REST API Design Guidance

Goals

- Elevate Microsoft's published REST API design guidelines.
- Highlight common design decisions and factors to consider when designing.
- Provide additional resources to inform API design in areas not directly addressed by the Microsoft guidelines.

Common API Design Decisions

The Microsoft REST API guidelines provide design guidance covering a multitude of use-cases. The following sections are a good place to start as they are likely required considerations by any REST API design:

- URL Structure
- HTTP Methods
- HTTP Status Codes
- Collections
- JSON Standardizations
- Versioning
- Naming

Creating API Contracts

As different development teams expose APIs to access various REST based services, it's important to have an API contract to share between the producer and consumers of APIs. Open API format is one of the most popular API description format. This Open API document can be produced in two ways:

- Design-First - Team starts developing APIs by first describing API designs as an Open API document and later generates server side boilerplate code with the help of this document.
- Code-First - Team starts writing the server side API interface code e.g. controllers, DTOs etc. and later generates and Open API document from it.

Design-First Approach

A Design-First approach means that APIs are treated as "first-class citizens" and everything about a project revolves around the idea that at the end these APIs will be consumed by clients. So based on the business requirements API development team first start describing API designs as an Open API document and collaborate with the stakeholders to gather feedback.

This approach is quite useful if a project is about developing externally exposed set of APIs which will be consumed by partners. In this approach, we first agree upon an API contract (Open API document) creating clear expectations on both API producer and consumer sides so both teams can begin work in parallel as per the pre-agreed API design.

Key Benefits of this approach:

- Early API design feedback.

- Clearly established expectations for both consumer & producer as both have agreed upon an API contract.
- Development teams can work in parallel.
- Testing team can use API contracts to write early tests even before business logic is in place. By looking at different models, paths, attributes and other aspects of the API testing can provide their input which can be very valuable.
- During an agile development cycle API definitions are not impacted by incremental dev changes.
- API design is not influenced by actual implementation limitations & code structure.
- Server side boilerplate code e.g. controllers, DTOs etc. can be auto generated from API contracts.
- May improve collaboration between API producer & consumer teams.

Planning a Design-First Development:

1. Identify use cases & key services which API should offer.
2. Identify key stakeholders of API and try to include them during API design phase to get continuous feedback.
3. Write API contract definitions.
4. Maintain consistent style for API status codes, versioning, error responses etc.
5. Encourage peer reviews via pull requests.
6. Generate server side boilerplate code & client SDKs from API contract definitions.

Important Points to consider:

- If API requirements changes often during initial development phase, than a Design-First approach may not be a good fit as this will introduce additional overhead, requiring repeated updates & maintenance to the API contract definitions.
- It might be worthwhile to first try out your platform specific code generator and evaluate how much more additional work will be required in order to meet your project requirements and coding guidelines because it is possible that a particular platform specific code generator might not be able to generate a flexible & maintainable implementation of actual code. For instance If your web framework requires annotations to be present on your controller classes (e.g. for API versioning or authentication), make sure that the code generation tool you use fully supports them.
- Microsoft TypeSpec is a valuable tool for developers who are working on complex APIs. By providing reusable patterns it can streamline API development and promote best practices. We have put together some samples about how to enforce an API design-first approach in a GitHub CI/CD pipeline to help accelerate it's adoption in a Design-First Development.

Code-First Approach

A Code-First approach means that development teams first implements server side API interface code e.g. controllers, DTOs etc. and than generates API contract definitions out of it. In current times this approach is more widely popular within developer community than Design-First Approach.

This approach has the advantages of allowing the team to quickly implement APIs and also providing the flexibility to react very quickly to any unexpected API requirement changes.

Key Benefits of this approach:

- Rapid development of APIs as development team can start implementing APIs much faster directly after understanding key requirements & use cases.
- Development team has better control & flexibility to implement server side API interfaces in a way which best suited for project structure.
- More popular among development teams so its easier to get consensus on a related topic and also has more ready to use code examples available on various blogs or developer forums regarding how to generate Open API definitions out of actual code.
- During initial phase of development where both API producer & consumers requirements might change often this approach is better as it provides flexibility to quickly react on such changes.

Important Points to consider:

- A generated Open API definition can become outdated, so its important to have automated checks to avoid this otherwise generated client SDKs will be out of sync and may cause issues for API consumers.
- With Agile development, it is hard to ensure that definitions embedded in runtime code remain stable, especially across rounds of refactoring and when serving multiple concurrent API versions.
- It might be useful to regularly generate Open API definition and store it in version control system otherwise generating the OpenAPI definition at runtime might makes it more complex in scenarios where that definition is required at development/CI time.

How to Interpret and Apply the Guidelines

The API guidelines document includes a section on how to apply the guidelines depending on whether the API is new or existing. In particular, when working in an existing API ecosystem, be sure to align with stakeholders on a definition of what constitutes a breaking change to understand the impact of implementing certain best practices.

> We do not recommend making a breaking change to a service that predates these guidelines simply for the sake of compliance.

Resources

- Microsoft's Recommended Reading List for REST APIs
- Documentation - Guidance - REST APIs
- Detailed HTTP status code definitions
- Semantic Versioning
- Other Public API Guidelines
- Microsoft TypeSpec
- Microsoft TypeSpec GitHub Workflow samples

Last update: August 22, 2024