Основы Git

№ урока: 1 **Курс:** Основы Git

Средства обучения: Компьютер

Обзор, цель и назначение урока

Цель урока познакомиться с одной из самых популярных систем контроля версий - Git.

Изучив материал данного занятия, учащийся сможет:

- Создавать репозитории и работать с удаленными репозиториями.
- Создавать коммиты,

Содержание урока

1. Локальная работа с репозиторием
2. Базовые команды

Резюме

- Git - это бесплатная и открытая система контроля версий (SCM). Он создан для отслеживания изменений кода в ваших проектах. Это предоставляет вам контроль на каждом этапе развития вашего приложения.
- Git создан так, чтобы над проектом могла работать команда программистов. Это облегчает координацию задач между сотрудниками и помогает отслеживать изменений кода.
- Git - это просто инструмент. Он не привязан к конкретному языку программирования. С его помощью можно систематизировать код на Python, C#, JavaScript, Java, Ruby и других языках программирования.
- Только языки программирования? Нет, Git способен работать с любым текстом.
- Чтобы не привязывать наш курс к языку программирования, мы будем использовать псевдокод.

Установка

Чтобы начать работу с Git, его нужно установить. Скачайте установщик с официального сайта (https://git-scm.com/), запустите его и следуйте его указаниям. После этого в меню Пуск появятся две новые программы: Git Bash и Git GUI.

Как открыть консоль?

- Вариант 1. Напишите в поисковой строке мень Пуск cmd
- Вариант 2. Нажмите сочетание клавиш Win + R, в появившемся окне введите cmd и нажмите Enter.

Начало работы

- Откройте консоль Git Bash через меню пуск. Чтобы убедиться, что Git установлен корректно выполните команду: `git --version`. Если в ответ вы получите версию установленной программы, Git установлен успешно.
- Так как Git допускает работу в команде, каждому участнику проекта надо "представиться" перед Git'ом, чтобы он мог различать изменения от разных разработчиков.
- Выполните следующие команды, чтобы git узнал ваше имя и электронную почту (измените на свои данные):

```
git config --global user.name "Leonid Podriz"
git config --global user.email "leonidpodriz@gmail.com"
```

Создание пустого проекта

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Подготовка рабочей папки

Начните работу с создания пустого каталога с именем *"code"*, затем войдите в него и создайте там файл с именем `main.code` с таким псевдокодом:

```
выведи("Привет мир")
```

Создание пустого репозитория

- Теперь у вас есть каталог с одним файлом. Чтобы создать git репозиторий из этого каталога, выполните команду `git init`. Результат:

```
git init
```

Добавление файла в репозиторий

- Создание репозитория не значит, что Git сразу же начал следить за всеми файлами в папке. Файлы необходимо добавить вручную. Это дает контроль над файловой структурой Git репозитория и гарантирует, что никакие конфиденциальные данные не попадут в открытый доступ без вашего согласия.
- Теперь давайте добавим в репозиторий файл с кодом.

```
git add main.code
git commit -m "Add my fisrt code file"
```

- Команда git commit - служит для фиксации изменений в коде. Мы можем добавить много файлов с помочью команды git add и только один раз зафиксировать изменения.

Проверка состояния репозитория

Используйте команду `git status`, чтобы проверить текущее состояние репозитория.

```
git status
```

Эта команда возвращает информацию о репозитории:

1. сколько файлов изменено с последнего коммита
2. сколько файлов было добавлена файл, но не добавлено в репозиторий
3. Какие файлы были удалены и так дальше.

По факту, `git status` смотрит на последний коммит и на текущее состояние файлов в рабочей папке. Если есть различия - git сообщает.

Не стесняйтесь по чаще использовать команду `git status` - это может уберечь вас от не преднамеренных ошибок. В нашем случае вы увидите:

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

Команда проверки состояния сообщит, что коммитить нечего. Это означает, что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Мы будем использовать команду git status, чтобы продолжать отслеживать состояние репозитория и рабочего каталога.

Изменения

Теперь наша цель научиться отслеживать состояние рабочего каталога. Изменим код в main.code:

```
выведи("Привет, Мир")
```

В код добавлена запятая и "Мир" написано с большой буквы. Как на это отреагирует Git?

```
C:\Users\38099\Desktop\git_lesson> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.code

no changes added to commit (use "git add" and/or "git commit -a")
```

- Git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории.
- Обратите внимание на то, что сообщение о состоянии дает вам подсказку о том, что нужно делать дальше. Если вы хотите добавить эти изменения в репозиторий, используйте команду git add. В противном случае используйте команду git checkout для отмены изменений.

Обработка изменений

Добавьте изменения

После изменений файлов, надо проиндексировать изменения, выполнив команду git. Проверьте состояние:

```
C:\Users\38099\Desktop\git_lesson>git add main.code

C:\Users\38099\Desktop\git_lesson>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.code
```

- Изменения файла main.code проиндексированы. Git теперь знает об изменении, но изменение пока не перманентно (читай: не навсегда) записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения.
- Если вы решили, что не хотите коммитить изменения, команда состояния напомнит вам о том, что с помощью команды git restore можно снять индексацию этих изменений.

Коммит

- Отдельный шаг индексации в git позволяет вам продолжать вносить изменения в рабочий каталог, а затем, в момент, когда вы захотите взаимодействовать с версионным контролем, git позволит записать изменения в малых коммитах, которые фиксируют то, что вы сделали.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Предположим, что вы отредактировали три файла (main.code, module.code, и math.code). Теперь вы хотите закоммитить все изменения, при этом чтобы изменения в main.code и module.code были одним коммитом, в то время как изменения в math.code логически не связаны с первыми двумя файлами и должны идти отдельным коммитом.

