DESARROLLO WEB CON PHP 7 Y MYSQL UTILIZANDO MVC

Crear repositorio Git

Ponente: Alejandro Amat Reina

¿Qué es Git?

- Es un sistema de control de versiones
- Nace a partir del proyecto de desarrollo del kernel de Linux
- Está pensado para ser distribuido, rápido y sencillo
- Es capaz de manejar grandes proyectos

Instalación de Git

- Ejecutamos el comando:
  - sudo apt-get install git
- Más información en:
  - https://git-scm.com/book/es/v1/Empezando-Instalando-Git

Obtener ayuda

- Para obtener ayuda usamos el comando:
  - git help
  - git help nombre_comando_git

Configurando git

- Establecer el nombre del usuario:
  - git config --global user.name "Alejandro Amat"
- Establecer el e-mail del usuario:
  - git config --global user.email user_email
- Mostrar colores al ejecutar los comandos:
  - git config --global color.ui true

Iniciando un repositorio local

- En el directorio del proyecto ejecutamos el siguiente comando:
  - git init
- Este comando creará el directorio .git/, en el que se guardarán todos los cambios de los ficheros a medida que vayamos realizando commits

Flujo de trabajo

- Cuando creamos un nuevo fichero en el proyecto nos aparece como fuera de seguimiento (untracked)
- Si lo añadimos al seguimiento pasaría a estar en nuestro escenario o área de trabajo (staging area)
- Si queremos tomar una instantánea del estado del fichero en un momento dado, hacemos un commit de los cambios
- En el commit se guardarán los cambios existentes en todos los ficheros que están en el escenario desde que se realizó el anterior commit

Ver el estado del repositorio

- Ejecutamos el comando:
  - git status
- Nos dirá qué ficheros tenemos en seguimiento y qué nuevos ficheros se han creado y están sin seguimiento

Añadir un fichero al área de trabajo

- Ejecutaremos el comando:
  - git add index.php
- El fichero index.php pasará a estar en seguimiento
- Cuando hagamos un commit se guardarán sus cambios

Guardar los cambios

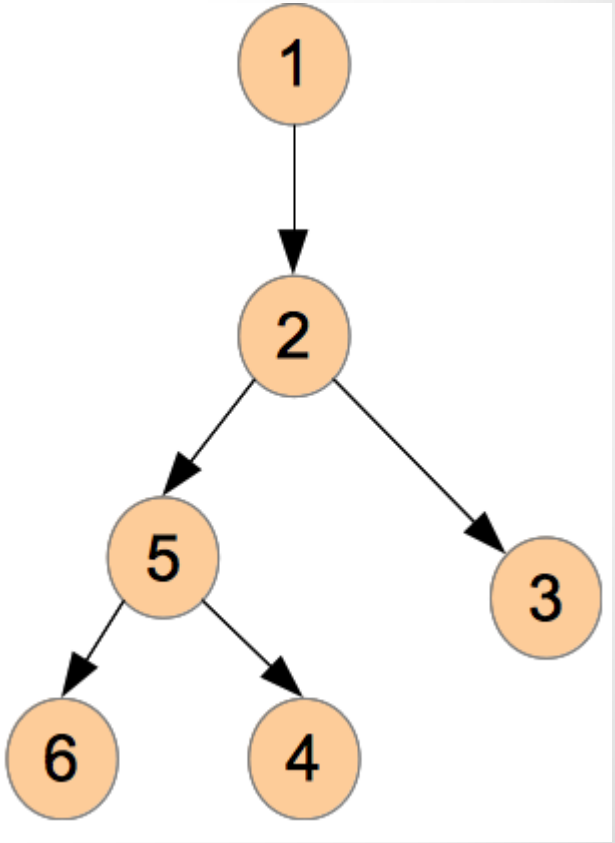
- Para hacer un commit ejecutamos el comando:
  - git commit –m "mensaje explicativo del commit"
- Se guardarán los cambios y se creará una instantánea del estado del proyecto en el repositorio local
- Si después de hacer un commit ejecutamos el comando git status veremos que no aparece ningún cambio

Añadir varios ficheros al escenario

`$ git add <list of files>`	Add the list of files
`$ git add --all`	Add all files
`$ git add *.txt`	Add all txt files in current directory
`$ git add docs/*.txt`	Add all txt files in docs directory
`$ git add docs/`	Add all files in docs directory
`$ git add "*.txt"`	Add all txt files in the whole project

Funcionamiento interno

- La estructura interna de un repositorio Git es un grafo acíclico dirigido:
  - Conjunto de elementos llamados nodos unidos por enlaces llamados aristas que tienen una dirección definida, no permitiéndose además que existan cíclos



