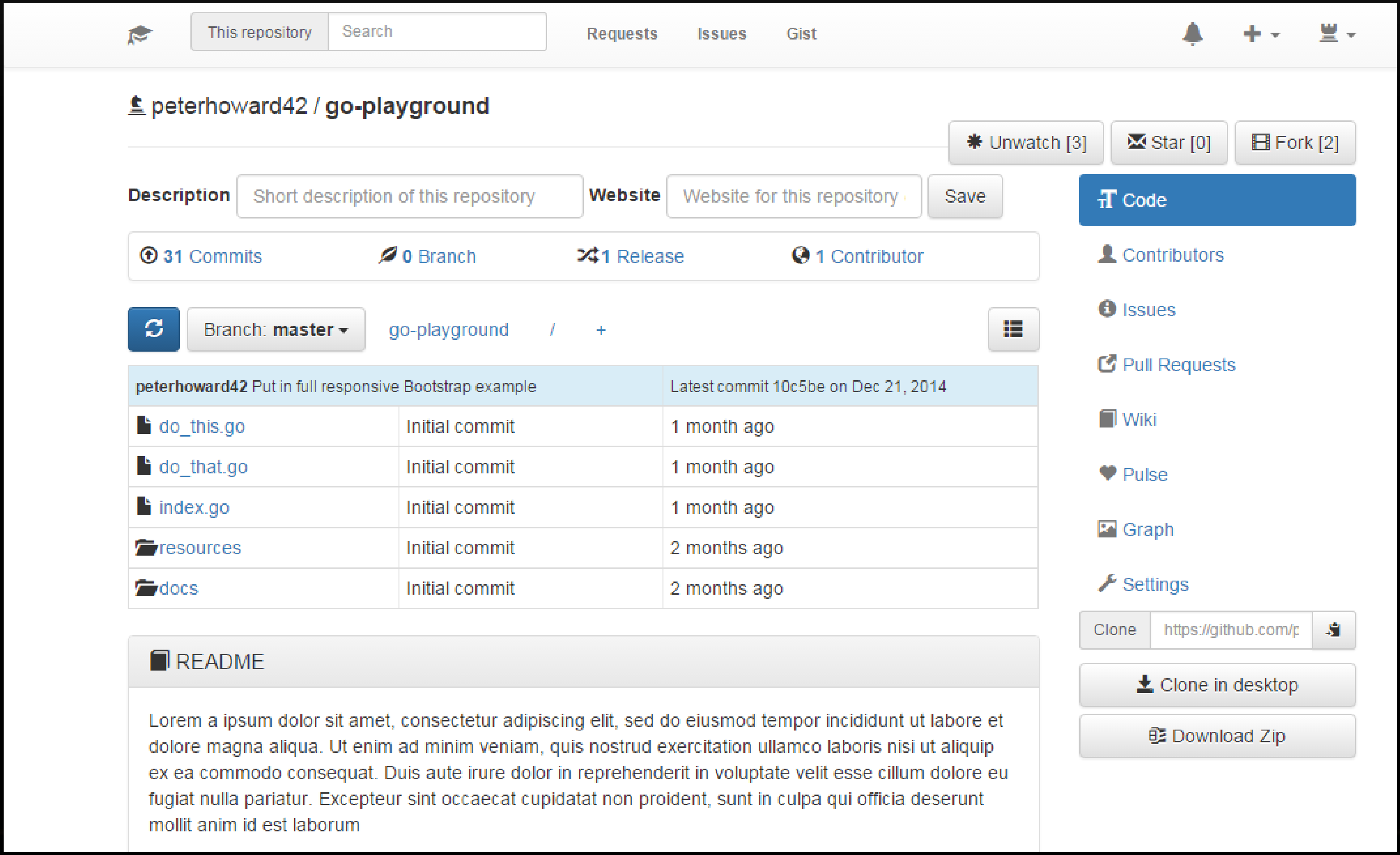
MAKING A STAND-ALONE GUI APP WITH GO

Pete Howard

Altran UK

LIKE THIS GUI [1]



