Application Development MCOMD1ADC

Dr. Alexios Louridas

2021-01-20

Contents

Introduction

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports, e.g., a math equation $a^2 + b^2 = c^2$.

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

Chapter 1

Recap

1.1 Module Structure

1.2 Revision

The purpose of revision is not just to remember what you have done previously. Revision helps you understand what you could not on a first pass of a material, it helps you identify concepts and material you did not see before, it helps you deepen your knowledge on a subject matter and of course it helps you remember.

You have now used visual studio as an IDE for a few months now. How well do you think you know it?

1.2.1 Creating or opening an application

How would you create an application what do you need to know before you create one?

1.3 Software Development

1.3.1 Software Development Lifecycle

1.3.2 Intellectual Property

1.3.3 Referencing

1.4 Version Control

While developing software, development teams use tools to help them reduce development times and produce more stable and better solutions. The most

popular of these are version control or source control tools. The aim of these tools is to keep a record of all changes that happen on the source code of an application or applications being developed.

A version control tool keeps a record of all changes while recording the person or team responsible, the time of a change and the content that has been altered. All of these are recorded in a specialised database usually called a repository or repo for short, that can be accessed at any point to review code and provide a fix or an addition to the source code by identifying the best place, time and/or person to do so.

Try and remember how many times have you lost work that you wish you have backed up or you have but you do not remember where it is located, or you have not backed up as frequently as you would have wanted with the specific change you are looking. All of these cases would disappear or be minimised by using a version control tool and having a repository to store all your changes.

In this module you will learn how to use one of the most popular version control tool. Before we go on to play and start creating or obtaining repositories let us see the types of repositories that exist.

1.4.1 Client based model

A single repository exists that users connect and exchange information. The repository is centralised and all repository functions are done centrally. The users connect and obtain a **part** of the working copy of the repo. The users have the ability to change the source files but any changes to the repository would need to be communicated directly to the central repository.

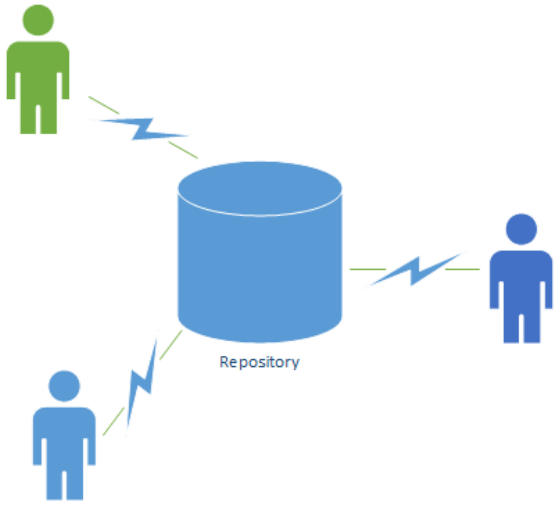

Figure 1.1: Client data model

1.4.2 Distributed based model

Each user has a working **full** copy of the repository. Each user can make changes to the repository locally and thus work offline if required. Once back online the users can distribute all changes to the repository where all users would then have access to view.

