

Průvodce rozšířeními

NetLogo umožňuje uživatelům napsat si nové příkazy a reportéry v Javě a pak je použít ve svých modelech. Této funkcionalitě je věnována následující kapitola.

První část se zaměřuje na to, jak v modelu použít rozšíření (extensions), které jste si již napsali či někde získali.

Druhá část je určena programátorům v Javě, kteří chtějí využít API rozšíření NetLoga pro psaní vlastních rozšíření.

- Jak použít rozšíření
- Jak napsat rozšíření

Další podrobnosti najdete v podkapitole Specifikace API rozšíření NetLoga.

Jak použít rozšíření

Rozšíření použijete v modelu tak, že na začátek panelu **Procedures** přidáte klíčové slovo extensions, a to ještě před deklaraci rodů či proměnných.

Po extensions následuje seznam názvů rozšíření v hranatých závorkách, např.:

```
extensions [sound speech]
```

Příkaz `extensions` NetLogu řekne, aby našlo a otevřelo dané rozšíření a umožnilo aktuálnímu modelu používat uživatelské příkazy a reportéry, které se v rozšíření nacházejí. Tyto příkazy a reportéry tak můžete použít stejným způsobem, jako by to byla vestavěná primitiva NetLoga.

Kde se rozšíření nacházejí

NetLogo hledá rozšíření na několika místech:

1. v adresáři aktuálního modelu,
2. v adresáři `extensions` nacházejícím se na stejném místě jako aplikace NetLogo.

Každé rozšíření NetLoga sestává z adresáře se stejným názvem jako rozšíření, vše psáno malými písmeny. Tento adresář musí obsahovat soubor JAR, jehož název se shoduje s názvem adresáře.

Např. rozšíření **sound** (zvuk) je uloženo v adresáři `sound`, který má uvnitř soubor nazvaný `sound.jar`. Více informací o obsahu adresáře rozšíření najdete v podkapitole [Jak napsat rozšíření](#).

Chcete-li rozšíření NetLoga nainstalovat tak, aby ho mohl použít jakýkoliv model, vložte adresář daného rozšíření do adresáře `extensions`, nacházejícího se v adresáři NetLoga. Další možností je umístit adresář daného rozšíření do adresáře konkrétního modelu, který rozšíření používá.

Některá rozšíření potřebují další soubory. Tyto soubory se budou spolu se souborem JAR nacházet v adresáři daného rozšíření. V adresáři se mohou nacházet i další soubory, např. dokumentace a ukázkové modely.

Applety

Rozšíření mohou používat i modely uložené jako applety (pomocí **Save as Applet** v menu **File**), podmínkou je, že dané rozšíření musí být umístěno ve stejném adresáři, v jakém se nachází soubor modelu. Applety však nemohou používat rozšíření vyžadující další externí soubory JAR. (Tento nedostatek bychom rádi opravili v příští verzi NetLoga.)

Jak napsat rozšíření

Předpokládáme, že už máte zkušenosti s programováním v jazyce Java.

Shrnutí

Rozšíření NetLoga obsahuje adresář s následujícím obsahem.

Povinné:

- Soubor JAR, jehož název se shoduje s názvem rozšíření, a dále:
 - ♦ jednu či dvě třídy implementující `org.nlogo.api.Primitive`,
 - ♦ hlavní třídu implementující `org.nlogo.api.ClassManager`,
 - ♦ soubor manifest daného rozšíření NetLoga, jenž má následující čtyři tagy:
 - ◊ `Manifest-Version`, vždy 1.0,
 - ◊ `Extension-Name`, název rozšíření,
 - ◊ `Class-Manager`, plně kvalifikovaný název třídy implementující `org.nlogo.api.ClassManager`,
 - ◊ `NetLogo-Extension-API-Version`, verze API rozšíření NetLoga, pro kterou je tento soubor JAR určen. Pokud uživatel otevře rozšíření v NetLogu, které má odlišnou verzi API rozšíření, objeví se varovná hláška. Chcete-li zjistit, jakou verzi API rozšíření NetLogo podporuje, zvolte položku **About NetLogo** (O NetLogu) v menu **Help** (Nápověda) a klikněte na **System** (Systém). Nebo můžete spustit `NetLogo.jar` s argumentem `--extension-api-version`.

