5 Anti-Patterns in API DESIGN

Ali Kheyrollahi
@aliostad
API Iceberg

“This huge mass underneath the water that you can't see, the private API, is the biggest part of the whole opportunity.”

Daniel Jacobson, Netflix - 2011
“… the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API”

Martin Fowler - Bliki
REST Stateless
REST Layered Architecture
REST Caching
REST Uniform Interface

Resources and representations

Self-descriptive messages

Hypermedia
REST Client-Server
HTTP Client-Server

**Server Concern:** e.g. returning the resource when requested with a different casing
/car/123 vs /Car/123

**Client Concern:** e.g. Handling 301 (moved permanently)

**Mixed Concern:** e.g. Most HTTP concepts such as Content Negotiation or Caching/Concurrency
“Client and server must define and live within their own bounded context”
CSDS Client-Server Domain Separation

Server

- Workflows
- Messages
- Storage
- Processes
- Data
- Service bus

Public Domain

API

Domain objects
API Restaurant
CSDS: Client-Server Domain Separation

**Client**

1. Can be a server itself
2. Uses services of server(s) to bring value to end-user (directly or indirectly)
3. Free to take dependency on server’s public domain (URI, exchange domain)
4. Normally keeps state but does not master it
Anti-Pattern Transparent Server
Anti-Pattern: Transparent Server

“server exposes its internal implementation to its clients”

server’s private domain or the domain of its underlying dependencies **bleeds** into its public API
Example 1

Anti-Pattern: Transparent Server

Creating

You can make a POST request to **photos** edge from the following paths:
- `/{page_id}/photos`

<table>
<thead>
<tr>
<th>nectar_module</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nectar module. Internal apps only</td>
</tr>
</tbody>
</table>
Anti-Pattern: Transparent Server

Example 2

1. Always for a customer
2. Only for customers currently shopping
3. Get expired after inactivity
4. Max one basket per customer

A couple of tables or a document database
Anti-Pattern

Transparent Server

Example 2

POST /baskets?cid=908
201 Created
Location: /baskets/123435455456

POST /baskets/123435455456
{"..."}
200 OK
Anti-Pattern

Transparent Server

Example 2

POST customer/me/basket

{...}

200 OK

✓

Shopping Basket service
“designing the API from server's perspective”

Server pushes its thinking and process to the client resulting in the client becoming a subordinate
Anti-Pattern Chauvinist Server

HATEOAS
(and not hypermedia itself)

Hypermedia as the Engine of Application (state)
Don’t try to spare the client composing URLs

**Microservices**: servers smaller, containing a couple of resources
Anti-Pattern Chauvinist Server

HATE OAS?
Anti-Pattern Demanding Client
Anti-Pattern Demanding Client

“client enforces its special needs onto the API signature”

certain clients limitations (or reluctance to implement) become server’s default behaviour
Anti-Pattern: Demanding Client

Examples

1. Client enforces use of **query string** over HTTP headers
2. Client pushes for consistency of **parameter** names with other [external] APIs
3. Client pushes for consistency of **behaviour** with other [external] APIs
4. Client asks for **simpler** model since does not need the extra data
Anti-Pattern: Assuming Server
Anti-Pattern Assuming Server

“server assumes the role of defining client experience”

server makes decisions on issues that are inherently client concerns
Anti-Pattern: Assuming Server

Example 1

GET /api/catalogue/products/pages/1
GET /api/catalogue/products/pages/2
Anti-Pattern Assuming Server

Example 1

GET /api/catalogue/products?from=1&count=30
Anti-Pattern: Assuming Server

Example 2

Browser sniffing
Anti-Pattern: Presumptuous Client
Anti-Pattern Presumptuous Client

“client takes on responsibilities that cannot fulfil”

Client presumes it can fulfil some responsibilities that are inherently server’s
Anti-Pattern: Presumptuous Client

Examples

1. Client implements an algorithm that needs to be centralized on server
2. Client act as an authority for authentication or authorisation
3. Client takes control of cache invalidation
ReCap

1. Microservices take importance of APIs to a new level
2. Think of your API as a restaurant and remember the contrast
3. Transparent Server: Exposing internals
4. Chauvinist Server: Client becoming a subordinate
5. Demanding Client: Client enforcing special needs to API signature
6. Assuming Server: Server deciding client experience
7. Presumptuous Client: Client taking responsibilities cannot fulfil
Thank You

http://byterot.blogspot.com

@aliostad