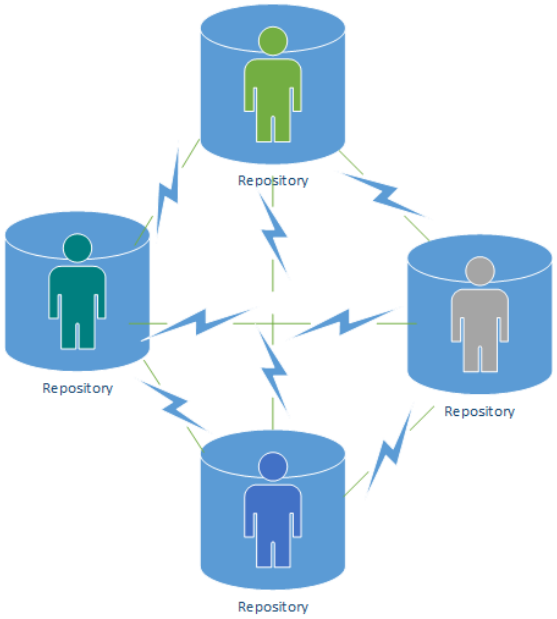

Figure 1.2: Distributed data model

1.5 Git and GitHub

Git is a distributed version control system that has approximately 90% usage in the industry. The main reason of its popularity is its performance compared to its competitors. It provides an outstanding flexibility and security and everything is inbuilt and free. In addition to Git advantages, because of its popularity there are many third party software tools and services that have integrated with Git including IDEs such as visual studio, issue and project tracking software, such as Jira, and code hosting services like GitHub.

1.5.1 How does Git Work

To start understanding what git provides let us look at the following scenario:

A software application is being developed and it is currently in version 1.5 and

the developers are working on version 2.0. Eventually the development finishes and version 2.0 is released. A customer though does not want to go to version 2.0 as they do not have a budget to support the update, but they would like a feature that they have seen in version 2.0 to be included in their version of their software (1.5).

The developers only have to go to Git history and "copy" the required change associated with the feature needed and add it to the release version of 1.5 and thus create a new release 1.5.1. No new code really needed to be developed for what the customer needed.

From this example you can see that Git offers flexibility to control your changes and releases without time constraints.

Let us look at the most common commands that you can issue to git.

config

init

clone

add

commit

diff

stash

status

clean

push

1.5.2 Beyond Source Code

You are a student and a member of a group that is working on a group assignment set by Gordon. You have two progress checks to pass before your final submission. Each member of the group needs to do certain aspects of the assignment but each depends on a part of each others work in order to be able to complete your parts. You make changes to your group assignment, adding some files for the upcoming progress check of your assignment, then **commit** and **push** those changes with descriptive messages in order for your fellow students in your group to be able to do their parts. You then work on a second part of your assignment that is required only for the end report and **commit** those changes too. As you have committed these separately they are also stored separately in the version history of your Git repo.

Another member of the group is having issues with a part of the assignment that is needed to be submitted in the progress check. You decided to help them out so you **pull** the latest repository and fix some of the issues your fellow student

has and **commit** and **push**. The other student can then **pull** the latest and finish his task. Once all students of the group finish the all their changes they **commit** and **push** at their own time and the end product is complete.

When marking Gordon is able to see the **log** and understand how much work everyone has done and how valuable you have been on the team by looking at the **commits** you have done.

At some point in the near future you complete your degree and you are being interviewed for a position in a company. The interviewer asks if you have done something similar to the group assignment that Gordon had set several year ago. You answer "yes" and share your repo which resides in your GitHub account. Another candidate has similar experience but has never used Git or GitHub and thus they cannot share any of their work.

Although you and the other candidate have similar experience and knowledge you get the job just because you can easily show and demonstrate your work. That is all that it takes sometimes to gain or lose a work position.

Chapter 2

Memory and Intorduction to Classes

Chapter 3

Arrays and Collections

3.1 Multi-Dimensional and Jagged Arrays

3.2 Lists

Chapter 4

Objects and Structures

4.1 Classes and Objects

4.2 Members

4.3 Structures

4.4 Methods and Properties

Chapter 5

Algorithms

Chapter 6

GUI and more on Classes

6.1 Introduction to access modifiers (Public and Private)

6.2 Overriding

6.3 User Interface Design Guidelines

6.3.1 General

6.3.2 Windows Forms Specific

6.4 Forms

6.4.1 Add controls in a form

6.4.2 Designer

6.4.3 Controls Types

6.4.4 Programmatically controlling them

Chapter 7

Testing and Data Validation

7.1 Introduction to Test Driven Development

7.2 Introduction to Unit Testing

Chapter 8

File System

8.1 Access files, directories and drives

8.2 Create files and directories

8.3 Write a simple

8.4 Introduction to object relationship