Building an API

That people will actually use

In this session we will:

- Define "API"
- Discuss API best practices
- Expose a Drupal installation via an API
- Support API developers by providing:
  - Documentation
  - SDKs
  - Example implementations
- Learn to plan for the future

What is an API?

- Application Programming Interface
- Expose your data to the world
- Expose your application via two-way communication
- Enable developers to extend your application

Why expose an API?

- Extend the reach of your and data
- Expand implementations of your application
- Crowdsource feature development
- Reach a wider audience

What is a *good* API?

- Logically represent your application and data
- Support multiple request formats
- Provide multiple response formats
- Support developers
- Plan for the future

Path patterns

- Define objects granularly
- Use plural nouns
- Use query parameters for filtering
- Avoid using verbs; use HTTP methods
- Bonus: shallow/depth

Dos and Don'ts

Don't

- */api/v1/nodes/[node_id]*
  - nodes don't define objects
- */api/v1/video/[video_id]*
  - non-plural nouns
- */api/v1/videos/funny*
  - filters in path
- */api/v1/videos/create*
  - verbs in path
- */api/v1/videos/[video_id]/comments*
  - ideally, avoid traversing more than two levels

Do

- */api/v1/articles/[article_id]*
  - content types define objects
- */api/v1/videos/[video_id]*
  - plural denotes a container
- */api/v1/videos?category=funny*
  - query strings are better
- *POST /api/v1/videos*
  - use HTTP methods
- */api/v1/comments?content=video&id=[video_id]*
  - query strings offer many benefits

Services

drupal.org/project/services

Services

The Services module exposes elements of your Drupal installation via an API.

- Built-in request formats:
  - bencode, json, jsonp, php, rss, xml
- Built-in response formats:
  - json, xml, form-data, etc.

Data exposure methods

- Baked-in methods
  - CRUD operations, Relationships, and Actions

Screenshot of baked-in Services operations

Data exposure methods

- Baked-in methods
  - CRUD operations, Relationships, and Actions

- Content API
  - drupal.org/project/contentapi

Screenshot of Content API options

Data exposure methods

- Baked-in methods
  - CRUD operations, Relationships, and Actions
- Content API
  - drupal.org/project/contentapi
- Custom methods
  - hook_services_resource()

Example hook_services_resource() implementation

Authentication

- Drupal session
- OAuth
- Custom authentication methods

Introducing:
Services Documentation

drupal.org/project/services_documenation

Support developers

1. Documentation
2. Examples
3. SDKs

Documentation

- Automatically generate documentation for resources, operations, and arguments
- Provide request and response examples
- Version Control!
- **Fully themable!!!**

Example of a documentation implementation

SDKs

Offer SDKs that developers can implement to use your API:

- Facilitate rapid application development
- Provide a language-agnostic service
- Empower a community to grow organically

Example of an SDK implementation

Examples

Provide examples of applications that harness your API with or without an SDK.

- Widgets
- Interfaces
- jQuery plugins
- Mobile apps

Example of an example implementation

Planning ahead

Future-proofing your API

Versioning

- Only release "major versions"
- Avoid deprecation which breaks third-party applications

Mitigating growth

- API platform providers
  (Mashape, Mashery, 3scale, Apigee, Mulesoft)
  - Authentication
  - Response caching
  - Analytics
  - Integrate multiple APIs behind a facade

Thanks for listening!

Now go build something awesome

Resources