Bestandsorganisatie & Setup

Organisatie van de bestanden

Het originele bronmateriaal bestaat uit:

- de door de studenten ingescande pdf
- de originele "digitally born" Word-files

De ingescande pdf bevat zowel kopieën van stencils met typoscripten uit ca 1978, als kopieën van de uitgeprinte Word-files uit ca 1994. Voor die laatste zullen we op later tijdstip trachten een conversie te doen van `.doc` → `.docx` (= xml) → [via Pandoc?] → (La)TeX → Markdown. Nu concentreren we op het digitaliseren van de stencils uit ca 1978.

De OCR output in één enkele file (`/bronmateriaal/originele dump/PDF cursus en OCR output/Stromingsleer.txt`) werd opgesplitst in meerdere, beter handelbare bestanden, op basis van de paginering (= proxy voor hoofdstukken). Deze bestanden kregen als *file name* hun resp. *page range*, bvb `E.1-E.40.txt` bevat de tekst die op originele stencils staat op de bladzijden van E.1. tot E.40. De bestanden leveren de basis voor editeerwerk en zijn te vinden onder `/rauwe OCR/`.

Bedoeling is deze bestanden één per keer te bewerken door er een kopie van te nemen en alvast de bestandsextensie `.txt` te veranderen in `.md` (voor Markdown syntax). Best plaatst ge elk dergelijk bestand (kopie van de originele OCR output) in een eigen mapje, dat je benoemt volgens het betreffende hoofdstuk.

Ik ben bij wijze van voorbeeld begonnen met bestand `E.1-E.40.txt`, gekopieerd als `E.1-E.40.md` en geplaatst in z'n eigen mapje `HOOFDSTUK E - Vloeistoffen`. Dat mapje bevat een mapje (subdirectory/folder) `figuren` waarin de afbeeldingen staan die gebruikt worden in de tekst.

Logistieke Setup

Dropbox Alle bestanden van het project staan verzameld in één enkele map `Stromingsleer`. Die map wordt gesynchroniseerd met Dropbox in een gedeelde map waarop gij, Pieter en ikzelf toegangsrechten hebben.

Ge hebt Dropbox op uw machine geïnstalleerd en de gedeelde map `Stromingsleer` gekopieerd. Alle bestanden zijn dus lokaal op uw machine beschikbaar en bewerkbaar. Telkens ge een bestand bewerkt (de inhoud ervan, de bestandsnaam of de plaats van het bestand binnen de directory/folder structuur) zal Dropbox (het programmaatje op uw PC) die wijzigingen synchroniseren met de kopie van die bestanden op de server van Dropbox (i.e. "in the cloud). Zodra ik online ben, zal Dropbox (de server) die wijzigingen

"pushen" naar de kopie die ik van die bestanden heb op mijn computer, zodat ik eveneens de meest recente versie heb. (En vice versa.) Op die manier blijven we dankzij Dropbox "in synch".

Dropbox beschikt echter over zeer rudimentair versioneringsbeheer. D.w.z. Dropbox zal voor elk bestand een versiegeschiedenis bijhouden (max. 30 dagen) waardoor per ongelijk gewijzigde of verwijderde bestanden kunnen worden teruggehaald. Dropbox stelt ons echter niet in staat om gedetailleerd (op paragraaf-, lijn- of tekenniveau) de editeergeschiedenis op te vragen. Daarvoor gebruiken we Git en Github.

Git(hub) Git is versioneringssoftware ontworpen om verschillende software-programmeurs te laten samenwerken aan een project met vele verschillende files, zonder dat ze elkaar voor de voeten lopen en elkaars wijzigingen overschrijven. Er wordt een *repository* ("repo") aangemaakt voor het project, die voor een aangeduide folder (de projectfolder) alle wijzigingen zal bewaren. Elke medewerker moet de Git-software op z'n machine geïnstalleerd hebben. De Git-software houdt de projectfolder continu in het oog (folder file change watch) en zal alle wijzigingen op een "stapel" leggen; de gebruiker kan vervolgens de reeks wijzigingen evalueren en beslissen ze naar de *repo* te "committen." Zodra een *commit* gedaan wordt, zitten de wijzigingen in de versiegeschiedenis van de repo.

Indien meerdere medewerkers samenwerken aan een project moeten ze een manier hebben om hun versiegeschiedenissen met elkaar te synchroniseren. Net zoals bij Dropbox, gebeurt dit over het internet, met een kopie van de gehele repo ergens op een centrale plaats "in the cloud" (i.e. ergens op een server). Ge kunt zelf, op uw eigen server, een Git-server opzetten, maar meestal wordt gebruik gemaakt van een third-party service. Wij gebruiken Github.

Ik heb voor het project een repo aangemaakt (op mijn machine) waardoor thans alle wijzigingen door Git (op mijn computer) in de versiegeschiedenis bewaard worden, zodra ik ze commit. Ik heb op Github vervolgens een kopie van die repo aangemaakt, die als centrale hub zal dienen waarin we alle bestanden van het project geversioneerd bewaren.

Wanneer ik wijzigingen commit naar de repo op mijn machine worden ze (i.t.t. bij Dropbox) daarmee nog niet meteen en automatisch gesynchroniseerd met de kopie in de centrale repo op Github. Daartoe moet ik manueel en bewust mijn commits naar die centrale repo op de servers van Github "pushen". Als ik *push* worden alle gewijzigde files, samen met hun versiegeschiedenis in verschillende commits, gekopieerd naar de repo op Github en zijn ze beschikbaar voor de andere medewerkers. Die kunnen vervolgens (opnieuw manueel en bewust) deze meest recente versie(s) "pullen" om "in synch" te komen.

Kortom Het zou ons veel te ver leiden om Git te installeren op uw PC en u te leren *committen*, *pullen* en *pushen*. Het is voldoende dat ge weet dat we het

hele project versioneren met Git en de files ervan centraal bewaren op Github.

Onze werkwijze is als volgt: ge bewerkt naar believen de bestanden, lokaal op uw PC, in de map `Stromingsleer` en Dropbox zal ze (mits ge online zijt) voor u automatisch in de achtergrond synchroniseren met de kopie op de Dropbox-server. Ik krijg dan meteen die gesynchte wijzigingen doorgevoerd in de kopieën van de bestanden op mijn computer. Vervolgens neem ik op regelmatige basis "snapshots" van het project door ze (bewust en manueel) te committen en vervolgens te pushen naar de repo op Github.

Note bene! Alleen mijn commits worden dus geversioneerd! Er zal dus geen versiegeschiedenis beschikbaar zijn tussen de snapshots in.

Afbeeldingen en grote bestanden (binary files) Alle bestanden in de projectfolder `Stromingsleer` worden in Dropbox gesynched, maar niet alle bestanden worden in de versiegeschiedenis (de repo) bewaard. De vele honderden grote beeldebestanden (`.jpg`, `.pcx`, `.bmp`, `.gif`, ...) zouden de repo immers nodeloos groot maken. We zullen alleen de bewerkte, definitieve figuren in de repo bewaren (als `.png`-bestanden).