GCG

-

A Generic Code Generator

Vincent Hénaux

April 17, 2024

Contents

1 Introduction

GCG, short for Generic Code Generator, is a versatile tool designed to automate the creation of repetitive code segments in software development projects. It dynamically populates template files with data sourced from configuration files, operating independently of language, framework, or technology constraints. The output generated by GCG is compatible with all compilers and interpreters, ensuring seamless integration into existing development environments.

In software engineering, the creation of repetitive code segments is an unavoidable aspect of project development, dictated by language specifications, framework requirements, and design patterns. GCG's core objective is to alleviate software engineers from the burden of writing this type of code, liberating valuable time for more creative and high-value tasks.

By leveraging the power of GCG, software engineers can significantly enhance productivity by minimizing the time spent on mundane and error-prone coding tasks. These repetitive code segments, often serve as the foundation of software projects, rendering their accuracy and consistency critical to project success. GCG empowers developers to fortify their codebases against errors and inconsistencies, thereby enhancing overall maintainability and reliability.

2 Mission statement

The mission of GCG, is to empower software engineers with a versatile and efficient tool that optimizes their coding workflows. GCG is committed to enhancing developers' productivity and code quality by streamlining repetitive tasks, promoting consistency, and facilitating collaboration within development teams.

GCG is designed to simplify complex coding tasks, minimize errors, and enable developers to focus on more innovative and value-added aspects of software development. Our goal is to foster a culture of efficiency and excellence, accelerating project delivery and driving positive outcomes in software development endeavors.

3 Solution overview

GCG offers a versatile solution for code generation, available in multiple forms tailored to different development environments. Whether you're working on Windows or Linux, GCG provides the tools you need to automate code generation and streamline your development workflows.

At the heart of GCG is its ability to interpret template files containing special commands and placeholders, and populate them with data sourced from the configuration file. These special commands act as directives for GCG, guiding the generation process and ensuring that the output meets the desired specifications.

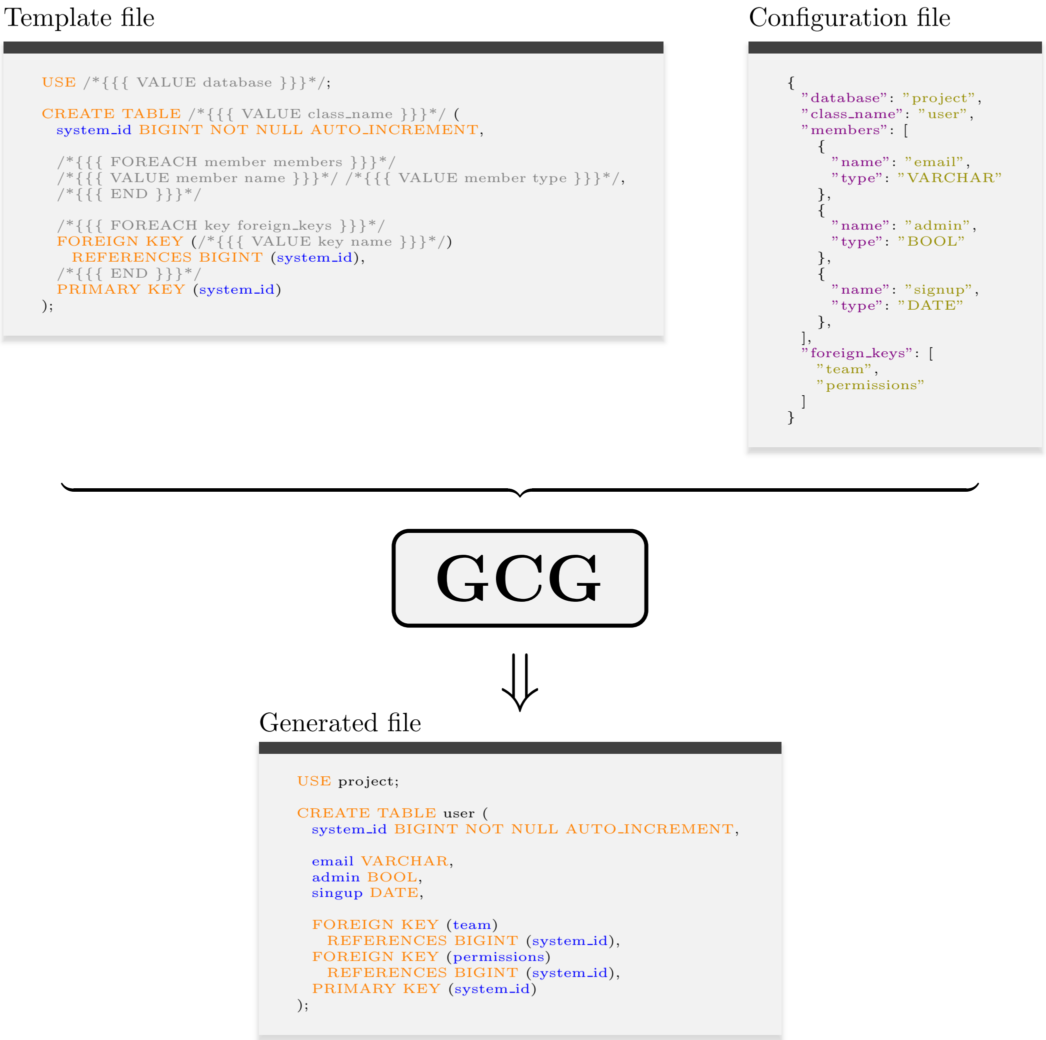
Using GCG is a simple and intuitive process. Users begin by creating a template file that defines the structure and layout of the code to be generated. This template file may include placeholders for dynamic data, denoted by special commands recognized by GCG.