Description: Short description of this repository | Website: Website for this repository | Save

31 Commits | 0 Branch | 1 Release | 1 Contributor

peterhoward42 Put in full responsive Bootstrap example		Latest commit 10c5be on Dec 21, 2014
do_this.go	Initial commit	1 month ago
do_that.go	Initial commit	1 month ago
index.go	Initial commit	1 month ago
resources	Initial commit	2 months ago
docs	Initial commit	2 months ago

Lorem a ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

1. A ROUGH COPY OF GITHUB'S GUI

WHY I LIKE GO

- Extraordinary combination of Simplicity with Power
- Inspired set of language design trade-off decisions [1, 2]

1. HTTPS://GOLANG.ORG/DOC/FAQ
2. HTTPS://TALKS.GOLANG.ORG/2012/SPLASH.ARTICLE

BUT GO DOESN'T HAVE A GUI ☹

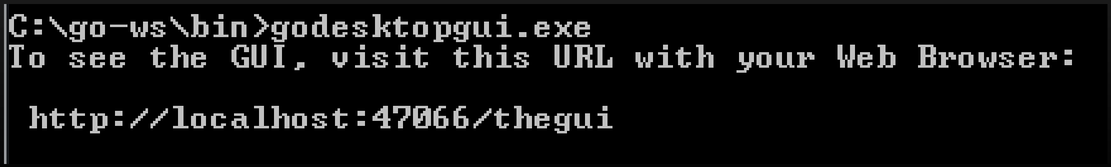
- However, HTML5 + CSS + Bootstrap[1] makes great in-browser GUIs
- And it's real simple to make a local web server in Go
- And to deploy them together as a single executable

- What follows shows you how,
  - and provides the source code
  - and shows a little of Go

1. POPULAR CSS LIBRARY

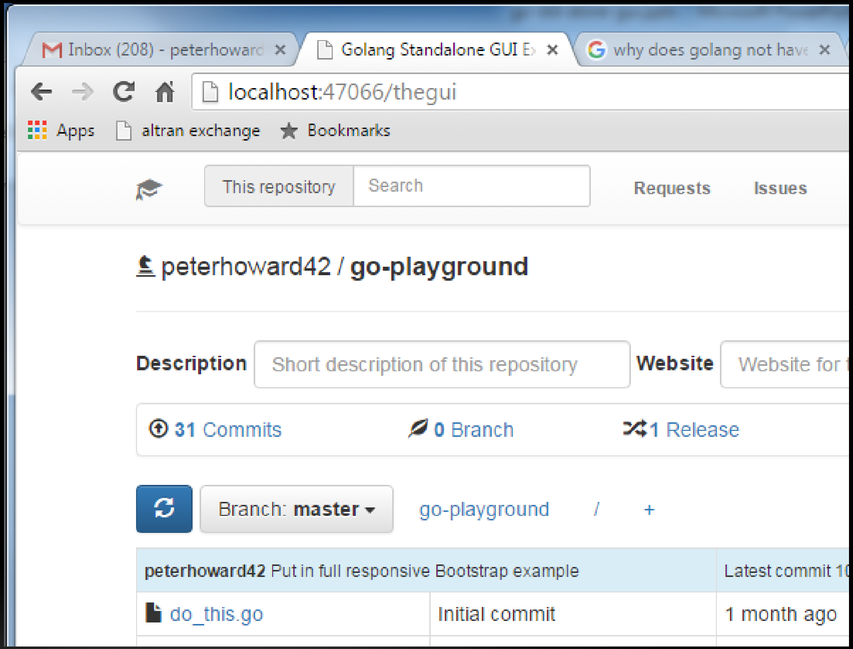
WHAT YOU SEE

Deploy and run just the single executable



```
C:\go-ws\bin>godesktopgui.exe
To see the GUI, visit this URL with your Web Browser:

 http://localhost:47066/thegui
```

