Obsah

Zadání projektu Monolith-2-Projects

Popis problému:

Uživatel vytváří učebnici pro výuku programování . Tato učebnice v sobě obsahuje několik ukázkových úloh, které mají pomoci žákům při lepším pochopení probírané látky. Uživatel potřebuje nástroj, kterým by mohl jednotlivé úlohy vytvářet z předlohy, aniž by musel vše znovu vytvářet od začátku. Tento nástroj by měl mít speciální syntaxi, díky které se tyto změny dají snadno a jednoduše definovat.

// UPDATE 10.5.2015

Výstup musí být stejný, jako z generátoru dodaného uživatelem.

Příklad:

Uživatel vytvořil ukázkový projekt ESHOP. V první kapitole knihy je ESHOP popsán s touto adresářovou strukturou.

Kapitola 1

- ESHOP
  - WEB-INF
    - Web.xml
  - Index.jsp

V druhé kapitole knihy se uživatel rozhodl přidat další dvě podsložky **RESOURCES** a **CLASSES**, ve kterém bude nový soubor **Servlet.java**. Stejně tak se uživatel rozhodl modifikovat soubor **Index.jsp** a přidal do něj nový obsah. Výsledná struktura vypadá takto:

Kapitola 2

- ESHOP
  - WEB-INF
    - CLASSES
      - Servlet.java
    - Web.xml
  - RESOURCES
  - Index.jsp * (modified)

A v dalších kapitolách by tyto modifikace pokračovali. Uživatel bude postupně přidávat další soubory a modifikovat staré, tak aby odráželi skutečný stav projekty. Všechny tyto další modifikace si uživatel přeje dělat pomocí nástroje Monolits-2-Projects.

Řešení Monolith-2-Projects

Textové soubory:

Všechny změny, které se mají provést v textovém souboru při spuštění projektu Monolith-2-Projects budou zapsány přímo v zdrojovém kódu textového souboru. Díky tomu, lze v jednom souboru držet všechny potřebné změny a pouze určovat, které se mají aktuálně použít.

Zapsání změn v textovém souboru:

- Prostý text v souboru se vždy pouze zkopíruje, jediné změny se provádějí v blocích, kde je to specifikováno pomocí anotace.
- Označení změny se zapisuje dvěmi zpětnými lomítky a znakem zavináče
  - **//@**
- Následuje speciální keyword – „**USE_IN**" (**//@ USE_IN**) za kterým následuje popis kapitol, ve kterých se má označený text použít. Blok textu, který se má upravit je ukončen pomocí klíčového slova **„USE_END“ (//@ USE_END)**
- Pokud tento ukončovací znak chybí, jako blok se bere aktuální řádek od začátku až do konce.

Popis použití v kapitolách

- Za klíčovým slovem „**USE_IN**“ následuje popis kapitol, kde se bude text používat. *Tyto kapitoly musí korespondovat s zadanými názvy v souboru* ***chapters.json –*** *který je popsán níže.*
- Kapitoly lze specifikovat jednotlivě, každá kapitola musí být nakonci oddělena pomocí středníku. Na pořadí kapitol zde nezáleží.
  - **//@ USE_IN Kapitola1; Kapitola78; Kapitola99; Kapitola5;**
- Kapitoly lze taktéž specifikovat v intervalu. Daný interval určuje, že se tento text bude vyskytovat od první kapitoly až po poslední kapitolu definovou v intervalu. kapitola_min:kapitola_max;
  - **//@ USE_IN Kapitola1:Kapitola5;**
    - Tento blok textu bude použit od Kapitoly1 a ve všech mezi až po Kapitolu5 (včetně).
- Intervalů lze definovat více. Každý interval musí být oddělen středníkem. Tyto intervaly se nemůžou překrývat.
  - **//@ USE_IN Kapitola1:Kapitola2; Kapitola3:Kapitola8;**
    - Od Kapitoly1 do Kapitoly2, Od Kapitoly3 do Kapitoly8

- Pokud neznáme počáteční nebo koncovou kapitolu, můžeme použít otazník, jako wildcard. Tento znak značí, maximální rozsah. **“?“**
    - **//@ USE_IN Kapitola4:?;**
        - Od Kapitoly4 až po poslední existující kapitolu
    - **//@ USE_IN ?:Kapitola5;**
        - Od první existující kapitoly, až po Kapitolu5;