Next, users provide a configuration file containing the necessary data to populate the placeholders within the template file. This configuration file serves as a repository for information such as variable values, function parameters, or any other data needed to customize the generated code.

Once the template and configuration files are prepared, users can utilize GCG in various forms:

- **Executable**: The GCG executable provides a standalone solution for code generation. Users can run the executable directly from the command line, providing the necessary template and configuration files as input to initiate the code generation process.
- **C Dynamic Library**: GCG is available as a dynamic library for users who wish to integrate its code generation capabilities into their applications via function calls. This flexible option allows developers to incorporate GCG's functionality directly into their projects, regardless of the programming language being used.
- **Python Wrapper for the C Library**: GCG also offers a Python wrapper for its C dynamic library, providing an easy way to call GCG in script languages. This wrapper simplifies integration with Python scripts, enabling users to leverage GCG's code generation features within their projects without language constraints.

Each form of GCG is designed to operate seamlessly in its respective environment, ensuring compatibility and ease of use across different development platforms. Whether you prefer working with executables, C libraries, or Python wrappers, GCG provides the flexibility and functionality you need to automate code generation and enhance your development process.



Template file

```
USE /*{{{ VALUE database }}}*/;

CREATE TABLE /*{{{ VALUE class_name }}}*/ (
  system_id BIGINT NOT NULL AUTO_INCREMENT,

  /*{{{ FOREACH member members }}}*/
  /*{{{ VALUE member name }}}*/ /*{{{ VALUE member type }}}*/,
  /*{{{ END }}}*/

  /*{{{ FOREACH key foreign_keys }}}*/
  FOREIGN KEY (/*{{{ VALUE key name }}}*/)
    REFERENCES BIGINT (system_id),
  /*{{{ END }}}*/
  PRIMARY KEY (system_id)
);
```

Configuration file

```
{
  "database": "project",
  "class_name": "user",
  "members": [
    {
      "name": "email",
      "type": "VARCHAR"
    },
    {
      "name": "admin",
      "type": "BOOL"
    },
    {
      "name": "signup",
      "type": "DATE"
    },
  ],
  "foreign_keys": [
    "team",
    "permissions"
  ]
}
```

Generated file

```
USE project;

CREATE TABLE user (
  system_id BIGINT NOT NULL AUTO_INCREMENT,

  email VARCHAR,
  admin BOOL,
  singup DATE,

  FOREIGN KEY (team)
    REFERENCES BIGINT (system_id),
  FOREIGN KEY (permissions)
    REFERENCES BIGINT (system_id),
  PRIMARY KEY (system_id)
);
```

4 Use cases

GCG offers a wide range of use cases, empowering developers to automate various aspects of software development and streamline their workflows. Some key use cases of GCG include:

- **Generating CRUD Functions for Database Interaction**: GCG simplifies the creation of CRUD (Create, Read, Update, Delete) functions for database interaction. Developers can define templates for common database operations and use GCG to generate the corresponding code, reducing manual effort and ensuring consistency across projects.

- **Creating REST API Endpoints**: With GCG, developers can automate the generation of REST API endpoints. By defining templates specific to APIs, GCG can automatically generate the necessary code to expose data and functionality over HTTP, accelerating the development of web applications and services.
- **Generating Data Validation Logic**: GCG facilitates the generation of data validation logic for input processing. Developers can define templates for data validation rules and use GCG to generate validation code, ensuring data integrity and security in their applications.
- **Automating Repetitive Tasks in Specific Frameworks or Libraries**: GCG can automate repetitive tasks associated with specific frameworks or libraries. Whether it's generating boilerplate code, configuring settings, or implementing common patterns, developers can leverage GCG to automate routine tasks and focus on more challenging aspects of development.
- **Generating Interfaces Between Projects and Frameworks**: GCG can generate interfaces between projects and frameworks, facilitating seamless integration and simplifying framework version upgrades. By defining templates for interface components, developers can use GCG to automatically adapt their codebase to changes in framework requirements, reducing the effort and risk associated with framework upgrades.
- **Automating Code Documentation**: GCG can automate the generation of code documentation, reducing the manual effort required to document code changes and ensuring consistent documentation across projects.
- **Custom Code Generation**: Beyond the predefined use cases, developers can create custom code generation workflows tailored to their specific requirements. GCG's flexibility and extensibility allow developers to adapt the tool to a wide range of use cases and project.