And point your browser at localhost



peterhoward42 / go-playground

Description: Short description of this repository
Website: Website for

31 Commits | 0 Branch | 1 Release

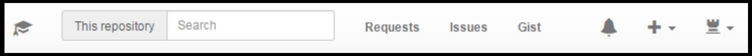
Branch: master go-playground / +

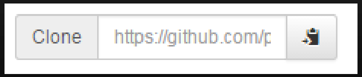
peterhoward42 Put in full responsive Bootstrap example | Latest commit 1

do_this.go | Initial commit | 1 month ago

HTML WITH BOOTSTRAP CSS

Very high-level components

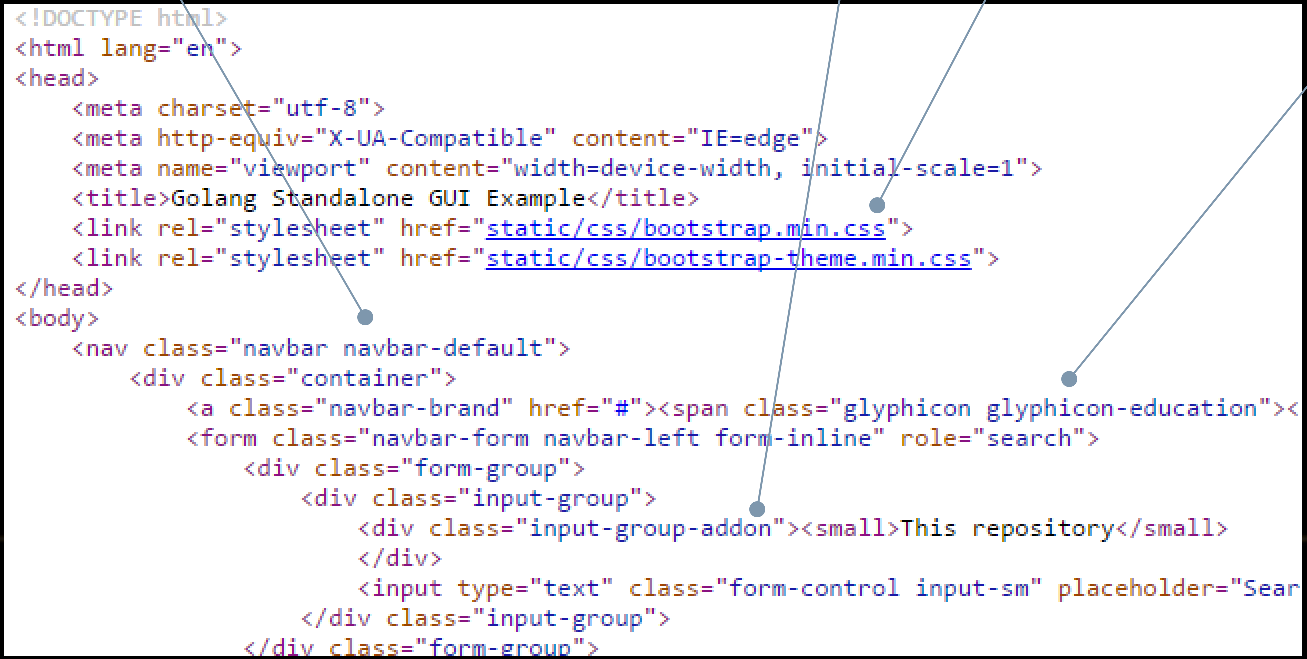




Finely-grained elements

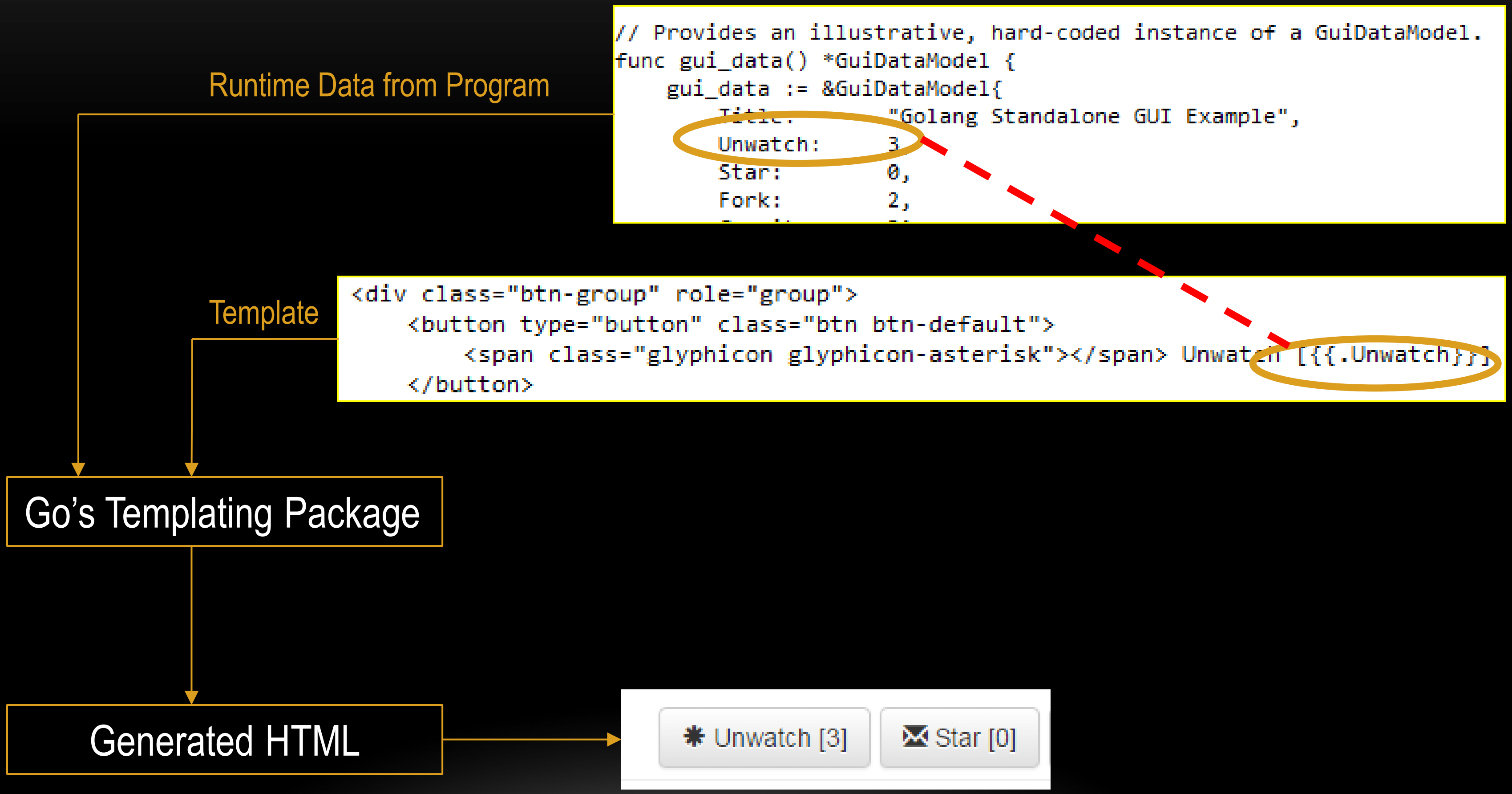
CSS library

Bundled icon graphics



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Golang Standalone GUI Example</title>
    <link rel="stylesheet" href="static/css/bootstrap.min.css">
    <link rel="stylesheet" href="static/css/bootstrap-theme.min.css">
</head>
<body>
    <nav class="navbar navbar-default">
        <div class="container">
