реализация гипермедиа.
3. **HATEOAS: пример**



1. **HATEOAS: пример**





1. **REST:** **общепринятые правила**
   1. Общий префикс для всех ресурсов сервиса …/api/…

http://bstu.by/api/...

* 1. Два типа ресурсов: коллекция (users, students, …), элемент коллекции …/api/users/238, …/api/students/ef3d26.
  2. Иерархическая связь …/api/users/238/cars/aah4899
  3. Использовать существительные во множественном числе. Если несколько слов, то использовать kebab-case

…/api/customers/33245/delivery-addresses

* 1. Использовать HTTP статус коды, сопроводить сообщение дополнительным кодом (например 20003, 404001,…), сделать отдельный ресурс (HATEOAS link) для пояснения ошибок <http://ccc/api/errors/20003>.
  2. Подавление статуса ответа

…/api/students/ef3d26?status\_code=200.

* 1. Версионность

…/api/v7/students/ef3d26 или …/api/students/ef3d26?v=7.

* 1. Постраничное получение данных: параметры limit, offset.

…/api/students?offset=10&limit=5

* 1. Сортировка: параметр sort.

…/api/students?sort=+group,+name

* 1. Все фильтры вынести за знак вопроса: …/api/students?minbday=19980101&maxbday=20001231&gender=m.
  2. Пользователь получает только то, что хочет: …/api/students?field=bday,surname,gender.
  3. Обозначать в запросе формат сообщений (желательна поддержка нескольких форматов, рекомендуется JSON и XML):

…/api/students.json?field=bday,surname,gender; один из форматов должен быть по умолчанию; могут применяться заголовки Accept и Content-Type со значениями application/xml и application/json для запроса или обозначения в ответе формата.

* 1. Глобальный поиск: …/api/search?q=19600107+Иванов.
  2. Документация.

1. **REST:** достоинства

* производительность (кэширование);
* надежность (отсутствие состояния);
* простота (унифицированность интерфейса, использование повсеместных стандартов,…);
* изменяемость;
* масштабируемость.

1. **REST:** недостатки

* нет общепризнанного стандарта RESTful API;
* не все браузеры поддерживают полный словарь REST-методов (PUT, DELETE); на практике часто используется только GET и POSТ(insert, delete, update);
* не однозначны коды состояний.