Volitelné:

- jeden a více modelů NetLoga demonstrujících použití rozšíření,
- jeden a více souborů JAR, které rozšíření vyžaduje,
- adresář lib s vyžadovanými nativními knihovnami,
- adresář src obsahující zdrojový kód modelu,

- dokumentace.

Při vytvoření vlastního rozšíření musíte do přístupové cesty tříd přidat soubor NetLogo.jar.

Příklady

V NetLogu se nachází několik ukázkových rozšíření s plným zdrojovým kódem v Javě. Ostatní si můžete stáhnout <http://ccl.northwestern.edu/netlogo/extensions.shtml>.

Tutorial

Zkusme si napsat rozšíření, jehož výsledkem bude jednoduchý reportér nazvaný `first-n-integers`.

Reportér `first-n-integers` dostane jako vstup celé číslo `n` a vrátí seznam celých čísel od 0 do `n - 1`. (Je to pouze příklad, tuto operaci bychom celkem snadno mohli udělat přímo v NetLogu.)

1. Vytvořte adresář rozšíření

Rozšíření je v podstatě adresář s několika soubory, takže začneme vytvořením samotného adresáře. V tomto případě ho nazveme `example` (ukázka). Veškerá naše práce bude probíhat zde. Vytvoříme rovněž podadresář `src`, který bude obsahovat zdrojový kód v Javě, a podadresář `classes` určený pro zkompilované třídy.

2. Napište primitiva

Primitiva jsou implementována jako jedna či více tříd Javy. Soubory `.java` pro tyto třídy by měly být uloženy v podadresáři `src`.

Příkaz spustí akci, reportér vrátí hodnotu. Abyste vytvořili nový příkaz či reportér, vytvořte třídu, jež implementuje rozhraní `org.nlogo.api.Command` nebo `org.nlogo.api.Reporter` rozšiřující `org.nlogo.api.Primitive`. Ve většině případů můžete rozšířit abstraktní třídu pomocí `org.nlogo.api.DefaultReporter` či `org.nlogo.api.DefaultCommand`.

`DefaultReporter` vyžaduje následující implementaci:

```
Object report (Argument args[], Context context)
    throws ExtensionException;
```

Jelikož náš reportér dostane argument, musíme rovněž implementovat:

```
Syntax getSyntax();
```

Následuje implementace našeho reportéru v souboru nazvaném `src/IntegerList.java`:

```
import org.nlogo.api.*;

public class IntegerList extends DefaultReporter
{
    // take one number as input, report a list
    public Syntax getSyntax() {
        return Syntax.reporterSyntax(
            new int[] {Syntax.TYPE_NUMBER}, Syntax.TYPE_LIST
        );
    }

    public Object report(Argument args[], Context context)
        throws ExtensionException
    {
        // create a NetLogo list for the result
        LogoList list = new LogoList();

        int n ;
        // use typesafe helper method from
        // org.nlogo.api.Argument to access argument
        try
        {
            n = args[0].getIntValue();
        }
        catch( LogoException e )
        {
            throw new ExtensionException( e.getMessage() ) ;
        }

        if (n < 0) {
            // signals a NetLogo runtime error to the modeler
            throw new ExtensionException
                ("input must be positive");
        }

        // populate the list
        // note that we use Double objects; NetLogo numbers
        // are always doubles
        for (int i = 0; i < n; i++) {
            list.add(new Double(i));
        }
        return list;
    }
}
```

Upozornění:

- Všimněte si, že číselné objekty, jež vkládáme do seznamů, jsou typu „Double“, nikoliv celá čísla. Všechna čísla použitá jako hodnoty NetLoga musejí být Double, i když nemají žádnou desetinnou část.
- K argumentům se dostanete pomocí metod třídy typesafe `org.nlogo.api.Argument`, jako např. `getDoubleValue()`.
- V případě běhové chyby volejte výjimku `org.nlogo.api.ExtensionException`, signalizující uživateli běhovou chybu Netloga.

Příkaz `Command` vytvoříme stejně jako `Reporter` až na to, že reportéry implementují `Object report (...)`, kdežto příkazy `void perform (...)`.

3. Napište třídu `ClassManager`

Každé rozšíření musí kromě určitého počtu tříd příkazů a reportérů zahrnovat i třídu, jež implementuje rozhraní `org.nlogo.api.ClassManager`. Třída `ClassManager` řekne NetLogu, která primitiva jsou součástí daného rozšíření. V jednoduchém případě rozšiřte abstraktní třídu `org.nlogo.api.DefaultClassManager` poskytující prázdné implementace těch metod z `ClassManager`, které pravděpodobně ani nebudete potřebovat.