            <a class="navbar-brand" href="#"><span class="glyphicon glyphicon-education"><
            <form class="navbar-form navbar-left form-inline" role="search">
                <div class="form-group">
                    <div class="input-group">
                        <div class="input-group-addon"><small>This repository</small>
                        </div>
                        <input type="text" class="form-control input-sm" placeholder="Sear
                    </div class="input-group">
                </div class="form-group">
```

GENERATING HTML FROM GO PROGRAM



```
// Provides an illustrative, hard-coded instance of a GuiDataModel.
func gui_data() *GuiDataModel {
    gui_data := &GuiDataModel{
        Title:         "Golang Standalone GUI Example",
        Unwatch:       3,
        Star:          0,
        Fork:          2,
```

```
<div class="btn-group" role="group">
    <button type="button" class="btn btn-default">
        <span class="glyphicon glyphicon-asterisk"></span> Unwatch [{{.Unwatch}}]
    </button>
```

STARTING POINT FOR A GO WEB SERVER [1]

I'm going to react to http requests on port 12345 on localhost

If the URL is /hello, then…

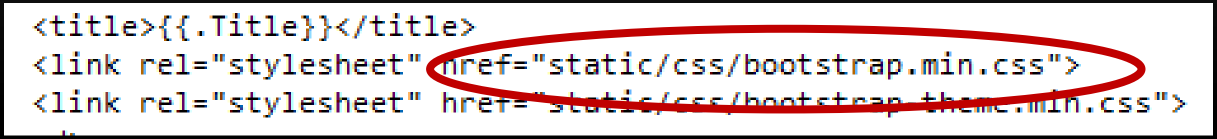
I will send back the old favourite plain text hello world string

```go
package main

import (
        "io"
        "net/http"
        "log"
)

// hello world, the web server
func HelloServer(w http.ResponseWriter, req *http.Request) {
        io.WriteString(w, "hello, world!\n")
}

func main() {
        http.HandleFunc("/hello", HelloServer)
        err := http.ListenAndServe(":12345", nil)
        if err != nil {
                log.Fatal("ListenAndServe: ", err)
        }
}
```

1. THIS IS FROM GO DOCS - AND WILL RUN AS IT STANDS

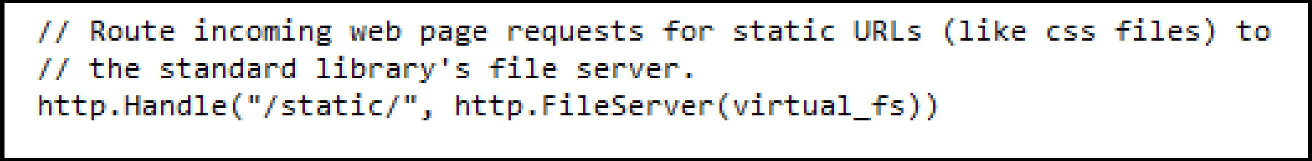
SERVING STATIC FILES

When the HTML page refers to a static link…



