API CodeFlow

An agile, design-first workflow that integrates API descriptions into the build process, so API docs and code are continually in sync.

Why Design-First?

API design-first is a lifecycle model that starts with a conscious, collaborative API design process. API description languages like OpenAPI allow fast, evolutionary prototyping. API mock services can easily simulate a working implementation, and documentation formats with integrated sandbox testing make it easy for API client developers to test your API, even before you start coding.

The initial design focus allows your team to design "outside-in," so the design reflects the API client developer's view of the business domain, and aligns with the client developer's real-world usage patterns.

When you're ready to start implementation, code generation creates a scaffold, or stub code. Developers don't have to write repetitive boilerplate code, and they get a head start with generated code that exemplifies recommended patterns and practices for API implementation.

Challenges with API Design-First

Most API design-first projects generate code as a one-time event, following an initial design phase. Subsequent changes to the API have to be made separately in the code and the API specification, sometimes even by different team members.

As the API evolves, the specification and code tend to drift out of sync, and the team may not realize it until users report an inconsistency. Once the code and specs diverge, we've lost the benefits of consumer-driven design, rapid prototyping, and auto-generated scaffolding.

Design-First, All the Way Down

Here's a thought experiment: what if API descriptions are just another kind of code? If we integrate code generation into the automated build, the API spec becomes a first-class part of the service implementation codebase.

API CodeFlow is API design-first, taken all the way through to implementation. API changes start with the specification, and flow automatically to the generated stub code. Done correctly, regeneration is a safe operation, and clearly indicates any breaking changes so we know exactly how to update our code to the modified API design.

Instead of a one-time event, code generation happens continuously. Design-first goes from waterfall to an agile, iterative process, as API documentation and code evolve in lockstep.

CodeFlow Gets Real with RepreZen

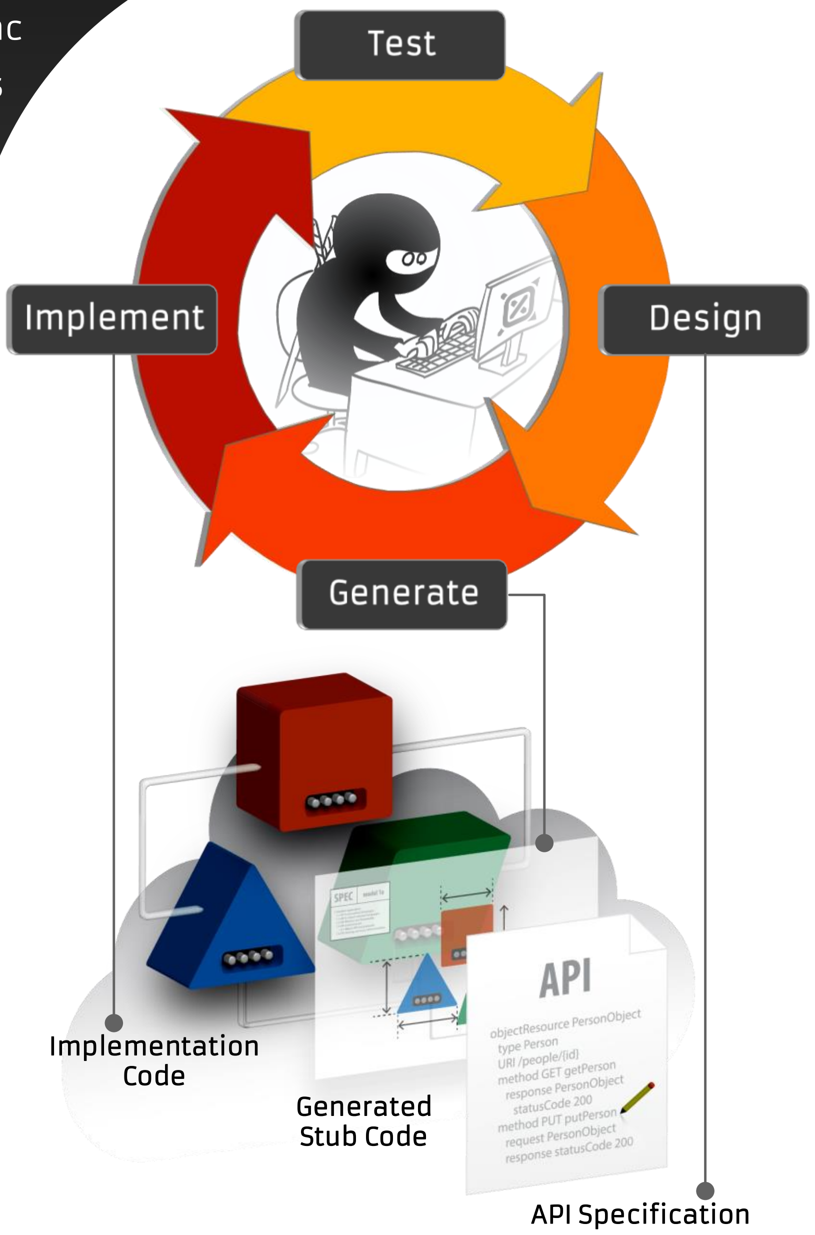
RepreZen API Studio has everything you need to start building APIs today:

- **Fully integrated API design, documentation and development** in a single environment, optimized for breakthrough productivity. Choose the standalone API Studio Desktop, or install API Studio for Eclipse into your own IDE.
- **First-class OpenAPI and RAPID-ML editing** with template-driven code assist; live documentation, diagram and Swagger UI views; powerful multi-file support; and integrated version control.
- **Extensible Code Generation** with over 200 generators ready-to-run, and the powerful GenFlow open source framework to build your own custom code generators. All generators run from the IDE, the command line, and your automated CI/CD builds with Maven and Gradle.
- **Project Templates and Learning Resources** show you the ins and outs of API CodeFlow with popular programming frameworks in Java, NodeJS, and others. Join the community and show us how *you* CodeFlow!

API CodeFlow

RepreZen API Studio

- Agile, API design-first workflow
- API specification & code evolve together, always in sync
- Rapid prototyping encourages customer feedback
- Efficiency of design-first + flexibility & control of code-first



1

- **Design Your API** in OpenAPI or RAPID-ML.
- **Use Live Docs and Diagrams** for real-time visual feedback.
- **Test-Drive Your API** with the mock service and Swagger-UI view.
- **Evolve the API Specification** incrementally, adding documentation, responding to feedback and new req's.

2

- **Automate Code Generation** as part of the build process.
- **Version the API Spec** as part of your codebase.
- **Embed the API Specification** as a static resource, used at runtime for *straight-through* API documentation.

3

- **Implement the API,** extending generated code as provided by the language, framework, and generator.
- **Maintain strict separation** of implementation code vs. generated stub code, so it's always safe to regenerate.
- **Detect & fix breaking changes** to the API. Dependencies between generated code and your implementation should expose inconsistencies as compiler errors or test failures.

http://RZen.io/APICodeFlow