Grafo

- Para simplificar, podemos interpretar que cada uno de los nodos es un commit
- Cada commit tiene un id único (SHA1) calculado a partir de los contenidos del mismo
- Al hacer un commit nos da una serie de información:
  - [master (root-commit) 716cb49] commit inicial
    - *master*: rama en la que el commit ha sido insertado. Estudiaremos las ramas más adelante. Inicialmente sólo hay una rama, la rama master
    - *root-commit*: indica que el commit es el primero, el objeto raíz del grafo
    - *716cb49*: primeros dígitos del identifcador del commit
    - *commit inicial*: mensaje descriptivo de lo que incluye ese commit
- Cada commit tiene también una referencia al commit anterior (arista)

HEAD

- El grafo de git tiene un puntero (HEAD) que apunta al commit en el que estamos trabajando actualmente
- De esta forma podemos movernos por los distintos commits, simplemente modificando el puntero HEAD

Historial

- Para ver el historial de commits de nuestro repositorio ejecutamos el comando:
  - git log

Diferencias

- Para ver las diferencias desde el último commit hacemos:
  - git diff: Esto nos da las diferencias siempre que no hayamos hecho git add
  - Git diff --staged: Esto nos da las diferencias aunque hayamos hecho git add

Sacar del escenario

- Si queremos sacar un fichero del área de trabajo hacemos:
  - git reset HEAD fichero
  - Cuando hacemos git status nos da ayuda para realizar esta operación

Descartar cambios

- Para descartar los cambios de un fichero del escenario y volver a la versión del commit anterior:
  - Git checkout -- fichero

Commit + staging

- Si queremos añadir todos los ficheros modificados al escenario y hacer commit al mismo tiempo:
  - git commit –a –m "mensaje explicativo del commit"
  - **OJO!!** Esto no añadirá al escenario ficheros nuevos, sólo aquellos que ya estaban en el escenario en el anterior commit

Añadir un fichero al commit anterior

- Si después de hacer commit descubrimos que se nos ha olvidado añadir algún fichero podemos añadir ficheros al último commit:
  - git commit --amend –m "mensaje explicativo del commit"
  - Esto también sirve para cambiar el mensaje del último commit

Deshacer commits

- Podemos deshacer un commit de dos formas:
  - git reset --soft HEAD^
    - Deshace el último commit y deja los cambios que habían en el escenario
  - git reset --hard HEAD^
    - Deshace el último commit y elimina los cambios
- Para deshacer más de un commit:
  - git reset --hard HEAD^^
    - Deshace dos commits

Repositorios remotos

- Cuando más de un desarrollador trabaja en el mismo proyecto, debemos tener un repositorio central en el que vayamos unificando los cambios que realicen los distintos desarrolladores
- Para esto se utilizan los repositorios remotos

Añadir repositorios remotos

- git remote add origin url_repositorio_remoto

  Origin: Nombre del repositorio remoto

Mostrar repositorios remotos

- git remote -v

Subir los cambios al repositorio remoto

- git push –u origin master
  - Subirá los cambios existentes en la rama master al repositorio origin
  - Nos preguntará el usuario y password del repositorio remoto

Descargar los cambios del repositorio remoto

- git pull
  - Esto descargará los cambios existentes en la rama master del repositorio origin
  - Podría darse el caso de que hubieran conflictos, esto lo estudiaremos más adelante

Eliminar repositorio remoto

- git remote rm nombre

IMPORTANTE

- Después de hacer push nunca debemos ejecutar uno de estos comandos:
  - git reset --soft HEAD^
  - git reset --hard HEAD^
  - git commit --amend -m "mensaje del commit"
  - git reset --hard HEAD^^

Clonar un repositorio remoto

- En ocasiones nos puede interesar clonar un repositorio remoto en una máquina para poder disponer de él en local, por ejemplo:
  - Si tenemos un equipo de trabajo de más de una persona, uno creará el proyecto y el repositorio y los demás lo clonarán
  - Si trabajamos en más de un equipo, podemos clonar el proyecto en más de un equipo para poder trabajar en distintos sitios (por ejemplo en la oficina y en casa) con el mismo proyecto

Git clone

- Necesitamos conocer cuál es la url del repositorio remoto
- Bastará con ejecutar el comando git clone pasándole dicha url:
  - git clone url_repositorio_remoto
- Si queremos cambiar el directorio donde se creará el repositorio local, podemos pasarle el nombre del directorio como segundo parámetro
  - git clone url_repositorio_remoto nuevo-dir

¿Qué hace git clone?

- Descarga el repositorio remoto al directorio local
- Añade el repositorio origin apuntando al remoto del cual hemos clonado
- Activa la rama master y pone el HEAD apuntando al último commit de dicha rama