```
<title>{{.Title}}</title>
<link rel="stylesheet" href="static/css/bootstrap.min.css">
<link rel="stylesheet" href="static/css/bootstrap-theme.min.css">
```

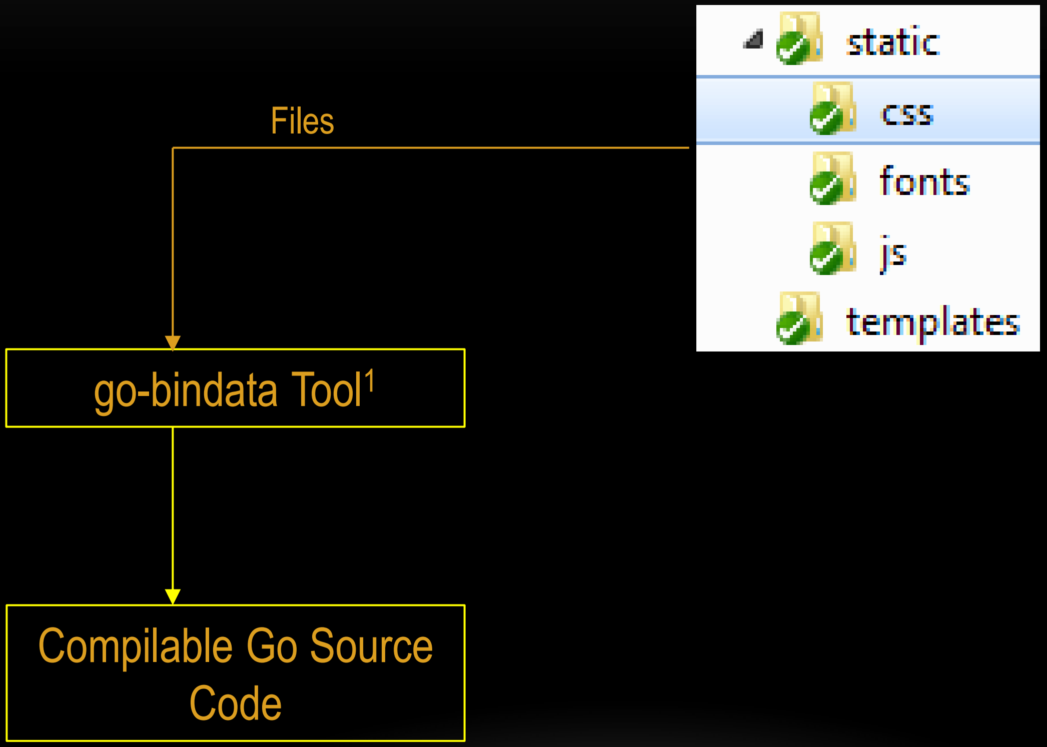
Our server handles it with Go’s http.FileServer



```
// Route incoming web page requests for static URLs (like css files) to
// the standard library's file server.
http.Handle("/static/", http.FileServer(virtual_fs))
```

BUT WE DIDN'T SHIP ANY CSS FILES?

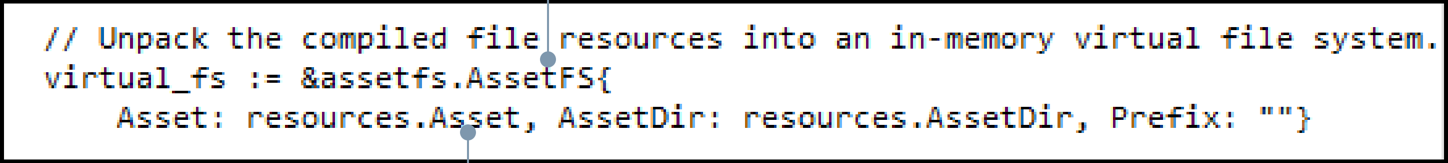
Instead, we compiled them into the app as resources



1. HTTPS://GITHUB.COM/JTEEUWEN/GO-BINDATA

AND UN-MARSHALED THEM BACK INTO A VIRTUAL FILE SYSTEM

https://github.com/elazarl/go-bindata-assetfs



```
// Unpack the compiled file resources into an in-memory virtual file system.
virtual_fs := &assetfs.AssetFS{
    Asset: resources.Asset, AssetDir: resources.AssetDir, Prefix: ""}
```

The compiled resource files

CODE AVAILABLE

Less than 200 lines of code

Get it here: https://github.com/peterhoward42/godesktopgui

(You don't need Git – there is a zip file download button)

Pre-compiled demo binary also available in repository (for Windows)

Contact:

peterhoward42@gmail.com
peter.howard@altran.com