By offering a versatile and customizable code generation solution, GCG empowers developers to increase productivity, maintain code quality, and accelerate project delivery across various domains and industries.

5 Key features

GCG provides a comprehensive set of features designed to streamline code generation and enhance development workflows. Some of the key features of GCG include:

- **Template-Based Code Generation**: GCG allows developers to define templates for code generation, enabling the creation of reusable structures for common coding tasks.

- **Configuration-Based Customization**: Developers can customize code generation by providing configuration files containing the necessary data to populate template placeholders. This configuration-based approach ensures flexibility and adaptability to project requirements.
- **Special Commands and Placeholders**: GCG supports special commands and placeholders within templates, allowing developers to insert dynamic data and logic into generated code.
- **Cross-Platform Compatibility**: GCG is available for both Windows and Linux environments, ensuring compatibility across different development platforms.
- **Multiple Forms of Deployment**: GCG is available in various forms, including an executable, a C dynamic library, and a Python wrapper for the C library, providing flexibility in deployment options to suit different project needs.
- **Error Handling and Logging**: GCG provides robust error handling and logging mechanisms, allowing developers to diagnose and troubleshoot issues effectively during the code generation process.

By offering a rich set of features and capabilities, GCG empowers developers to automate code generation, increase productivity, and maintain code quality across various development projects and environments.

6 Intellectual property

GCG is committed to respecting and protecting the intellectual property rights of its users. Our approach to intellectual property ensures that the rights of our clients are upheld and that they maintain full ownership of their code and related assets.

The core component of GCG, the code generator itself, is protected intellectual property owned by GCG. This includes the algorithms, logic, and functionality that enable code generation based on provided templates and configurations. However, it's important to note that GCG does not claim ownership of any templates, configuration files, or generated code produced by the tool.

Templates and configuration files provided by clients remain their exclusive property. These files contain the unique specifications, structures, and data necessary for code generation. GCG respects the confidentiality and ownership of these assets and does not retain any rights over them.

Similarly, the code generated by GCG belongs entirely to you. Whether it's CRUD functions, REST API endpoints, data validation logic, or any other code generated using GCG, clients retain full ownership and control over the resulting codebase. GCG does not assert any rights over the generated code and does not impose any restrictions on its use, modification, or distribution.

7 Benefits

GCG is a powerful tool designed to streamline software development processes and enhance the productivity of engineering teams. Here are some key ways in which GCG can benefit your organization:

- **Time and Effort Savings**: GCG significantly reduces the time and effort spent on writing boilerplate or repetitive code, which is prone to errors and can be tedious for developers. By automating these tasks, GCG allows engineers to focus on more innovative and value-added aspects of their projects, increasing overall productivity.
- **Consistency and Maintainability**: With GCG, the use of templates ensures consistency across the codebase. As templates are reused, the codebase becomes more standardized and easier to maintain, leading to improved code quality and reduced technical debt over time.
- **Bug Fix Propagation**: By addressing bugs or making updates in a template, GCG ensures that every project utilizing that template is automatically fixed or updated. This centralized approach to bug fixing increases efficiency and reduces the risk of inconsistencies across projects.
- **Competitive Advantage Through Intellectual Property**: By amassing intellectual property through templates created with GCG, companies can gain a competitive advantage in the market. Templates represent valuable assets that enable organizations to deliver software solutions faster, cheaper, and of higher quality to their clients. This strategic use of intellectual property sets companies apart from competitors and positions them as leaders in their respective industries.

In addition to these benefits, GCG offers distinct advantages over traditional approaches such as abstraction and meta-programming:

- **Simplified Complexity and Enhanced Maintainability**: GCG templates maintain a similar level of complexity to plain code, making them easier to understand and maintain compared to abstracted or meta-programmed solutions. This simplicity promotes wider knowledge dissemination within development teams and ensures better long-term maintainability of the codebase.
- **Streamlined Debugging Process**: With GCG, developers can debug generated code more efficiently, as they can read and analyze it as if it had been written by someone else. This eliminates the need to navigate through layers of abstraction or interpret complex meta-programmed constructs, leading to faster issue resolution and improved code quality.

By harnessing the power of GCG and strategically leveraging intellectual property through templates, companies can transform their software development processes, gain a competitive edge, and deliver superior solutions to their clients.

8 Pricing and licensing

	Number of users		
	1-9	10-199	200+
GCG full access	✓	✓	✓
Maintenance	✓	✓	✓
Free basic customizations	✗	✓	✓
License	See genericcodegenerator.com		

9 Prerequisites

GCG is designed for immediate use, with no prerequisites or extensive setup required.

10 Contact information

genericcodegenerator.com

genericcodegenerator@gmail.com