Real World REST - Advanced

International PHP Conference 2015

Tobias Schlitt / @tobySen

2015-06-09

passion for software quality

international PHP2015 conference - spring edition -

REST?

There is no REST

Choose your degree of RESTfulness

`http://qa.fo/trade-offs`

Constraints by Example

- Time & money
- Project spread
- Consumers
- API weighting
- Weighting of REST attributes
  - Scalability
  - Simplicity
  - ...
- Backwards compatibility

Outline

passion for software quality

Resources

- Entities or collections
- Tree structure
- URI as **unique** identifier
- Examples:
  - `http://plan.qafoo.com/users`
  - `/users/toby`
  - `//jobs/23`
  - `../`

Evaluation

Mandatory!

passion for software quality

HTTP Methods

- GET
- HEAD
- OPTIONS
- TRACE
- POST
- PUT
- DELETE
- . . .

Evaluation

- No need to support all methods
- Support at least `GET`
- Simply ignore
  - TRACE
  - Possibly OPTIONS
- Stick to method properties!

Outline

Basics

HATEOAS

Caching

Authentication

Conclusion

Media Types

- Assign Semantic
- Drive Application State

```
application/psr.com.qafoo.plan-job+xml; charset=UTF-8
```

HATEOAS

```
GET /job/23
```

```
<?xml version="1.0"?>
<job xmlns="urn:psr.com.qafoo.plan-job"
    xmlns:atom="http://www.w3.org/2005/Atom">
    <!-- ... -->
    <atom:link rel="urn:psr.com.qafoo.plan-job-assignments"
        type="application/psr.com.qafoo.plan-job-assignment-list+xml"
        href="/jobs/23/assignments" />
</job>
```

...

```
POST /job/23/assignments
```

```
<?xml version="1.0"?>
<assignment xmlns="urn:psr.com.qafoo.plan-job-assignment"
    xmlns:atom="http://www.w3.org/2005/Atom">
    <atom:link rel="urn:psr.com.qafoo.plan-assignee"
        type="application/psr.com.qafoo.plan-user+xml"
        href="/users/benjamin" />
    <days>2</days>
</assignment>
```

Versioning with Media Types

- `application/....plan-job+xml`
- `application/....plan-job-v2+xml`
- `application/....plan-job+xml; version="2"`

Content Negotiation

`Accept:`

- `application/....plan-job+xml`

-
  ```
  application/....plan-job+json,
  application/....plan-job+xml; q=0.5
  ```

-
  ```
  application/....plan-job+xml; version="2",
  application/....plan-job+xml; q=0.5,
  application/*; q=0.2,
  */*; q=0.1
  ```

Evaluation

- Media types
  - 1 type / resources
  - Helps evolving
  - JSON won, sadly
- Links
  - Be prepared: People won’t use them
  - Still good documentation
- Versioning
  - Precondition: Media types
  - Can be added later
- Content Negotiation
  - No client will use it
  - Don’t bail out

Outline

passion for software quality

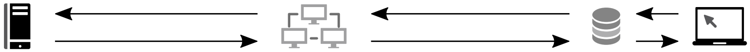
There is no Real-Time

- Given 10 req/sec
- 5 seconds caching
- safes 49 calculations

Caching in HTTP

- Shared
- Private

- Expiration
- Validation



“There are only two hard things in Computer Science: cache invalidation and naming things.” – Phil Karlton

Cachability

- “Sensible defaults”
  - Method (`GET`/`HEAD`)
  - Request headers
  - Response status
- Origin server
  - `Cache-Control`:
    - `public` / `private`
    - `no-cache` / `no-store`
- Client
  - `Cache-Control`:
    - `no-cache` / `no-store`

Expiry

- Origin server
  - `Date:`
  - `Expires:`
  - `Cache-Control:`
    - `max-age=<seconds>`
    - `s-maxage=<seconds>`
- User agent
  - `Cache-Control:`
    - `max-age`
    - `min-fresh`
    - `max-stale`
- Attention: Expiry heuristics

Validation

- Origin server
  - `ETag:`
  - `Last-Modified:`
- Client
  - `If-None-Match:`
  - `If-Modified-Since:`
- Optimistic locking!

