

Building the API - REST and Web API Primer



Kevin Dockx

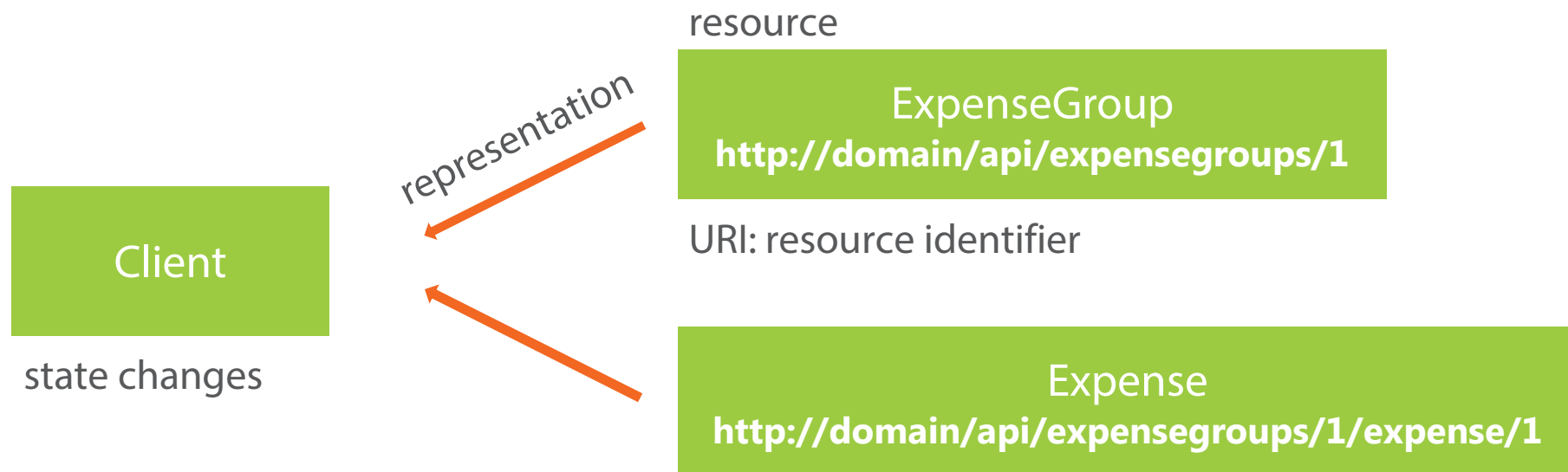
@KevinDockx | <http://blog.kevindockx.com/>


REST

Representational State Transfer

Architectural style of networked systems (not a standard!), coined by **Roy Fielding**
(<http://bit.ly/1rbtZik>)

Intended to evoke an image of how a well-designed web application behaves





Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.

— Roy Fielding



Architectural Constraints

To be called REST-ful, the API/system should adhere to these constraints

Uniform Interface

Statelessness

State to handle the request is contained within the request itself

Client-Server

Client and server are separated

Cacheable

Each response must define itself as cacheable or not

Layered System

A client cannot tell whether it's directly connected to the end server

Code on Demand

Server can extend/customize client functionality

Architectural Constraints

Uniform Interface

Identification of resources

resources are identified through a URI
resource \neq representation (often, JSON)

`http://domain/api/expensegroups/1`

`http://domain/api/expensegroups/1/expenses`

`http://domain/api/expensegroups/1/expenses/1`

HTTP GET, POST, PUT, PATCH, DELETE

Manipulation of resources through representations

Enough info to modify/delete the resource on the server

Self-descriptive message

Message must include enough info to process the message

HATEOAS (Hypermedia as the Engine of Application State)

A Few Words on HATEOAS

Hypermedia as the Engine of Application State

Hypermedia

Generalization of hypertext
Adds other media types

drives how to consume & use the API

Self-documenting, very dynamic API

Allows changes to API without having to
rewrite the client

The Case for Pragmatism



Not adhering to all constraints does not a bad API make

Don't get lost in adhering to them, no matter what the cost

Positioning ASP .NET Web API



ASP.NET

WEB API

- **ASP.NET Web API** is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices
- It's not because you're using the Web API framework that you're building a REST-ful API

NuGet Package: `Microsoft.AspNet.WebApi`

Summary



REST is an architectural style

- Uniform Interface
- Statelessness
- Client-Server
- Cacheable
- Layered System
- (Code on Demand)

Pragmatism is not a swear word

ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients