JAVA

Nástroje v JDK

Nástroje

- javac
- javadoc
- jdb
- javah
- jconsole
- jshell
- ...

JAVA

javac

javac

- parametry
  - -cp
  - -encoding

  - -g debugovací informace
  - -g:none

  - -target verze bytecode (7, 8, 9,...)
  - --release

  - -source verze jazyka

  - -d adresář pro generovaný bytecode

  - …

JAVA

jshell

jshell

- interaktivní shell
- od Java 9

JAVA

javadoc

Přehled

- nástroj na automatické generování dokumentace ze zdrojových kódů
- deklarace tříd atd. plus dokumentační komentáře
  - dokumentace je přímo v kódu
  - snadno se udržuje aktualizovaná
- výstup – (implicitně) HTML stránky
- dokumentační komentáře

  `/** komentar */`
  - před popisovaný element
  - uvnitř – text + speciální tagy + html kód
- program `javadoc`
  - ve standardní distribuci
  - vygeneruje dokumentaci

Komentáře

- dokumentační komentáře těsně před popisovaný element

```
/** Komentar ke tride */
public class MyClass {
  /** Komentar k atributu */
  public int a;
  /** Komentar k metode */
  public void foo() {
    ...
  }
}
```

Komentáře

- jinde jsou ignorovány (jako normální komentáře)

```
/** ignorovano */
import java.util.*;

public class MyClass {
   void foo() {
     /** ignorovano */
   }
}
```

Víceřádkové komentáře

- komentáře typicky přes více řádků
- úvodní mezery a hvězdička na dalších řádcích jsou ignorovány
- bez hvězdičky mezery nejsou ignorovány (od 1.3)

```
/** Tohle je viceradkovy komentar.
  *  Uvodni mezery a hvezdicky
  *  jsou ignorovany a odstraneny.
  */

/** Zde uvodni mezery ignorovany
    nejsou - chybi hvezdicka.
  */
```

Sekce komentářů

- dvě sekce v dokumentačních komentářích
  - hlavní popis
  - sekce s tagy
- nejdřív je hlavní popis, pak sekce s tagy
  - pořadí sekcí nelze prohodit
  - sekce s tagy začíná prvním tagem (`@neco`)

```
/** Hlavni popis elementu. Stale hlavni
  * popis elementu
  * @see java.lang.Object
  */
```

- komentáře můžou mít pouze jednu sekci

Druhy tagů

- "block tags"
  - `@tag`
  - samostatné tagy
  - můžou stát jen na začátku řádku (úvodní mezery a hvězdička ignorovány)
    - znak @ jinde než na začátku řádku je normální znak
- "in-line tags"
  - `{@tag}`
  - můžou stát kdekoliv v textu
  - i hlavním popisu

```
@deprecated  As of JDK 1.1,
   replaced by {@link #setBounds(int,int,int,int)}
```

Komentáře

- první věta komentáře = shrnutí
  - věta končí první tečkou následovanou bílým znakem (nebo prvním tagem)
  - zobrazena
    - v přehledu členů třídy (metody, atributy)
    - v krátkém popisu třídy

- jeden komentář k více atributům

```
/** Komentar patri k obema atributum */
public int x, y;
```

HTML

- text komentářů – HTML
- lze používat HTML tagy

```
/** Tohle je <b>dokumentacni</b>
 * komentar
 */
```

- znaky `< > &` zapisovat jako v HTML
  - `<` ... `<`
  - `>` ... `>`
  - `&` ... `&`
- není vhodné používat některé tagy
  - např. hlavičky `<h1> <h2>`
  - naruší strukturu vygenerované dokumentace

Dědění komentářů

- pokud není komentář definován, dědí se od nadřazené třídy
  - u předefinovaných metod
  - u implementovaných metod z interfaců
- dědí se ta část komentáře, která není definována
  - platí od 1.4
  - do 1.3 – přítomnost dokumentačního komentáře zamezila dědění z předků
- explicitní dědění `{@inheritDoc}`

Popis balíku

- dokumentační komentář k balíku
- soubor package.html
- ve stejném adresáři jako třídy
- obsahuje HTML stránku
- do dokumentace se vloží vše mezi <body> a </body>
- píše se bez /** ... */
- první věta – krátký popis balíku

- popis skupiny tříd
- soubor overview.html
- jako package.html

Tagy

Tag	od JDK	Tag	od JDK
@author	1.0	@return	1.0
@{code}	1.5	@see	1.0
@{docRoot}	1.3	@serial	1.2
@deprecated	1.0	@serialData	1.2
@exception	1.0	@serialField	1.2
{@inheritDoc}	1.4	@since	1.1
{@link}	1.2	@throws	1.2
{@linkplain}	1.4	{@value}	1.4
{@literal}	1.5	@version	1.0
@param	1.0		

Tagy u metod

```
/** Hlavni popis.
   *  @param p1 popis parametru p1
   *  @param p2 popis parametru p2
   *  @throws IOException kdy je vyhozena
   *  @throws MyException kdy je vyhozena
   *  @returns  co vraci
   */
int foo(int p1, long p2) throws
    IOException, MyException;
```

