Logger - funkcionalita

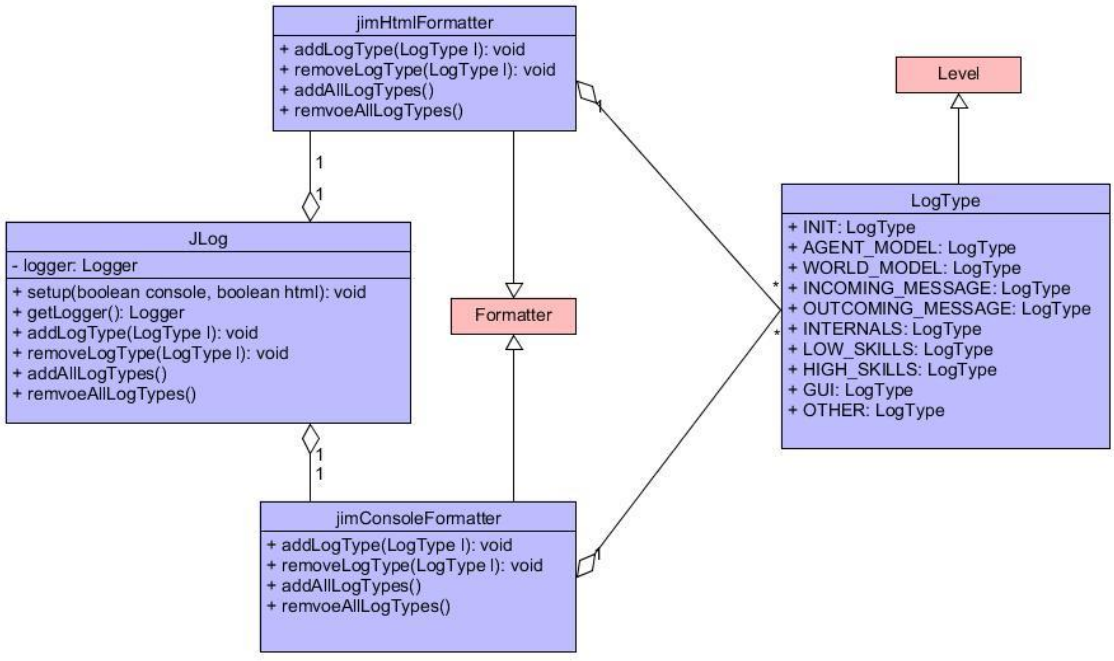
Keďže logovanie v projekte Jim nie je jednotné, rozhodli sme sa ho upraviť a vytvoriť nový logger, ktorý by sa používal v novom kóde, ktorý bude do projektu doplnený. Rovnako budeme refaktorovať i pôvodný kód a prepisovať logovanie na nový model logovania.

Analýza

V projekte Jim sa používa niekoľko metód logovania. Tím Gitmen používal na logovanie štandardný logger z Java API. Na mnohých miestach sa nachádzajú výpisy priamo na konzolu pomocou System.out.println(), čo je nežiaduce. V projekte sa nachádza i logger logujúci dodefinované udalosti v balíku sk.fiit.jim.log. Cieľom je zjednotiť rôzne metódy logovania do jednej.

Návrh

Jednotný Logger bude používať štandardný logger z JavaAPI. Logger definovaný v sk.fiit.jim.log bude odstránený, no dodatočné logovacie typy budú zakomponované do jednotného loggera, ktorý nakoniec nahradí balík sk.fiit.jim.log. Tieto typy bude predstavovať trieda LogType. Logovať bude možné do HTML súboru alebo na konzolu. Postrájú sa o to triedy JimConsoleFormatter a JimHtmFormatter odvodené od triedy Formatter, ktorá formátuje výstup loggera.



Implementácia

Logger používa ako základ štandardný logger z JavaAPI. Ten je obalený triedou JLog. Tu možno pomocou statickej metódy získať logger volaním getLogger(), no predtým musí byť konfigurovaný volaním JLog.setup(). Logger môže logovať aj na konzolu, aj do HTML tabuľky.

