HerikLyma / **CPPWebFramework** Public

Code | Issues 5 | Pull requests 1 | Actions | Projects | Security and quality | Insights

master | 1 Branch | 0 Tags | Go to file | Go to file | Code

HerikLyma Update README.md — 4311f73 · 2 years ago

CPPWebFramework	Adjust configuration initialization	6 years ago
documentation	SSL improvements	8 years ago
examples	Experimental ORM examples and readme …	7 years ago
.appveyor.yml	Experimental ORM examples and readme …	7 years ago
.travis.yml	Travis adjust	7 years ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	8 years ago
CONTRIBUTING.md	Create CONTRIBUTING.md	8 years ago
LICENSE.txt	New write method for QJsonObject on Http…	8 years ago
PULL_REQUEST_TEMPLATE.md	Update PULL_REQUEST_TEMPLATE.md	8 years ago
README.md	Update README.md	2 years ago
environment.Dockerfile	adding docker support as well a hello world…	8 years ago

README | Code of conduct | Contributing | MIT license

C++ Web Framework (CWF)

The C++ Web Framework (CWF) is a MVC web framework, Open Source, under MIT License, using C++ with Qt to be used in the development of web applications. The CWF was designed to consume few computational resources, such as memory and processing, and have a low response time for requests. With MVC (Model-View-Controller) architecture, where you can create classes to take care of the business layer (Model), use CSTL (C++ Server Pages Standard Tag Library) within the Web Pages to take care of data presentation (View) and use the controllers as a between the two layers (Controller). The CWF had inspirations in Java Servlets, JSTL and Spring Framework.

Because it is created in Qt, the C++ Web Framework can run on the same platforms supported by Qt:

- **Desktop:** Linux, OS X, Windows
- **Embedded and RTOS:** Linux, QNX, VxWorks, Windows
- **Mobile:** Android, iOS, Windows

The CWF has only one configuration file, called CPPWeb.ini and a policy of using only C++ and Qt in the development of its components in order to avoid the installation of numerous libraries and conflicts, maintain multiplatform characteristics, facilitate installation and keep the learning curve low in order to make web development as simple as possible, even for beginners.

Hello World - Example

```
#include "cppwebapplication.h"

class HelloWorldController : public CWF::Controller
{
public:
    void doGet(CWF::Request &request, CWF::Response &response) const override
    {
        response.write("<html><body>Hello World!</body></html>");
    }
};

//Call
//http://localhost:8080/hello
int main(int argc, char *argv[])
{
    CWF::CppWebApplication server(argc, argv, "/PATH_TO_EXAMPLE/server/");
    server.addController<HelloWorldController>("/hello");
    return server.start();
}
```

MVC (Model-View-Controller) - Example

```
//hellomodel.h (Model)
#include <QObject>

class HelloModel : public QObject
{
    Q_OBJECT
public slots:
    QString greeting() const
    {
        return "Hello User!";
    }
};

//helloview.view (View)
<html>
    <head>
        <title>Hello</title>
    </head>
    <body>
        <out value="#{model.greeting}"/>
    </body>
</html>

//hellocontroller.h (Controller)
#include <cwf/controller.h>
#include <model/hellomodel.h>

class HelloController : public CWF::Controller
{
public:
    void doGet(CWF::Request &request, CWF::Response &response) const override
    {
        HelloModel model;
        request.addAttribute("model", &model);
        request.getRequestDispatcher("/pages/helloview.view").forward(request, response);
    }
};

//main.cpp
```

```
#include <cwf/cppwebapplication.h>
#include <controller/hellocontroller.h>

//Call
//http://localhost:8080/hello
int main(int argc, char *argv[])
{
    CWF::CppWebApplication server(argc, argv, "/PATH_TO_EXAMPLE/server/");
    server.addController<HelloController>("/hello");
    return server.start();
}
```

REST Web Service - Example

```
#include <cwf/sqlquery.h>
#include <cwf/cppwebapplication.h>
#include <cwf/sqldatabasestorage.h>

/*
 * SQL Script
 * create table countries (co_id serial primary key, co_name varchar unique);
 * insert into countries (co_name) values ('BRAZIL'), ('UNITED STATES OF AMERICA'), ('CANADA');
 */

CWF::SqlDatabaseStorage storage("QPSQL", "localhost", "postgres", "postgres", "1234", 5432);

class CountriesController : public CWF::Controller
{
public:
    void doGet(CWF::Request &request, CWF::Response &response) const override
    {
        CWF::SqlQuery qry(storage);
        qry.exec("select * from countries");
        response.write(qry.toJson());
    }
};

//Call
//http://localhost:8080/countries
int main(int argc, char *argv[])
{
    CWF::CppWebApplication server(argc, argv, "/PATH_TO_EXAMPLE/server/");
    server.addController<CountriesController>("/countries");
    return server.start();
}
```

