ПWS

ПОИТ-4

Лекция 2

**REST API**

1. **REST**: **Representational State Transfer** - передача состояния представления.
2. **REST**: представление – это URI (идентификатор ресурса), управление ресурсом 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**: реализация (альтернатива) RPC.
6. **RESTful**: описание web-службы, поддерживающей REST-интерфейс в полном объеме (со строгим соблюдением правил).
7. **REST**: нет официального стандарта, но REST использует стандарты HTTP, URL/URI, XML, JSON.
8. **REST:** два основныхтипа ресурса – коллекции и элемент коллекции: /api/users, /api/users/288.
9. **REST**: шесть обязательных ограничений:

* **модель клиент-сервер**;
* **отсутствие состояния на сервере**: сохранение состояния на стороне клиента или в другом сервисе (например, в БД);
* **кэширование на стороне клиента**, сервер явно управляет кэшированием;
* **единообразие интерфейсов** (уникальные URL, стандартные запросы, самодостаточные сообщения, HATEOAS);
* **сервер как единое целое**: клиент видит сервер как одну систему, не зная внутренностей.
* **код по требованию**: сервер может отправлять клиенту дополнительный код для расширения функционала.

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



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

<http://API.BSTU.BY/>...

* 1. Два типа ресурсов: коллекция (users, students, …), элемент коллекции /api/users/238, /api/students/ef3d26.
  2. Иерархическая связь …/api/users/238/cars/aah4899
  3. Ограничить количество HTTP-статусов, сопроводить сообщение дополнительным кодом (например 20003, 404001,…),сделать отдельный ресурс (HATEOAS link) для пояснения ошибок <http://ccc/api/errors/20003>.
  4. Подавление статуса ответа …/api/students/ef3d26?status\_code=200.
  5. Версионность /api/students/ef3d26?v=7.
  6. Постраничное получение данных: параметры limit, offset.
  7. Сортировка: параметр sort.
  8. Все фильтры вынести за знак вопроса: …/api/students?minbday=1998101&maxbday=20001231&gender=m.
  9. Пользователь получает только то, что хочет: …/api/students?field=bday,surname,gender.
  10. Обозначать в запросе формат сообщений (желательна поддержка нескольких форматов):

…/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.

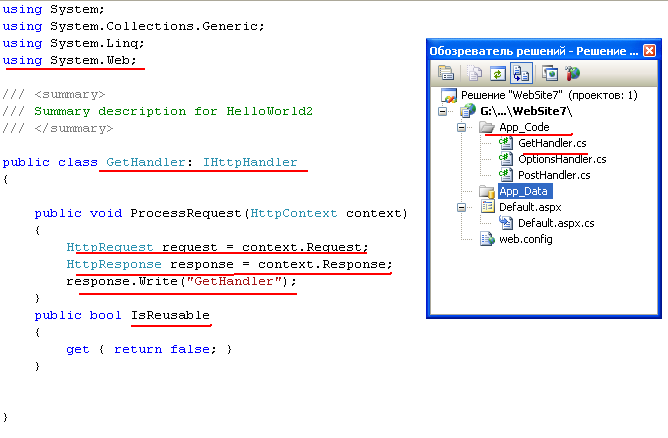
* 1. Глобальный поиск: ../api/search?q=19600107+Иванов.
  2. Документация.

1. **REST:** недостатки

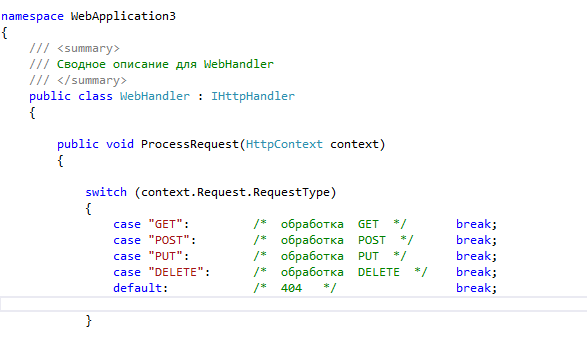
* нет общепризнанного стандарта RESTful API;
* не все браузеры поддерживают полный словарь REST-методов (PUT, DELETE); на практике часто используется только GET и POSТ(insert, delete, update);
* не однозначны коды состояний.



1. **REST: создание REST-службы на базе ASP.NET HTTP Handler**

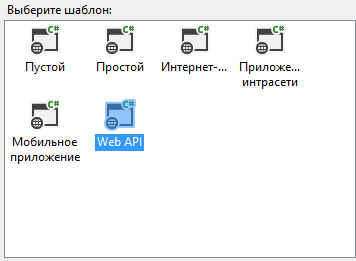
****

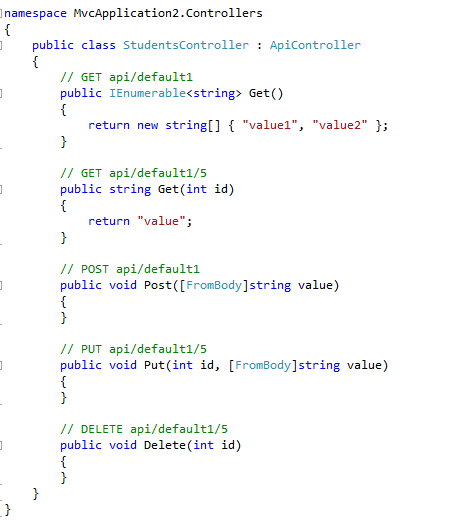
****

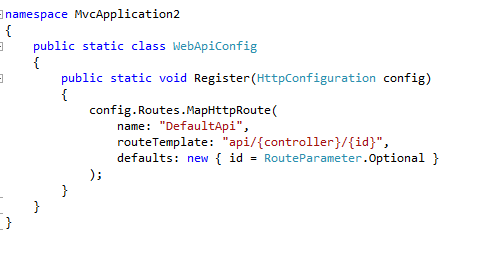
****

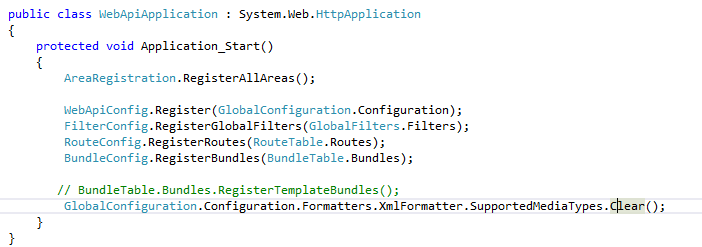
****

1. **REST: создание REST-службы на базе ASP.NET MVC WEB API**

****

****

****

****