Na formátovanie výstupu do konzoly a do HTML súboru sa používajú triedy JimConsoleFormatter a JimHtmlFormatter, ktoré sú odvodené od triedy Formatter slúžiacej na formátovanie výstupu loggera. Výstup v HTML vyzerá nasledovne:

Tue Oct 21 11:11:46 CEST 2014

Loglevel	Time	File	Log Message
FINEST	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
FINER	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
FINE	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
INFO	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
WARNING	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
SEVERE	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
INCOMING_MESSAGE	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
GUI	okt 21,2014 11:11	sk.fiit.jim.log.LogTestMain : main()	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Logger je rozšírený o nové typy v súbore LogType. Konkrétne ide o tieto typy:

```
LogType.INIT
LogType.AGENT_MODEL
LogType.WORLD_MODEL
LogType.INCOMING_MESSAGE
LogType.OUTCOMING_MESSAGE
LogType.INTERNALS
LogType.LOW_SKILL
LogType.HIGH_SKILL
LogType.GUI
LogType.OTHER
```

Nové typy možno ľubovoľne pridávať do triedy LogType. Triedy JimConsoleFormatter a JimHtmlFormatter majú zoznam týchto typov, ktoré sa logujú navyše oproti štandardným logovacím levelom. V prípade, že chceme logovať aj tieto typy, je nutné použiť metódy JLog.addLogType(LogType l), ktorá pridá daný level do zoznamu. Všetky levely definované v LogType možno pridať pomocou JLog.addAllLogTypes(), pričom sa netreba starať dopĺňaním kódu inde ako pridaním nového typu do LogTypes, nakoľko metóda využíva reflexiu z JavaAPI:

```
public void addAllLogTypes() {
    Field[] f =  LogType.class.getFields();
    for(int i = 0; i < f.length; i++)
    {
        Object o = new Object();
        try {
            o = f[i].get(o);
        } catch (IllegalArgumentException | IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        if (o instanceof LogType) {
            levels.add((LogType) o);
        }
    }
}
```

Testovanie

Keďže logger používa štandardný logger z JavaAPI, testovanie nebolo rozsiahle. Boli manuálne overené rôzne konfigurácie loggera a jeho výstupy boli porovnané s očakávaným výstupom. Počas testovania sme chybu neodhalili.

Používateľská dokumentácia

V metóde, kde sa loguje treba získať logger pomocou `JLog.getLogger()`. Ideálne je vytvoriť členskú premennú pre celú triedu, ktorá bude logger obsahovať:

```
Private Logger LOG = JLog.getLogger();
```

Loguje sa potom pomocou štandardného API triedy Logger [5]. Navyše, oproti úrovniam v triede `Level` sú definované tieto typy logov:

```
LogType.INIT
LogType.AGENT_MODEL
LogType.WORLD_MODEL
LogType.INCOMING_MESSAGE
LogType.OUTCOMING_MESSAGE
LogType.INTERNALS
LogType.LOW_SKILL
LogType.HIGH_SKILL
LogType.GUI
LogType.OTHER
```

Tieto typy umožňujú logovať špeciálne udalosti. Tieto udalosti určuje názov typu. Napríklad v prípade použitia `LogType.LOW_SKILL` môžeme zrozumiteľne logovať rozhodnutia na úrovni low skillov. Do triedy `LogType` možno pridať aj vlastné typy logov, ak je to potrebné, pričom treba sledovať logiku pridávania týchto typov v `LogType`.

Logger sa konfiguruje vo funkcii `main()`. Pred jeho prvým spustením treba zavolať metódu `JLog.setup()`, ktorá prijíma 2 boolean hodnoty. Prvá určuje, či sa bude logovať na konzolu, druhá, či sa bude logovať do súboru. Súbor sa nachádza

v koreňovom adresári projektu a má názov *jim_logs.html*. Logy sú uložené v prehľadnej *HTML* tabuľke.

Bez ďalších nastavení sa logger obalený triedou `JLog` správa ako štandardný `Logger` z Java API. Ak chceme logovať aj dodatočné typy logov menované vyššie, musíme v main() tieto typy pridať pomocou metódy `addLogType()`:

```
JLog.addLogType(LogType.INCOMING_MESSAGE);
```

Ak chceme pridať všetky typy, ktoré definuje LogType, zavoláme metódu `addAllLogTypes()` triedy JLog. Potolm možno logovať tieto nové typy pomocou metódy log triedy Logger:

```
LOG.log(LogType.INCOMING_MESSAGE, "asdfasdf");
```

Logy by mali byť zrozumiteľné. Každý zápis do logu má mať opodstatnenie a má byť zmysluplný a formulovaný tak, aby sa dalo pochopiť, čo sa udialo a prečo. Logovať treba len pomocou tu opísaného API. Na logovanie sa nepoužívajú vlastné metódy a **hlavne nie System.out.println().** Nie je vhodné definovať ďalšie Logger-y ale treba používať ten, ktorý je nakonfigurovaný pomocou JLog.

Vhodné je obmedziť sa len na používanie metód `log(), finest(), finer(), fine(), config(), info(), warning(), severe()`.