Programmazione per il Web
Note per la correzione del progetto

Igor Melatti

Installazione

- Verificare di avere Python 2.x, con i seguenti moduli installati:
  - lxml (almeno versione 2.3.2)
  - mechanize (almeno versione 0.2.5)
  - cookielib
  - xml
  - psycopg2 (almeno versione 2.4.5)
- Copiare questa intera directory all'interno della Web Application (quindi, allo stesso livello di WEB-INF, jsp e html), mantenendo il nome `correzione`.
- Modificare la seconda riga del file `test.txt`, sostitendo `XXXXXXX` e `YYYYYYY` con lo username e la password che vengono richiesti dalla pagina `http://localhost:8080/manager/html`
- Modificare il file `test.txt`, rimpiazzando ovunque la stringa `MMMMMMM` con il numero di matricola
- Assicurarsi di aver creato il DB, usando il file `files/db_ddl.sql` (sotto Linux, è sufficiente eseguire il comando `psql -f files/db_ddl.sql template1`). Per essere sicuri che il tutto vada bene, va ricreato prima di ogni re-invocazione del correttore.

1 Esecuzione

Da terminale, eseguire il comando:

```
python main.py
```

2 Per non far fallire la correzione

- Modificare `server.xml` (sotto Linux è nella directory `/var/lib/tomcat8/conf`), cambiando `autoDeploy="true"` in `autoDeploy="false"`; è necessario riavviare il server tomcat8 dopo queta modifica (sotto Ubuntu: `sudo service tomcat8 restart`).
- Non mettere namespaces, né nella root di `web.xml` (che quindi dev'essere semplicemente `<web-app>`) né nel tag `html` delle pagine HTML.
- Non usare *mai* il tag `<br>` o `<br/>` (al suo posto usare, ad esempio, `<p>, </p>`).
- È inutile modificare i file dati, dal momento che la correzione verrà poi ripetuta dal docente sul suo computer. Tuttavia, il file `test.txt` può essere temporaneamente cambiato per aggiungere delle righe di debug come `save` (vedere sotto).

3 Per capire come funziona `main.py`

Ogni riga di `test.txt` è un comando per un emulatore di browser (nel seguito denominato *browser virtuale*); i significati di alcuni comandi sono riportati qui di seguito:

- Le righe che cominciano con `#` sono commenti.
- Ogni comando che abbia l'effetto di cambiare l'attuale pagina del browser virtuale effettua anche un controllo sulla validità dell'HTML ritornato.
- Una riga `open|url` ha l'effetto di aprire `url` scrivendolo sulla barra degli indirizzi del browser virtuale.
- Una riga
  `form|nomeForm|submitForm|nomeInput+|"valInput1"...|"valInput`n`"|nomeInputBis1|"valInput"`
  ha l'effetto di trovare il form di nome `nomeForm`, settare l'input multiplo di nome `nomeInput` ai valori `valInput1`, ..., `valInput`n e l'input singolo di nome `numeInputBis` al valore `valInput`, e poi sottomettere il form cliccando sul pulsante HTML di nome `submitForm`. Notare che:
  - possono essere settati molteplici input, sia singoli che multipli; è possibile anche non dare alcun input;
  - gli apici `"` che circondano i valori vengono scartati al momento del settaggio;
  - i caratteri `+` e `1` che seguono i nomi degli input vengono scartati anch'essi (servono, rispettivamente, a denotare un input multiplo od un input singolo);

  - se il form ha un solo pulsante di sottomissione, allora `submitForm` può essere omesso.

- Una riga `link|nomeLink` clicca sul link di nome `nomeLink`.
- Una riga `save|nomeFile` salva l'attuale pagina del browser virtuale sul file `nomeFile` (*utile per debugging*).
- Una riga `check_present|val1|...|valn` cerca le stringhe `val1` ... `valn` nell'attuale pagina del browser virtuale (nell'ordine dato).
- Una riga `reload` fa il `reload` dell'intera Web Application (ad esempio, in seguito al cambiamento del `web.xml`).
- Una riga `check_if_equal|url1|url2` controlla che il risultato dell'aprire (scrivendo gli URL sulla barra degli indirizzi) tanto `url1` quanto `url2` sia identico (a livello di sorgente HTML).