В теории, вы можете сделать следующее:

```
git add main.code
git add module.code
git commit -m "Changes for main and module"
git add math.code
git commit -m "Unrelated change to math"
```

Разделяя индексацию и коммит, вы имеете возможность с легкостью настроить, что идет в какой коммит.

Комитим изменения

- Вернемся к нашему примеру. Мы изменили и проиндексировали файл main.code.
- Ранее мы использовали git commit для коммита первоначальной версии файла main.code в репозиторий. Мы включили метку -m, которая делает комментарий в командной строке.
- Команда commit позволит вам интерактивно редактировать комментарии для коммита. Если вы опустите метку -m из командной строки, git перенесет вас в редактор по вашему выбору.

Выполните:

```
git commit
```

Вы увидите в вашем редакторе:

```
|
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git restore --staged <file>..." to unstage)
#
#   modified:   main.code
#
```

В первой строке введите комментарий: «Fix typo mistakes». Сохраните файл и выйдите из редактора (для этого в редакторе по умолчанию (Vim) вам нужно нажать клавишу ESC, ввести ":wq" и нажать Enter). Вы увидите:

```
C:\Users\38099\Desktop\git_lesson>git commit
[master 52f5a44] Fix typo mistakes
1 file changed, 1 insertion(+), 1 deletion(-)
```

Проверьте состояние

В конце давайте еще раз проверим состояние.

```
git status
```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Вы увидите:

```
C:\Users\38099\Desktop\git_lesson>git status
On branch master
nothing to commit, working tree clean
```

Рабочий каталог чистый, можете продолжить работу.

Истрия

Чтобы получить историю комитов в репозитории используйте команду: `git log`.

Возможный результат:

```
commit 52f5a44d66e6a6040885b6ee73929d9d08e93e65 (HEAD -> master)
Author: Leonid Podriz <leonidpodriz@gmail.com>
Date:   Tue Aug 18 11:11:41 2020 +0300

    Fix typo mistakes

commit e41ae66ba3b20c6f7f4287305a306513c24788d1
Author: Leonid Podriz <leonidpodriz@gmail.com>
Date:   Tue Aug 18 10:33:23 2020 +0300

    Add fisrt code file
```

Версии

- Git система контроля версиями. Пришло время узнать, как же переключаться между нашими версиями (комитами).
- В предыдущем разделе вы узнали, как можно смотреть историю комитов. Предположим, что мне нужно переключиться на самый первый коммит и продолжить работу с него. Для этого мне нужно узнать хеш этого комита.
- Чтобы узнать хеш можно снова воспользоваться коммандой git log и посмотреть на нужный комит. В моем случае это следующий комит:

```
commit e41ae66ba3b20c6f7f4287305a306513c24788d1
Author: Leonid Podriz <leonidpodriz@gmail.com>
Date:   Tue Sep 29 10:33:23 2020 +0300

    Add fisrt code file
```

`e41ae66ba3b20c6f7f4287305a306513c24788d1` - это хеш. Он генерируется в зависимости от изменений файла автоматически.

Чтобы вернуть свой проект к состоянию этого комита выполните команду `git checkout <hash>`:

```
C:\Users\38099\Desktop\git_lesson>git checkout
e41ae66ba3b20c6f7f4287305a306513c24788d1
Note: switching to 'e41ae66ba3b20c6f7f4287305a306513c24788d1'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.
```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:
    git switch -

Turn off this advice by setting config variable advice.detachedHead to
false

HEAD is now at e41ae66 Add fisrt code file
```

- Не стесняйтесь читать результат команд. Они ёмко объясняют все тонкости работы с Git.
- Обратите внимание, что содержимое файла main.code вернулось к состоянию комита.
- Чтобы вернуться к последним изменениям, используйте команду: `git checkout master`:

```
C:\Users\38099\Desktop\git_lesson>git checkout master
Previous HEAD position was e41ae66 Add fisrt code file
Switched to branch 'master'
```

`master` — имя ветки по умолчанию. Переключая имена веток, вы попадаете на последнюю версию выбранной ветки.

Что дальше?

- Git имеет много функций. Чтобы изучить их все, понадобится очень много времени. Мы знакомили вам с самым БАЗОВЫМ функционалом Git.
- Некоторую часть функционала вам, возможно, никогда не придется использовать. Что-то - напротив, вы будете использовать каждый день.
- Если вам понравилось работать с Git и вы желаете освоить его лучше, рекомендую изучить работу с ветками.

Закрепление материала

- Что такое Git?
- Для чего нужен коммит?
- Какая комманда используется чтобы создать git репозиторий?

Дополнительное задание

Задание
Добавьте несколько коммитов после того как выполните Задание 1 из раздела «Самостоятельная деятельность учащегося».

Самостоятельная деятельность учащегося

Задание 1
Создайте пустой проект и добавьте файлы в репозиторий

Рекомендуемые ресурсы

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Работа с ветками в Git https://git-scm.com/book/ru/v2/%D0%92%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-Git-%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D0%B2%D0%B5%D1%82%D0%BA%D0%B0%D0%BC%D0%B8

- Книга «Git для профессионального программиста» (С. Чакон, Б. Штрауб)

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com