# Gestione del proxy nei laboratori

**Autore: Alberto Regosini**

Per gestire il problema del proxy nei laboratori:

- Installare Fiddler (versione classic) https://www.telerik.com/fiddler/fiddler-classic

I vantaggi di questo programma sono principalmente due:

- Si installa a livello utente, quindi senza permessi di amministratore
- Rileva le impostazioni del proxy impostato nel sistema e si autentica in modo automatico usando direttamente le credenziali dell'utente loggato

Passaggi per l'installazione e cosa impostare perché i nostri strumenti di sviluppo possano finalmente funzionare. Dopo aver installato Fiddler, al primo avvio:

- appare una dialog con un messaggio di warning. cliccare su *Cancel* per evitare che tutte le volte mostri lo stesso messaggio
- sotto il menu *File* togliere la spunta a *Capture Traffic* (o premere F12). Non ci interessa catturare il traffico…
- sotto il menu *Rules* mettere la spunta alla voce *Automatically authenticate* (questo è importantissimo)
- Aprire la schermata delle impostazioni sotto il menu *Tools->Options…* e nella scheda *Connections* togliere la spunta alla voce *Act as system proxy on start up* (anche questo è importantissimo). Qui si vede anche che il proxy è attivo sulla porta 8888
- Riavviare Fiddler per avere le impostazioni attive.

# Come settare i vari programmi che hanno bisogno del proxy

## npm per nodejs

Dal terminale eseguire i seguenti comandi:

```
npm config set proxy http://localhost:8888
npm config set https-proxy http://localhost:8888
```

questi comandi creano il file `~/.npmrc`

## git

Dal terminale eseguire i seguenti comandi:

```
git config --global http.proxy http://localhost:8888
git config --global https.proxy https://localhost:8888
```

### Intellij e Netbeans

Settare il proxy locale nelle impostazioni di Intellij e Netbeans non basta poiché queste vengono usate dall'IDE solo per i propri aggiornamenti e per scaricare le versioni di Java. Serve configurare anche:

### gradle

Creare (se non esiste) il file `~/.gradle/gradle.properties` e aggiungere i dati per il proxy:

```
systemProp.http.proxyHost=localhost
systemProp.http.proxyPort=8888
systemProp.https.proxyHost=localhost
systemProp.https.proxyPort=8888
```

notare la s in https nelle ultime due righe

### maven

Creare (se non esiste) il file `~/.m2/settings.xml` e aggiungere i dati per il proxy:

```
<settings>
  ...
  ...
  <proxies>
   <proxy>
      <protocol>http</protocol>
      <host>localhost</host>
      <port>8888</port>
    </proxy>
  </proxies>
  ...
  ...
</settings>
```

### pip per python

Per usare pip dietro proxy ci sono vari metodi:

1. impostare le seguenti due variabili d'ambiente a livello utente:

```
HTTP_PROXY con http://localhost:8888
HTTPS_PROXY con http://localhost:8888
```

2. da terminale eseguire i seguenti comandi (e non so quale file modifica):

```
set http_proxy=http://localhost:8888
set https_proxy=http://localhost:8888
```

3. direttamente dal comando pip:

```
pip install --proxy http://localhost:8888 somepackage
```

### Composer per PHP

Scaricare il pacchetto di installazione di Composer, scegliere di installarlo come utente locale e nella schermata che chiede il proxy selezionare *Ignore system proxy settings* e inserire a mano nella casella che chiede il proxy la stringa 127.0.0.1:8888.

**Attenzione:** nelle ultime versioni viene richiesto lo scaricamento dei pacchetti tramite HTTPS e l'installatore va a settare solo una variabile d'ambiente relativa ad HTTP. Bisogna quindi aprire il pannello delle variabili d'ambiente e aggiungere la variabile *https_proxy* con valore sempre 127.0.0.1:8888.