Následuje třída `ClassManager` z našeho ukázkového rozšíření v `src/SampleExtension.java`:

```
import org.nlogo.api.*;

public class SampleExtension extends DefaultClassManager {
    public void load(PrimitiveManager primitiveManager) {
        primitiveManager.addPrimitive
            ("first-n-integers", new IntegerList());
    }
}
```

`addPrimitive ()` řekne NetLogu, zda náš reportér existuje a jaké je jeho jméno.

4. Napište soubor manifest

Rozšíření musí rovněž obsahovat manifest. Manifest je textový soubor, v kterém je uložen název rozšíření a umístění třídy `ClassManager`.

Manifest musí mít tři tagy:

- `Extension-Name`, název rozšíření;

- `Class-Manager`, plně kvalifikovaný název třídy implementující `org.nlog.api.ClassManager`;
- `NetLogo-Extension-API-Version`, verzi API rozšíření NetLoga, pro kterou je určen soubor JAR. Pokud uživatel otevře rozšíření v NetLogu, které má odlišnou verzi API rozšíření, objeví se varovná hláška. Chcete-li zjistit, jakou verzi API rozšíření NetLogo podporuje, zvolte položku About NetLogo (O NetLogu) v menu Help (Nápověda) a klikněte na System (Systém). Nebo můžete spustit `NetLogo.jar` s argumentem `--extension-api-version`.

Následuje manifest k našemu ukázkovému rozšíření, `manifest.txt`:

```
Manifest-Version: 1.0
Extension-Name: example
Class-Manager: SampleExtension
NetLogo-Extension-API-Version: 4.0
```

Řádek `NetLogo-Extension-API-Version` by se měl shodovat s aktuální verzí API rozšíření NetLoga, již používáte.

Ujistěte se, zda jsou všechny řádky, včetně poslední, zakončeny znakem nové řádky.

5. Vytvořte soubor JAR

Abyste vytvořili soubor JAR, zkompilujte jako obvykle třídy, buď z příkazového řádku, či pomocí IDE.

Důležité: Při kompilaci musíte do přístupové cesty tříd CLASSPATH přidat `NetLogo.jar` (z distribuce NetLoga).

Následuje ukázka, jak vypadá kompilace rozšíření z příkazového řádku:

```
$ mkdir -p classes      # create the classes subfolder if it does not exist
$ javac -classpath NetLogo.jar -d classes src/IntegerList.java
src/SampleExtension.java
```

Musíte také změnit argument přístupové cesty tříd CLASSPATH, aby byl určen souboru `NetLogo.jar` z instalace NetLoga. Tento příkaz zkompiluje soubor `.java` a umístí soubory `.class` do podadresáře `classes`.

Potom vytvořte soubor JAR obsahující výsledné soubory tříd a manifest. Například:

```
$ jar cvfm example.jar manifest.txt -C classes.
```

Další informace o souborech manifest, JAR a nástrojích Javy naleznete na stránce <http://java.sun.com>.

6. Použijte rozšíření v modelu

Uložte adresář `example` do adresáře Netloga `extensions` nebo do adresáře modelu, který bude rozšíření používat. V panelu **Procedures** napište na horní řádky:

```
extensions [example]
```

Nyní můžete používat `example:first-n-integers`, jako by to byl vestavěný reportér NetLoga. Napište do příkazového panelu v panelu **Interface** např. toto:

```
observer> show example:first-n-integers 5  
observer: [0 1 2 3 4]
```


Tipy pro vývoj rozšíření

Instance

Instance třídy `ClassManager` je vytvořena v okamžiku, kdy je načten model užívající dané rozšíření.

Objekty příkazů a reportérů jsou vytvořeny, kdykoliv je kompilován kód NetLoga, který tyto příkazy a reportéry užívá.

Přístupová cesta tříd CLASSPATH

Nezapomeňte, že při kompilaci musíte do přístupové cesty tříd CLASSPATH přidat `NetLogo.jar`, na což začátečníci píšící rozšíření často zapomínají. (Pokud kompilátor nenajde `NetLogo.jar`, dostanete chybovou hlášku, že třídy v balíku `org.nlogo.api` nebyly nalezeny.)