ORM (Object Relational Mapper) - Experimental - Example

```
#include <usermodel.h>
#include <cwf/cppwebapplication.h>
#include <cwf/sqldatabasestorage.h>
/*
* ORM (Experimental) - Tested only on PostgreSQL
*/
CWF::SqlDatabaseStorage conexao("QPSQL", "localhost", "postgres", "postgres", "1234", 5432);

class ORMController : public CWF::Controller
{
public:
    void doGet(CWF::Request &request, CWF::Response &response) const override
    {
```

```
        UserModel user{conexao};
        user.setName("Herik Lima");
        user.setPhone("+55 11 9 99999-0000");
        user.setCountry("Brazil");
        user.setState("São Paulo");
        response.write(QByteArray("<html><body>") + (user.save() ? "Saved" : "Error") + "</body></html>");
    }
};
//Call
//http://localhost:8080/orm
int main(int argc, char *argv[])
{
    CWF::CppWebApplication server(argc, argv, "/home/herik/CPPWebFramework/examples/ORM/server");
    UserModel{conexao}.updateDB();//Create or update the table in database
    server.addController<ORMController>("/orm");
    return server.start();
}

//usermodel.h
#ifndef USERMODEL_H
#define USERMODEL_H

#include <cwf/model.h>

class UserModel : public CWF::Model
{
    Q_OBJECT
    Q_PROPERTY(QString name READ getName WRITE setName)
    Q_PROPERTY(QString phone READ getPhone WRITE setPhone)
    Q_PROPERTY(QString country READ getCountry WRITE setCountry)
    Q_PROPERTY(QString state READ getState WRITE setState)
    QString name;
    QString phone;
    QString country;
    QString state;
public:
    explicit UserModel(CWF::SqlDatabaseStorage &connection) : CWF::Model(connection, "usuario") {}
public slots:
    QString getName() const { return name; }
    void setName(const QString &value) { name = value; }
    QString getPhone() const { return phone; }
    void setPhone(const QString &value) { phone = value; }
    QString getCountry() const { return country; }
    void setCountry(const QString &value) { country = value; }
    QString getState() const { return state; }
    void setState(const QString &value) { state = value; }
};

#endif // USERMODEL_H
```

Installation

1. Download and install Qt 5.9 or higher: https://www.qt.io/download-open-source/
2. Open the terminal
3. Install Qt Libraries: sudo apt-get install qt5-default (Linux)
4. Make a project clone: git clone https://github.com/HerikLyma/CPPWebFramework.git
5. cd CPPWebFramework/CPPWebFramework
6. qmake CPPWebFramework.pro
7. make
8. make install (use sudo on Linux)

jemalloc optional installation (recommended)

1. Install jemalloc: sudo apt-get install libjemalloc-dev (linux)
2. Enable jemalloc in the .pro file: LIBS += -ljemalloc

Steps to test the C++ Web Framework's examples

1. Open a .pro file from an example using Qt Creator
2. Change the path in the main.cpp file: CWF::CppWebApplication a(argc, argv, "/PATH_TO_EXAMPLE/server/");
3. Run the project
4. Open your browser and type: http://localhost:8080 to check if the server is online

Steps to test the C++ Web Framework's HelloWorldDocker example with Docker containers

1. Install Docker
2. sudo docker run -d -p 80:80 imacellone/cwf-helloworld:1.0
3. Open your browser and type: http://localhost to check if the server is online
4. If you want to test your own examples, please follow the steps demonstrated on the CPPWebFramework/examples/helloworld.Dockerfile file, build and run the image.

cppwebframework mailing list: https://groups.google.com/forum/#!forum/cppwebframework

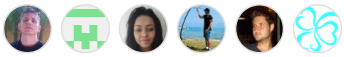
Releases

No releases published

Packages

No packages published

Contributors 6



Languages

C++ 95.8% **QMake** 2.8% **CSS** 0.5% **CMake** 0.4% **C** 0.2% **Roff** 0.2% **Dockerfile** 0.1%