Příklad syntaxe textového souboru:

```
using System;
//@ USE_IN Kapitola1:Kapitola5; Kapitola7;
using System.Collections.Generic;
//@ USE_END
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Threading;
using System.Data;

namespace Ukol6
{

    /**
     * Model for DB access
     **/
    class Model
    {

        private MainWindow window;
        private String customerId { set; get; } //@ USE_IN Kapitola3;

        /**
         * Construct model with gui and customerID
         * @param MainWindow window //@ USE_IN Kapitola2:?;
         * @param String customerID //@ USE_IN Kapitola3:?;
         **/

        //@ USE_IN Kapitola2;
        public Model(MainWindow window)
        {
            this.window = window;
        }
        //@ USE_END

        //@ USE_IN Kapitola3:?;
        public Model(MainWindow window, String customerID)
        {
            this.window = window;
            this.customerId = customerID;
        }
        //@ USE_END
```

Soubory a složky

Uživatel chce mít možnost definovat změny v souborech a složkách. K tomuto zápisu slouží soubor **chapters.json**. Tento soubor obsahuje definice všech kapitol a následných změn v složkách a souborech. Z tohoto souboru se taktéž generují výsledné projekty a názvy kapitol slouží pro definice v změnách textových souborů.

Popis struktury souboru chapters.json

Tento soubor je zapsán mocí JSONu. Jedná se o pole ve kterém jsou objekty kapitol, každá kapitola má povinný identifikátor **„name“**. V každé definici kapitoly existují tyto parametry:

- **Name (string)** -> *Toto je identifikátor kapitoly. Jedná se o povinný atribut.*
- **Add (array) - >** *Toto pole obsahuje realtivní cesty k souborům a složkám, které se mají do kapitoly zahrnout. Pokud se jedná o složku, budou zahrnuty i všechny její podsložky.*
- **Exclude (array) ->** *Toto pole obsahuje relativní cesty k souborům a složkám, které se mají z kapitoly odebrat.*
- **History (boolean) ->** *Všechny kapitoly dědí předchozí strukturu složek a souborů. Pokud si uživatel nepřeje dědit strukturu, nastaví tento parameter na „false“. Defaultní hodnota je „true“.*
- ***Add-once (array)*** *→ Přidání složky pouze pro danou kapitolu. Tato složka se nepoužije v jiných kapitolách.*

Nadřazené kapitoly dědí strukturu složek, které jsou definováni předními. *(Pokud Kapitola1 již má nějaký soubor, všechny další kapitoly, které jsou v hierarchii za ní, budou mít tento soubor).*

Každá kapitola dědí strukturu kapitoly předchozí. Pokud si uživatel přeje v některé kapitole soubor vynechat, musí ho uvést do pole „**exclude“.** Tento soubor se v této kapitole odebere a v nadřazených již nebude, pokud ho opět někdo nepřidá. Hierarchii kapitol určuje pořadí, v jakém jsou definovány v souboru **chapters.json**

// UPDATE 10.5.2015

Add value

Přidána podpora pro přesunutí souboru, během přidávání do kapitoly. Díky tomu, lze do hodnoty parametru přidat lokaci výsledného souboru.

Ukázka přemístění souboru pomocí zápisu do parametru **add**:

“puvodni_lokace_souboru::nova_lokace_souboru”

Add-once

Přidána podpora pro přidání složky pouze jednou. Složka se nebude dále zábrazovat v historii.

Ukázka souboru chapters.json

```
[
   {
      "name": "Kapitola1",
      "add": [
         "WEB-INF/web.xml",
         "Index.jsp"
      ]
   },
   {
      "name": "Kapitola2",
      "add": [
         "CLASSES/Servlet.java",
         "RESOURCES"
      ]
   },
   {
      "name": "Kapitola3",
      "add": [
         "CLASSES/Main.java"
      ],
      "exclude": [
         "CLASSES/Servlet.java"
      ]

   },
   {
      "name": "Kapitola3-clean",
      "add": [
         "WEB-INF/web.xml",
         "CLASSES/Main.java",
         "Index.jsp"
      ],
      "history": "false"

   }
]
```