Odstraňování chyb v rozšíření

V NetLogu je několik speciálních primitiv určených pro vývoj rozšíření a odstraňování problémů. Tato primitiva jsou experimentální a mohou se později změnit. (Proto začínají dvěma podtržítky.)

- `print __dump-extensions` vytiskne informace o spuštěných rozšířeních
- `print __dump-extension-prim` vytiskne informace o spuštěných primitivech rozšíření
- `__reload-extensions` způsobí, že NetLogo při další kompilaci modelu znovu načte všechna rozšíření. Bez tohoto příkazu se změny v souboru JAR daného rozšíření nepromítnou, dokud neotevřete model nebo nerestartujete NetLogo.

Soubory JAR z dalších zdrojů

Pokud rozšíření vyžaduje kód z jiných souborů JAR, zkopírujte ostatní soubory JAR do adresáře daného rozšíření. Kdykoliv je rozšíření importováno, bude mít přístup ke všem souborům JAR v tomto adresáři.

V případě, že chcete své rozšíření poskytnout ostatním uživatelům NetLoga, dodejte s ním i instrukce k instalaci.

Podpora starších verzí Javy

NetLogo pracuje s verzí Javy 1.4.1 a vyšší. Má-li být vaše rozšíření použitelné pro všechny uživatele NetLoga, mělo by podporovat Javu 1.4.1.

Nejjednodušší je tedy pracovat rovnou s verzí 1.4.1 JDK.

Rovněž je možné používat javovský kompilátor 1.5 či 1.6, ale nesmíte zapomenout na dvě věci:

- použijte v kompilátoru `javac` (či jeho ekvivalentu v IDE) parametr `-target 1.4`, kterým novějším kompilátorům řeknete, aby použily soubory kompatibilní se staršími verzemi Javy. V kódu se tak neobjeví funkcionality určené pouze verzím 1.5 či 1.6;
- použijte v kompilátoru `javac` (či jeho ekvivalentu v IDE) parametr `-bootclasspath`, pak bude kompilace probíhat pod knihovnami tříd ve verzi Javy 1.4. (Tento krok vyžaduje instalaci JDK 1.4.) Kód tak nebude používat žádná API volání určená pouze verzím 1.5 či 1.6.

Závěr

Kompletní informace o třídách, rozhraních a metodách naleznete v části [NetLogo API Specification](#).

Upozorňujeme vás, že pro uživatele neexistuje způsob, jak získat seznam příkazů a reportérů použitelných z rozšíření, takže je důležité, abyste ke každému rozšíření poskytli adekvátní dokumentaci.

Část NetLoga umožňující rozšíření ještě není dokončena, např. API ještě neobsahuje vše, co byste asi očekávali. Jiné funkcionality sice existují, ale nejsou ještě řádně zdokumentovány. Pokud by vám nějaká chyběla, napište nám. Rovněž nás neváhejte kontaktovat s jakýmkoliv dotazy na adrese feedback@ccl.northwestern.edu, možná vám budeme schopni pomoci či vám poskytneme další informace na místech, kde není naše dokumentace dostatečná.

Tato zpětná vazba nám umožní lépe určit, jakým směrem máme vyvíjet úsilí pro další verze NetLoga, protože chceme, aby NetLogo bylo flexibilnější a umožňovalo další rozšíření.

Copyright 1999-2009 by Uri Wilensky.
Všechna práva vyhrazena.

Aplikace NetLogo, modely i dokumentace jsou šířeny veřejnosti zdarma pro účel tvorby a studia modelů. Software, modely a dokumentaci je možné pro studijní a výzkumné účely používat a měnit, a to za podmínky, že je výsledný produkt nabízen bezplatně a s uvedením informace o autorských právech a jménem původce na všech kopiích a související dokumentaci.

Pro jiné využití - než jsou výše zmíněné nekomerční způsoby - celku i jednotlivých částí (a to jak v původní, nebo změněné podobě) je třeba předem požádat o svolení od Uri Wilensky. Software, modely ani dokumentace nesmějí být užívány, přepisovány, ani upravovány jako součást komerčního softwaru nebo hardwaru bez předchozího získání licence od Uri Wilensky. Nezaručujeme kompatibilitu tohoto systému s jakýmkoliv jiným systémem a neposkytujeme žádné záruky.

Pro účely citování v akademických publikacích používejte tento odkaz:
Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo>. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.