Další tagy

- `@since text`
  - lze použít všude
  - význam: od jaké verze softwaru daný element existuje
  - `@since 1.4`

- `@exception`
  - to samé jako `@throws`

- `@author jmeno`
  - jméno tvůrce
  - použití u třídy, balíku a v přehledu

Další tagy

- `@see reference`
  - "See also" hlavička ve vygenerované dokumentaci
  - tři formáty
  - `@see "retezec"`
    - `@see "The Java language specification"`
  - `@see <a href="URL#value">label</a>`
  - `@see package.class#member label`
    - `@see String#equals(Object) equals`
    - `@see java.io.File#exists() exists`
- `{@link package.class#member label}`
  - reference v textu
  - podobné jako @see

Další tagy

- `{@linkplain package.class#member label}`
  - stejné jako `{@link ...}`
  - použije se stejný font jako pro text
    - pro `{@link ...}` se použije jiný font
- `@deprecated text`
  - označuje API, které by se už nemělo používat (přetrvává z předchozích verzí)
  - text – vysvětlení proč
  - zpracovává ho i překladač
    - varování při použití takového API
  - od 1.5 – anotace @deprecated
- `{@docRoot}`
  - relativní cesta ke kořenovému adresáři vygenerované dokumentace

Další tagy

- `{@literal text}`
  - text se nebude nijak interpretovat
  - `{@literal a<b>c}`
    - v dokumentaci bude a<b>c
    - <b> se nebude interpretovat jako tag
- `{@code text}`
  - to samé jako `<code>{@literal text}</code>`

javadoc

- vygenerování dokumentace – javadoc
  - součást standardní distribuce
  - spouštění

```
javadoc [parametry] [baliky]
   [zdrojove_soubory]
   [-subpackages pkg1:pkg2:...]
```

Parametry pro javadoc

- `-overview cesta/soubor`
  - jiné umístění pro soubor overview.html
- `-public`
  - dokumentace bude obsahovat pouze public elementy
- `-protected`
  - dokumentace bude obsahovat public a protected elementy
  - implicitní chování
- `-package`
  - dokumentace bude obsahovat public, protected a elementy bez označení
- `-private`
  - dokumentace bude obsahovat všechny elementy

Parametry pro javadoc

- `-doclet trida`
  - doclet generuje dokumentaci
  - imlicitní doclet generuje HTML
- `-source 1.4`
  - pokud se používají assertions (pro JDK 1.4)
- `-sourcepath seznam_cest`
  - kde se mají hledat zdrojové soubory
- `-verbose  -quiet`
  - uroveň výpisů při generování
- `-locale jazyk_země_varianta`
  - musí být jako první parametr
- `-encoding kodovani`
  - kódování zdrojových souborů

Parametry pro javadoc

- `-d cesta`
  - kam vygenerovat dokumentaci
- `-version`
  - zahrnout tag `@version`
- `-author`
  - zahrnout tag `@author`
- `-windowtitle text`
- `-doctitle text`
- `-header text`
  - bude na začátku každé stránky
- `-footer text`
  - bude na konci každé stránky
- `-nodeprecated`
- `-nosince`

JAVA

ANT

Přehled

- http://ant.apache.org/
- nástroj (nejen) na překlad java programů
- obdoba **`make`**
- napsán v Javě
- rozšiřitelný
  - přidáváním dalších tříd
- vstupní soubor (buildfile)
  - (obdoba makefile u **`make`**)
  - XML
- NetBeans interně Ant používají pro překlad, spouštění,... projektů

Buildfile

- implicitní jméno `build.xml`
- obsahuje jeden `project`
- a alespoň jeden `target`

```
<?xml version="1.0" encoding="us-ascii" ?>
<project ...>
  <target ...>
    ...
  </target>
  <target ...>
    ...
  </target>
</project>
```

Project

- atributy
  - name
    - jméno projektu
  - default
    - implicitní target, který se bude provádět, pokud nebude žádný explicitně zadán
    - povinný parametr
  - basedir
    - adresář, od něhož budou všechny cesty v souboru odvozeny
- volitelný element <description>
  - popis projektu

```
<project name="Projekt" default="compile"
  basedir=".">
  <description>Dlouhy popis projektu</description>
```

Target

- posloupnost činností (task), které se mají provést
- může záviset na jiných targets
  - provede se až po těch, na kterých závisí
- atributy
  - name
    - jméno, povinné
  - depends
    - seznam targetů, na kterých závisí
  - description
    - krátký popis
  - if
    - jméno property, která musí být nastavena
  - unless
    - jméno property, která nesmí být nastavena

Target

```
<target name="compile" depends="init"
        description="Prelozi aplikaci">
   ....
</target>
```

Task

- kód, který může být vykonán
- různý počet parametrů
  - podle druhu
- vestavěné
- volitelné
- vlastní

```
<jmeno atr1="hodnota" atr2="hodnota" .../>

<javac srcdir="..." destdir="..."/>
```