Caching . . .

- Even more complicated . . .
  - forced re-validation
  - range caching
  - `Vary`
  - stale handling
  - `PURGE` method
- Be aware of eventual consistency

Evaluation

- Caching is hard
- You must take care
  - Proxies will apply heuristics
  - Clients will misbehave
- Important
  - private / public
  - expiry (for frequent reads)
- ETag
  - Nice
  - Hard to implement correct
  - Optimistic locks

Outline

Basics

HATEOAS

Caching

Authentication

Conclusion

Statelessness

- REST = stateless
- All information must be in request

HTTP Workflow

- $\rightarrow$ `POST /jobs` without auth
- $\leftarrow$ `401 Unauthorized` with `WWW-Authenticate`
- $\rightarrow$ `POST /jobs` with auth for *anonymous*
- $\leftarrow$ `401 Unauthorized` with `WWW-Authenticate`
- $\rightarrow$ `POST /jobs` with auth for *toby*
- $\leftarrow$ `201 Created`

Qafoo
passion for software quality

Basic / Digest Auth

- HTTP default auth methods
- Basic
  - → Some request
  - ← `WWW-Authenticate:  Basic realm="My API"`
  - → `Authorization:  Basic dG9ieTpxYWZvbw==`
- Digest
  - Hashing with server provided nounce
  - Slightly more secure (but not enough!)
- Use HTTPS with Basic and Digest!

API Key

- Authenticate an application
- Not a user
  - Shared secret exchange
  - Cryptographic signing
- Custom `WWW-Authenticate` / `Authorization` format
- Use HTTPS with API-key!

passion for software quality

API Key: Client Request Signing

```php
$clientId = "abc-my-id";
$clientApiKey = "123secret!";

$body = json_encode(array('foo' => 1));
$date = new \DateTime('now', new \DateTimezone('UTC'));

$signature = hash_hmac(
    'sha512',
    $date->format('r') . "\n" . $body,
    $clientApiKey
);

$dateHeader = "Date: " . $date->format('r') . "\n\r";
$authHeader = "Authorization: X-Qafoo " . $clientId . " " .
    $signature . "\n\r";
```

passion for software quality

API Key: Server Request Verification

```php
list($clientId, $clientSignature) = parseAuthHeader(
    $authHeader);
$clientDate = parseDateHeader($dateHeader);

$clientApiKey = loadApiKeyByClientId($clientId);

$expectedSignature = hash_hmac(
    'sha512',
    $clientDate . "\n" . $body,
    $clientApiKey
);

if ($clientSignature !== $expectedSignature) {
    echo "Forbidden";
} else {
    echo "Welcome, Toby!";
}
```

passion for software quality

API Key: JWS/JWT

- Attempt for standardization
- JSON Web Signature `http://qa.fo/jws`
- JSON Web Token `http://qa.fo/jwt`

OAuth2

- Authenticate users 3rd party apps
  - e.g. Twitter / Facebook / . . .
- Allows fine-grained permission system
  - Read personal information
  - Read friend list
  - Post as user
  - . . .
- `http://oauth.net/2/`
- OAuth2 requires HTTPS!

Evaluation

- Basic
  - Don’t use it!
- Digest
  - Works
  - Easy to implement
  - Use HTTPS!
- API Key
  - Good without user permissions
  - Easy to implement
  - Use HTTPS!
- OAuth2
  - Better than OAuth1
  - Still not “easy”
  - Necessary for user permissions
  - Expect debugging sessions

Outline

There is no REST

- There is no pure REST
- You must make trade-offs
- Decide wisely
  - REST architecture attributes
  - Project scope
  - Consumers
  - …
- Questions?

My project by heart

Smart PHP Timeline Profiler

http://tideways.io

30% for 3 month: BERLIN15

https://joind.in/talk/view/13503

Qafoo
passion for software quality

THANK YOU

Rent a quality expert
qafoo.com