Building the API – Implementing Basic Requirements

Kevin Dockx

@KevinDockx | http://blog.kevindockx.com/

The API should be friendly to consume & should be consumable from different client types

Designing Resource URI's

api/expensegroups

api/expensegroups/1

api/expensegroups/1/expenses

api/expensegroups/1/expenses/1

api/expenses/1

Designing Resource URI's

expensegroup

total

api/expensegroups/1

expensegrouptotal

api/expensegrouptotals/1

expensegroup

api/expensegroups/1/total

Interacting with Resources

HTTP Status Codes

- POST: 201 (created), 400 (bad request), 500 (internal server error)
- GET: 200 (ok), 404 (not found), 400, 500
- DELETE: 204 (no content), 404, 400, 500
- PUT: 200, 404, 400, 500
- PATCH: 200, 404, 400, 500

- General: 401 (unauthorized), 403 (forbidden), 405 (method not allowed)

Important Demo Setup Instructions

Copy over the DB files (ExpenseTrackerDB.mdf & ExpenseTrackerDB_log.ldf) to the App_Data folder of the API project

… alternatively, use the scripts & change the web.config

Introduction & Retrieving Resources

Solution and Web API overview

Learn how to retrieve a list of resources: GET

Formatters and Result Formatting

Learn how to support JSON and/or XML

Learn how to format the results

Retrieving a Single Resource

Learn how to retrieve a single resource: GET

Creating a Resource

Learn how to create a resource:
POST

Updating a Resource

Learn how to update a resource:
PUT

A Few Words on PATCH

PATCH is for partial updates

- IETF: https://tools.ietf.org/html/rfc6902
- JsonPatchDocument describes a sequence of operations to apply to a JSON document.
- Content-Type: application/json-patch+json
- NuGet Package: **Marvin.JsonPatch**

```
[
  { "op": "replace", "path": "/a/b/c",
    "value": "foo" },
  { "op": "remove", "path": "/a/b/c" },
  { "op": "copy", "from": "/a/b/d",
    "path": "/a/b/e" }
]
```

Partially Updating a Resource

Learn how to partially update a resource: PATCH

Deleting a Resource

Learn how to delete a resource:
DELETE

Relations and URI Mapping

Learn how to map URI's to support relations

Learn how to map two URI's to the same code

The API should support sorting

Sorting

Learn how to sort by different fields, ascending and descending

The API should support filtering

Filtering

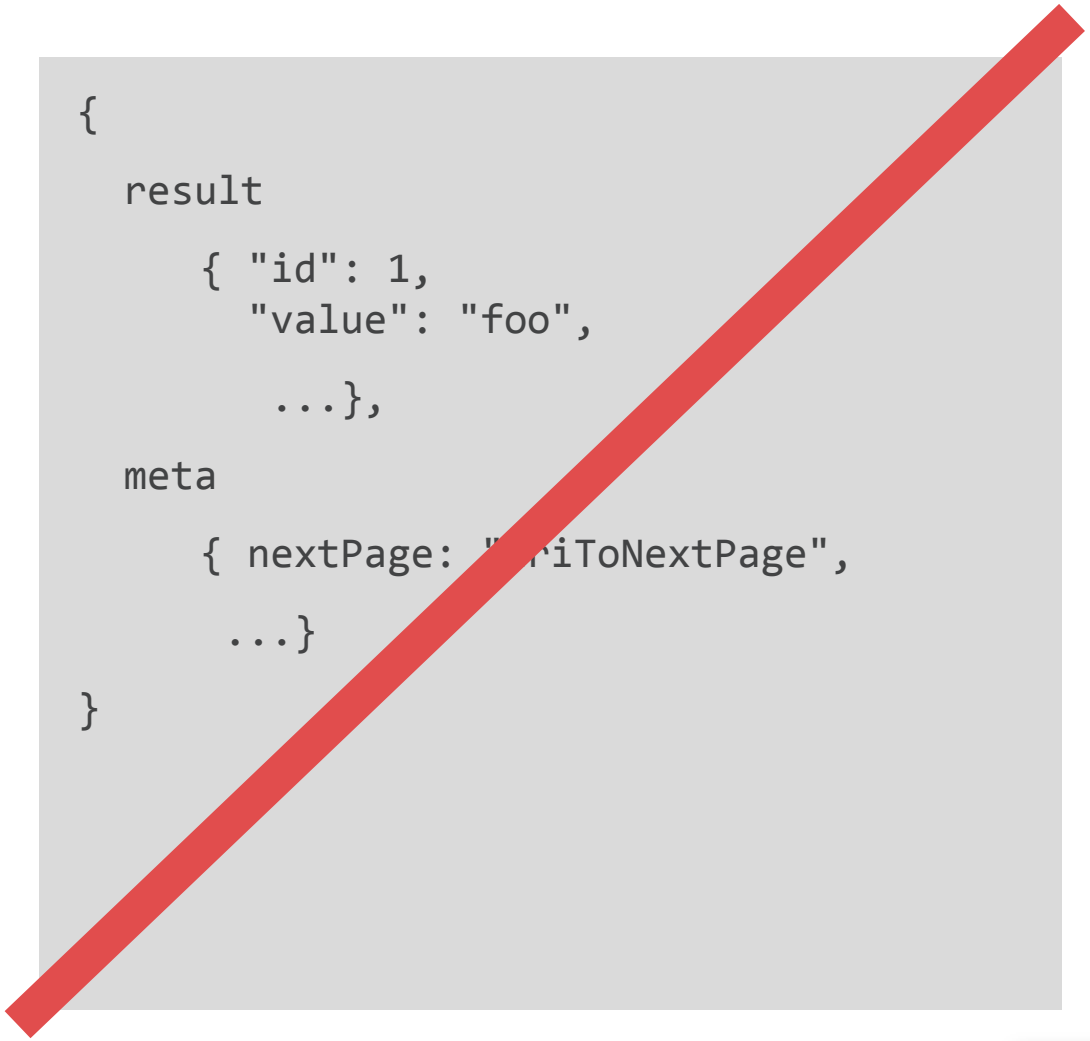
Learn how to filter lists of resources

The API should support paging

Implementing Paging Support

- At the very least, return URI's for previous / next page
- Optionally, include additional information like total count and total amount of pages
- Optionally, include page number and page size

- Add paging information to the response **header**!



Paging

Learn how to implement paging

Summary

We use nouns to describe our resources, HTTP verbs to interact with them

We should return correct HTTP Status codes

Sorting, filtering and paging is requested through the URI

Pagination info should go in the response header