Give Me A REST!

...

Amanda Folson
Developer Advocate @ GitLab

@AmbassadorAwsum

Who Am I to Judge?

- Developer Advocate at GitLab
- Average consumer of APIs and IPAs
- Well RESTed (you can boo at my pun)
- Professional conference attendee and tinkerer

@AmbassadorAwsum

APIs are Everywhere

@AmbassadorAwsum

Great, but why do I care?

- Provides a uniform interface for interacting with your application
- API allows you to shard off services
  - Decouples services
  - Website->API
  - Mobile->API
- This is cool if you're into SoA
  - I am.

@AmbassadorAwsum

Types of Application Programming Interfaces

- Language/Platform/Library APIs
  - Win32
  - C++
  - OpenGL
- Web APIs
  - SOAP
  - XML-RPC
  - REST

@AmbassadorAwsum

SOAP

- Stateful
- WS-Security
- Mostly obvious procedures (getRecord, delRecord)
  - Need to know procedure name
  - Need to know order of parameters

@AmbassadorAwsum

REST

- Gaining adoption
- HTTP-based (usually)
- No need to know order of parameters in most cases
- Can return XML, JSON, ?

@AmbassadorAwsum

Sounds Good, Let's Build!

@AmbassadorAwsum

NO

@AmbassadorAwsum

“The best design is the simplest one that works.”

-Albert Einstein

@AmbassadorAwsum

Slow Down

- Don’t rush into v1, v2, etc.
- Make sure you meet goals
- Involve users/engineers from the start
- This is hard

@AmbassadorAwsum

Design

- Immutable blueprint
- Contract between you and users
- SHARE
  - The worst feedback is the feedback you don't hear
- Account for existing architecture
- Changes should provide actual value not perceived/potential value
- Design for uniformity

@AmbassadorAwsum

Know Thy Audience

- Who is this for?
  - Us? Internal?
  - Them? Business partners/3rd parties?
  - ???
- What's the incentive?

@AmbassadorAwsum

Ask!

- Stakeholders will tell you what they need
- Regardless of version, feedback should make it into your spec
- Build something people want

@AmbassadorAwsum

Spec Tools

- API Blueprint
- Swagger
- RAML

The list goes on...

@AmbassadorAwsum

What is REST?

- Representational State Transfer
- HTTP-based routing and methods
  - PUT/GET/POST/etc.
- Stateless
  - No sessions
  - HATEOAS

@AmbassadorAwsum

Resources

- /resource is a gateway to an area of your application
  - /users
  - /places
  - /things
- CRUD actions can be performed on a single resource
  - GET vs getUser, getMessage, getThing
- Use plural nouns, no verbs (GET, CREATE, etc.)
  - Can be for a single item or a collection of items

@AmbassadorAwsum

Action Verbs

@AmbassadorAwsum

GET

- GET data from a /resource
- You do this daily by browsing the web. Go you.
- Uses query string to tell API what data to retrieve
- Returns 200 if everything's OK
    - It's not always okay...more on that later

@AmbassadorAwsum

POST

- Used to create/manipulate data
- Relies on a message body being sent vs query string (XML, JSON, ?)
- Returns 201 Created
- Should return URI to newly created resource

@AmbassadorAwsum

PUT

- Update a resource
- OVERWRITES EXISTING OBJECT WITH NEW ONE
    - You don't have to allow this, be careful with this because its use is inconsistent across APIs, should be used consistently across resources
    - Return 201 (Created) if you do this
    - Can't use PUT within the resource itself (/messages) without an ID
    - PUT should never be use to wrangle partial updates

@AmbassadorAwsum

PATCH

- Updates only the properties provided in the call
- 200 (OK) on successful update
- 304 (Not Modified) if nothing changed

@AmbassadorAwsum

DELETE

- Don’t allow on an entire collection (/users or /messages or /places) or limit it
- 200 (OK) if item or collection was deleted
- 204 (No Content) if item or collection was deleted but there’s no body to return
- 202 (Accepted) if request was accepted and deletion was queued (CDN, cache, etc.)

