Kehitysympäristön pystyttäminen

Kehitysympäristö kuntoon

1. Rekisteröidy githubiin. http://github.com/
2. Ilmoita githubin käyttäjätunnuksesi Juhalle.
    - Sähköpostilla juha.louhiranta@helsinki.fi
    - tai githubissa viesti käyttäjälle charn niin lisään sinut githubissa repositoryyn, jonka jälkeen sinulla on muokkausoikeudetkin.
      Ilman oikeuksia ei projektia pysty hakemaan alla myöhemmin mainitulla osoitteella.
3. Lataa eclipse, varmaan kannattaa ottaa suoraan se "Eclipse IDE for Java EE Developers", niin on kaikilla sitten melkolailla sama ympäristö.
4. Asenna Git (jos ei ole jo)
    - Linux: `sudo apt-get install git-core`
      Muita hyödyllisiä paketteja ovat mm. `gitk ja git-gui`
    - Windows: http://code.google.com/p/msysgit/
      Kannattaa varmaankin asentaa asennusohjelman tarjoamilla oletusvalinnoilla.
5. Generoi itsellesi ssh avain. Katso ohjeet alempaa kohdasta: SSH-avaimen generointi

Linux:
Mene konsolissa kansioon johon tahdot gitillä imaista projektin githubista (esim. eclipsen oletuskansio projekteille):

```
git clone git@github.com:charn/EoS-Addressbook.git
```

Windows:

- Käynnistä Git Bash
- Kirjoittamalla pelkästään cd pääset kotihakemistoosi (esim. C:/User/Käyttäjä/).
- Mennään eclipsen projekteille tarkoitettuun oletuskansioon:
  `cd workspace`
- Kirjoita komento:
  `git clone git@github.com:charn/EoS-Addressbook.git`

Tästä eteenpäin:

Kun olet asentanut Eclipsen niin git:in avulla haetun/kloonatun projektin saa auki Eclipsellä
`File -> Import -> General -> Existing Projects into Workspace`
- Select root directory -> 'Tähän projektin kansio' -> Finish

Kehitysympäristön pitäisi nyt olla pystyssä! ;)

SSH -avaimen generointi

Linux:
Suorita konsolissa komento: `ssh-keygen`
- Luo kaksi tiedostoa kotihakemistossasi olevaan .ssh -kansioon.
- id_rsa.pub on julkinen avaimesi, joka tulee laittaa githubiin.

- id_rsa on salainen avaimesi, jota ei kannata levitellä minnekään.

Windows:
Msysgit on siitä kiva paketti, että siinä tulee melkolailla kaikki tarpeellinen mukana.

Käynnistä Git Bash ja kirjoita komentoriville: `ssh-keygen.exe`
- Luo Linuxin tapaan kaksi tiedostoa käyttäjäkansiossasi sijaitsevaan .ssh -kansioon (esim. C:/Users/Juha/.ssh/)

Versionhallinta - Git

Komennolla `gitk` on helppo tarkastella versiohistoriaa. Komennolla `git gui` pystyy graaffisesti hallitsemaan commitin tekoa.

Mac:
- GitX - http://gitx.frim.nl joku kiva graafinen kilke Macille

Manuaaleja

Ensisijaisesti kannattaa varmaankin tutustua valmiisiin tietolähteisiin ja hiukan siihen, mistä gitissä on oikein kysymys. Kirjoittelen loppuun muuttamia peruskomentoja, mutta en listaa läheskään kaikkia mahdollisia. Komennot toimivat ainakin Linuxin puolella.

Versionhallintakurssin Git -kalvot - syksy 09 - http://cs.helsinki.fi/u/paksula/versionhallinta/s09/git.pdf

Gitin omaa dokumentaatiota - http://git-scm.com/documentation
- Kohtalaisen hyvä tutoriaali täällä on **official Git tutorial**

Ongelmatilanteissa Googlesta löytyy aika hyvin apua. Varsinkin komentojen kanssa.

Peruskomennot, joita itse yleensä käytän (Linux - komentorivi):

Haluat ladata itsellesi jo olemassa olevan repositoryn eli kloonata:

```
git clone git@github.com:charn/EoS-Addressbook.git
```

Uuden repositoryn tekeminen (tätä ei tietysti tarvitse nyt tehdä koska se on jo olemassa, mutta jos haluat testailla asioita, niin näin se onnistuu)**:**

```
mkdir gitrepo
cd gitrepo
git init
```

Muutosten lisääminen indexiin (eli ensin kerrotaan gitille, mitkä tiedostot aiotaan lisätä seuraavaan committiin)**:**

```
# Lisää kaikki kansiossa ja alikansioissa olevat muuttuneet tiedostot
git add .
```

Commitin tekeminen (Commitit menevät omaan paikalliseen repositoryyn)**:**

```
# jos haluaa antaa commitviestin komentorivillä
git commit -m 'Viesti'
```

tai

```
git commit # Viesti annetaan editorissa
```

Haluat lähettää kaikki tekemäsi commitit muiden nähtäväksi (eli githubiin), niin käytä komentoa:

```
git push
```

Haluat päivittää oman repositoryn keskusreposta eli hakea muiden tekemät muutokset:

```
git pull
```

Teit virheen:

```
# Kaikki viime commitin jälkeiset muutokset tuhotaan ja working copy
palautetaan viimeisimmän commitin tilaan.
git reset --hard

# Kaksi viimeistä tekemääsi committia olivat huonoja, joten haluat päästä
niistä eroon ja palata niitä edeltävään tilaan.
git reset --hard HEAD^^

# Olet tehnyt muutoksia useisiin eri tiedostoihin ja haluat palauttaa vain
yhden tiedoston edellisen commitin tilaan.
git checkout filename
```

Teit virheen ja kerkesit jo lähettämään muutokset keskusrepoon:

```
# Tekee uuden commitin jossa peruutetaan viimeisin muutos. Tämän voi sitten
pushata keskusrepoon.
git revert HEAD
```

Huom. Esimerkiksi komento `git revert HEAD^` peruuttaa vain toisiksi uusimman commitin mutta ei uusinta.

Git tukee myös brancheja eli voit tehdä projektista sivuhaaran ja kehittää sitä erillään master branchistä. Itse luulen, että selviämme kyllä yhdellä branchillä tässä kurssissa, mutta kannattaa ehkä tutustua tähän ominaisuuteenkin.