Configuration à prototypes

FIL — M2GL

Les formats `json`, `toml`, `yaml`, ou `ini` ayant tous différents inconvénients, on décide après mûre réflexion de remédier à ce problème en concevant un nouveau format : *NON*[1], qui décrit des *objets* définis par un *identifiant* et composés de *champs*.

```
univ:
.name 'Université Exemple'
.domain 'exemple.tld'

student:
.name .login
.mail .login '.etu@' univ.domain
.login @

alice: student
bob: student
.name 'robert'
```

Syntaxe Le format est basé sur les lignes du fichier. Pour simplifier, on limite les identifiants à une séquence de lettres, et on fera abstraction des séquences d'échappement dans les chaînes. Une déclaration d'objet commence par un identifiant en début de ligne, suivi par deux points. Ici, on a donc quatre objets `univ`, `student`, `alice`, et bob. Chaque ligne commençant par un *point* définit un champ de l'objet en cours. Après l'identifiant, le reste de la ligne spécifie la valeur du champ, qui peut être une chaîne de caractères, la valeur d'un autre champ, ou une concaténation :

- `'abc'` : chaîne litérale entre guillemets,
- `@` : identifiant de l'objet,
- `.foo` : valeur du champ `foo` du même objet,
- `obj.foo` valeur de `foo` pour l'objet `obj` du même ensemble d'objets,
- hors des chaînes, l'espace dénote la concaténation.

Les objets peuvent être définis à partir d'un autre objet modèle, dont l'identifiant est donné après les deux points ; cela est équivalent à recopier les définitions des champs du modèle. L'objet `alice` a donc les trois champs `login` (la chaîne `alice`), `name` (même chaîne), et `mail` (`alice.etu@example.tld`). Les définitions de champs de l'objet prennent précédence sur celles du modèle, donc l'objet bob a un `name` (`robert`) différent de son `login` (bob).

L'ordre des déclarations n'a d'impact ni sur les champs ni sur les objets : la valeur d'un champ est déterminée à la demande ; l'accès à un champ ou un identifiant d'objet inconnu provoque une erreur.

[1] …pour *New Object Notation* — `https://xkcd.com/927/`

Exercice Concevez le DSL externe décrit précédemment. Fonctionnalités attendues :

- construction programmatique d'objets NON,
- obtention d'un ensemble d'objets à partir d'une chaîne au format NON,
- accès aux objets et champs par identifiant,
- itération sur les objets et champs,
- union d'ensemble d'objets NON,
- sérialisation vers la syntaxe NON, à l'identique (modulo l'ordre des objets et champs),
- sérialisation vers la syntaxe NON aplatie (tous les champs résolus),
- sérialisation vers JSON, YAML…
- illustration et détection des problèmes sémantiques (définitions cycliques…).

Pour l'analyse syntaxique vous pouvez adopter PetitParser, une librairie de combinateur de parseurs.

```
NonDefs nds = Non.fromString(…);

Non a = nds.at("alice");
a.id();                       // alice
a.get("name");                // alice
a.get("login");               // alice
a.get("mail");                // alice.etu@exemple.tld

Non b = nds.at("bob");
b.get("login");               // bob
b.get("name");                // robert
b.get("mail");                // bob.etu@exemple.tld

nds.at("univ").get("login");  // NonFieldException
nds.at("foo");                // NonObjectException
```