@AmbassadorAwsum

OPTIONS

- Not for interacting with data
- Informs clients of available methods
- Return 200 or 204 (No Content)
- You could also return 405 (Method Not Allowed) but why would you do that you monster?
- Useful when method should work on items but not collections within a resource (don’t DELETE and collection of users or messages)

@AmbassadorAwsum

Content Types

- Be nice, return more than one!
  - JSON and XML are common
  - Different clients have different needs
  - Easy to add new types as needed if you design for this early on
- If you only allow one type, inform that others aren't allowed
- Use Content-type: to receive and parse client data
- Can use Accept header to see what format client wants back
- For simplicity you can just send back what they send
-

@AmbassadorAwsum

Replying

- Provide information on failures beyond HTTP status codes.
  - “API Key does not have access to modify resource” is better than 403 Forbidden alone
  - 200 OK, 201 Created, 204 No Content, 304 Not Modified, 5xx We Screwed Up, Not You
  - HTTP status codes let client decide what to do next
  - No true standardized error message format, but Google Errors and JSON API are trying
- Not Pokemon

@AmbassadorAwsum

HATEOAS

- Hypermedia As The Engine Of Application State
- Hypertext links to other resources (like links on a page)
  - Every object has actions to perform
- Choose your own adventure for navigation/pagination
  - Give clients list of actions to take
  - Client tracks state
  - Server provides options to change state
- HARD
  - Requires knowing possible ways to interact with object
- Clients have to know how to handle this, some will hardcode anyway

@AmbassadorAwsum

Hypermedia Specs

- Wishful thinking
- There are no standards for this
- JSON API? Custom? HAL?
- HAL/JSON API are good starting points with wide adoption

@AmbassadorAwsum

Versioning

- API is not the time to cowboy deploys/releases
- Design so that you never have to version
- Migrations are hard
- Maintaining n versions
- Deprecate carefully
    - Not everyone can update in a weekend

@AmbassadorAwsum

When to Version

- Backwards incompatible changes
  - Creating new data models
- When your API no longer meets you or your users' needs

BUT NOT

- When you add new resources or data fields

@AmbassadorAwsum

Version in URI

- https://api.myawesomestuff.com/v1/stuff
- Version is obvious to users
- Relative paths can’t always be hypermedia driven

@AmbassadorAwsum

Version in Content-type

- Content-type: application/json+v1
- Cleaner, not as obvious
- Can default to a version if none is specified
- Devs need to know that they need to send this

@AmbassadorAwsum

Version in Custom Header

- Version: 1
- MyApp: 2.6
- No standards
- Requires GREAT docs
- Confusing for users who aren't expecting it

@AmbassadorAwsum

Caching

- Sssccaaallleee
- Cache on API client end
- Cache-control header (public, expires-at, no-cache, max-age), Expires
- Important that this info end up in docs/SDKs

@AmbassadorAwsum

Authentication

- Kill Basic Auth
  - Keep username/passwords safe
- OAuth
  - Require users to explicitly authorize an app
  - Tricky for some people to implement
  - Restrict auth to HTTPS endpoints
  - Restrict domains allowed to auth
  - MITM attacks, make sure users store tokens well

@AmbassadorAwsum

Security

- Treat users as hostile
- Don’t rely on single method
- Apply layers of security
  - Permissions-based API keys/UAC
    - Per app, not per account. Will depend on your architecture
  - DNSBL
  - Content length/depth limits
    - ¿Recursive?
  - SQL injection
  - Rate limit/throttling

@AmbassadorAwsum

Prototyping

- Laravel/Lumen, Flask, Rails, Mojolicious
  - RESTful HTTP routing
  - Zero to API in ~1hr
- Specs
  - Apiary, Mockable, RAML
  - Frameworks allow importing of specs
  - Some spec tools can autogenerate SDKs for you (APIMatic)
- Chrome REST API client, Postman, jsfiddle

@AmbassadorAwsum

Now what?

@AmbassadorAwsum

Maintenance

- Plan to maintain API, SDKs, docs
- Don’t launch and leave
- Allocate maintenance resources
- Community? Paid service?

@AmbassadorAwsum

Things Bad People Do

- Log in to view API docs
  - Counter to open source mentality to use docs as a lead generation tool
- Use HTTP (needs more S)
- No documentation
- Require name/password in URL structure
  - Can be saved in browser/CLI which is very insecure

@AmbassadorAwsum

SDKs

- Nice when these are provided to users
- Should have maintenance plan like API
- Need to be kept in sync
- Should be made by language experts

@AmbassadorAwsum

Documentation

- API is useless if no one knows how to use it
- NEEDS to be part of design process
- All inclusive
  - Errors/methods/parameters
  - Reference and tutorial
  - In sync with changes to API
- Include how to get help
- Open source is nice
-

@AmbassadorAwsum

The best API is the one that exists.

@AmbassadorAwsum

/resources

- Build APIs You Won't Hate - Phil Sturgeon http://apisyouwonthate.com/
- Undisturbed REST - Mike Stowe http://mulesoft.com/restbook
- Apiary – http://apiary.io
- RESTful Web APIs - Leonard Richardson, Mike Amundsen, Sam Ruby

@AmbassadorAwsum

Thank you!

...

Amanda Folson - Developer Advocate at GitLab
amanda@gitlab.com

@AmbassadorAwsum