The FrameSquared

Carbone, John
john@qlcorp.net

April 6, 2019

Abstract

The FrameSquared performs database schema version control and scaffold generation for Postgres databases and the Treefrog C++ MVC framework using YAML to

The FrameSquared is very much a work in progress at this time and is not ready for prime time. Use at your own risk.

xxx

FS2	Postgres	C++/Qt	SOAP	ExtJS/JavaScript
string	varchar	QString	xsd:string	Ext.data.field.String
integer	integer	qint32	xsd:int	Ext.data.field.Integer
float	real	float	xsd:float	Ext.data.field.Number
double	double precision	double	xsd:double	Ext.data.field.Number
numeric	numeric	double	xsd:double	Ext.data.field.Number
boolean	boolean	bool	xsd:boolean	Ext.data.field.Boolean
uuid	uuid	QString	xsd:string	Ext.data.field.String
serial	serial	qint	xsd:int	Ext.data.field.Integer
datetime	timestamp	QDateTime	xsd:string	Ext.data.field.Date
	date			Ext.data.field.Date
	time			Ext.data.field.Date

1 Overview

The rise of iterative and incremental software development methodologies has been transformative. Continuous integration[1], deployment and delivery[2, 3] has met the challenge of regular changes in requirements. Unpredictable and changing requirements[4] have been factored into the development process to enable software architectures to evolve gracefully.

Iterative and incremental methodologies have been successfully applied to database schema revision control and migration[5]. Refactoring database schema effects the database and related queries but also has repercussions to external systems that access the database programmatically, these include models and controllers.

The nature of MVC frameworks lends itself to the use of boilerplate code. Some MVC frameworks create scaffolding using templates to generate boilerplate code. Scaffolding was made populate by Ruby-on-Rails. Most scaffold generators focus primarily on create, retrieve, update and delete (CRUD) operations using HTML[6, 7] and do not generally support single page applications[8] and associated messaging protocols such as SOAP or REST.

The FrameSquared is tool that can generate code to be used as final source files and/or migrations based on serial numbers and up/down direction and can execute programs. The FrameSquared uses the notion of "serial" numbers and "direction" to determine whether a migration "action" should be executed.

2 Getting Started

2.1 Required Libraries

The FrameSquared utilizes the Inja[9] template engine. Inja and The FrameSquared both rely on the nlohmann JSON library[10]. The FrameSquared also uses the jbeder YAML[11] library.

2.1.1 yaml-cpp

Download and compile yaml-cpp:

```
git clone https://github.com/jbeder/yaml-cpp.git
cd yaml-cpp/
mkdir build
cd build
cmake ..
make
sudo make install
```

Figure 1: Steps to building

Optionally, if you want to clean after the build:

```
        cd ..
        rm -rf build
```

Figure 2: Steps to clean

2.1.2 nlohmann/json

Download and compile nlohmann/json:

```
        git clone https://github.com/nlohmann/json.git
        cd json/
        mkdir build
        cd build
        cmake ..
        make
        sudo make install
```

Figure 3: Steps to building

Optionally, if you want to clean after the build:

```
        cd ..
        rm -rf build
```

Figure 4: Steps to clean

2.2 Building The FrameSquared

The following steps will compile the The FrameSquared tool.

```
          mkdir build
          cd build
          cmake ..
          make
          ./fs2_migration02
```

Figure 5: Steps to building

3 Configuration

3.1 config.yaml

All settings for The FrameSquared are configured using YAML and is located in the config.yaml file. This file can be found in the ”config” directory by default.

The location of the file may be overridden by using -c or –config on the command line.

3.2 paths

The ”paths” section provides the locations for files and directories that the system requires.

3.2.1 map file

The ”map” key has a string value that is the path to map file. This file is used to translate various values used be the system. The map file contains multiple maps. Each top level key in the file is a distinct map. Each map has a set of key/value pairs. Maps are accessed via the ”map”**??** and ”mapCompare”**??** functions.

```
    [Map Name]:
        [Key]: [Value]
```

Figure 6: Map file syntax

3.2.2 migrations dir

The value of ”migrations dir” contains the directory where the migration files are located. The system will recursively iterate over this directory to discover all migration files during the scan operation**??**. See **??** for more information regarding the syntax of the migration files.

3.2.3 migrations file

The value of the ”migration file” is the filename where the system will store the migration json file. This file can be thought of as a database of all the migrations.

During scan operations the system stores all discovered migration data in this file.

The system uses this file the migrations file to perform all operations to be applied to the system.

3.2.4 meta dir

The value of ”meta dir” contains the directory where the metadata for each of the entities are stored. The metadata represents the current state of the entity and is updated as each migration is applied to the entity.

3.3 sequences

3.4 actions

3.5 map.yaml

4 Overview???

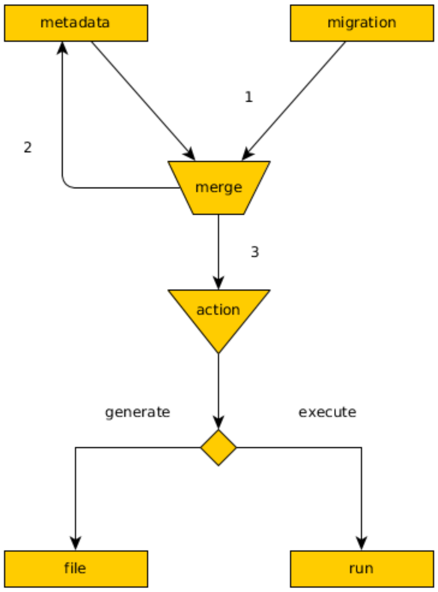

Figure 7: Merge Overview

References

[1] Paul M. Duvall, Steve Matyas, Paul Duvall, Andrew Glover *Continuous Integration: Improving Software Quality and Reducing Risk* Addison-Wesley, 2007

[2] Jez Humble and David Farley *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation* Addison-Wesley, August 2010

[3] Sten Pitet *Continuous integration vs. continuous delivery vs. continuous deployment* `https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment` Atlassian, Retrieved January 1st, 2019

[4] Neal Ford, Rebecca Parsons, Patrick Kua *Building Evolutionary Architectures: Support Constant Change* O'Reilly Media Inc, October 8, 2017

[5] Scott J Ambler and Pramod J. Sadalage *Refactoring Databases: Evolutionary Database Design* Addison-Wesley Professional, March 13, 2006

[6] *Scaffoling (programming)* `https://en.wikipedia.org/wiki/Scaffold_(programming)` Wikipedia, Retrieved January 1st, 2019

[7] Nick Harrison *Using Scaffolding to Create MVC Applications with Visual Studio* `https://www.red-gate.com/simple-talk/dotnet/asp-net/using-scaffolding-to-create-mvc-applications-with-visual-studio/` www.red-gate.com, 14 December 2015

[8] Paul Sherman *How Single-Page Applications Work:Getting to know the browser behaviors and APIs responsible for powering more and more web pages* `https://medium.com/@pshrmn/demystifying-single-page-applications-3068d0555d46` mediun.com, 11 April 2018

[9] Inja *A Template Engine for Modern C++* `https://github.com/pantor/inja` GitHub

[10] JSON for Modern C++ *A Template Engine for Modern C++* `https://github.com/nlohmann/json/` GitHub

[11] YAML *A YAML parser and emitter in C++* `https://github.com/jbeder/yaml-cpp` GitHub