Property

- jméno a hodnota
- jména – rozlišování velikosti písmen
- získání obsahu property - ${property}
- vestavěné property
  - `basedir`
  - `ant.file`
  - `ant.version`
  - `ant.project.name`
  - `ant.java.version`
  - systémové properties Javy
- vlastní property
  - `<property name="jmeno" .... />`

Příklad

```
<?xml version='1.0' encoding='us-ascii'?>
<project basedir="." default="compile" name="Project">
  <description>Project description</description>

  <property name="src" location="src"/>
  <property name="classes" location="classes"/>

  <target name="init">
    <mkdir dir="${classes}"/>
  </target>

  <target name="compile" depends="init"
  description="Compile">
    <javac debug="true" destdir="${classes}"
      srcdir="${src}" includes="**/*.java"
      classpath="${src}" />
  </target>

<!-- pokracovani -->
```

Příklad

```
  <!-- pokracovani -->

  <target name="run" depends="init,compile"
      description="Execute">
    <java fork="true" classname="Main"
       classpath="${classes}" />
  </target>

</project>
```

Spouštění

- `ant [parametry] [target [target2 ... ]]`

- parametry
  - `-projecthelp, -p`
    - napověda projektu
    - popis projektu + popis task
  - `-propertyfile <soubor>`
    - definuje property ze zadaného souboru
  - `-D<property>=<name>`
    - definice property
  - `-buildfile <soubor>`
  - `-file <soubor>`
  - `-f <soubor>`
    - buildfile

Task javac

- spouští překladač Javy
- překládá pouze ty soubory, které jsou potřeba
  - k souboru .java není .class nebo je starší
  - pozor!
    - určuje pouze podle jmen souborů
    - tj. neví o vnitřních třídách apod.
- atributy
  - `srcdir`
    - adresář s .java soubory
    - povinný
  - `destdir`
    - kam ukládat .class soubory
  - `classpath`
    - CLASSPATH

Task javac

- atributy
  - `encoding`
    - kódování
  - `source`
    - -source parametr pro javac
  - `compiler`
    - jaký překladač pro Javu se má použít
  - `fork`
    - true nebo false (implicitně false)
    - spustit překladač ve stejne JVM jako ant nebo v nové
- `srcdir, classpath` (a další) můžou být nahrazeny vnořenými elementy `<src>`, `<classpath>` (a další)

Task java

- spustí Java aplikaci
- atributy
  - `classname`
    - třída s metodou main
  - `jar`
    - jar-soubor ke spuštění
  - povinně buď classname nebo jar
  - `classpath`
  - `fork`
    - spustit aplikaci v nové JVM
- vnořené elementy
  - `<arg>`
    - parametry příkazové řádky

Task property

- nastaví property na danou hodnotu
- nelze měnit jejich hodnotu
- atributy
  - `name`
    - jméno property
  - `value`
    - hodnota property
  - `location`
    - absolutní cesta zadaného souboru
  - `file`
    - soubor, ze kterého se mají načíst property
  - `url`
    - url, ze kterého se mají načíst property

Task property

- příklady

```
<property name="src" location="src"/>
<property name="foo.dist" value="dist"/>
<property file="foo.properties"/>
<property url="http://...." />
```

Task javadoc

- vytvoří javadoc dokumentaci
- atributy
  - `sourcepath` – adresáře se zdrojáky
  - `sourcefiles` – přímo vyjmenované zdrojáky
  - `packagenames` – pro které balíčku vytvářet
  - `destdir` – do jakého adresáře generovat
  - `public, protected,package, private` – pro které elementy se má dokumentace generovat
  - `author` – zahrnout tag autor
  - `version` – zahrnou tag version
  - … mnoho dalších

Ostatní

- mnoho dalších tasků
  - `delete`
    - maže soubory/adresáře
  - `move`
    - přesun/přejmenování souborů
  - `mkdir`
    - vytoření adresáře
  - `copy`
    - kopírování
  - `echo`
    - vypíše text na standardní výstup

JAVA

Maven

Přehled

- http://maven.apache.org/
- nástroj pro správu projektů
  - „zjednodušeně“ si lze představit jako rozšíření Antu
    - ale není to rozšíření Antu
- poskytuje
  - správu závislostí
  - usnadnění „překládání“ (build) projektů
  - používání „best practices“
  - přidávání nových modulů
  - …

Používání

- vygenerování struktury projektu
  mvn archetype:generate
  -DarchetypeGroupId=org.apache.maven.archetypes
  -DgroupId=com.mycompany.app
  -DartifactId=my-app

  - archetype ~ šablona projektu

  - vygeneruje následující strukturu

Struktura projektu

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

POM – Project Object Model

- „definice“ projektu

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Životní cyklus „buildu“

- mvn „fáze“
  - vždy se provedou i předchozí fáze

1. process-resources
2. compile
3. process-test-resources
4. test-compile
5. test
6. package
7. install
8. deploy

JAVA

Gradle

Gradle

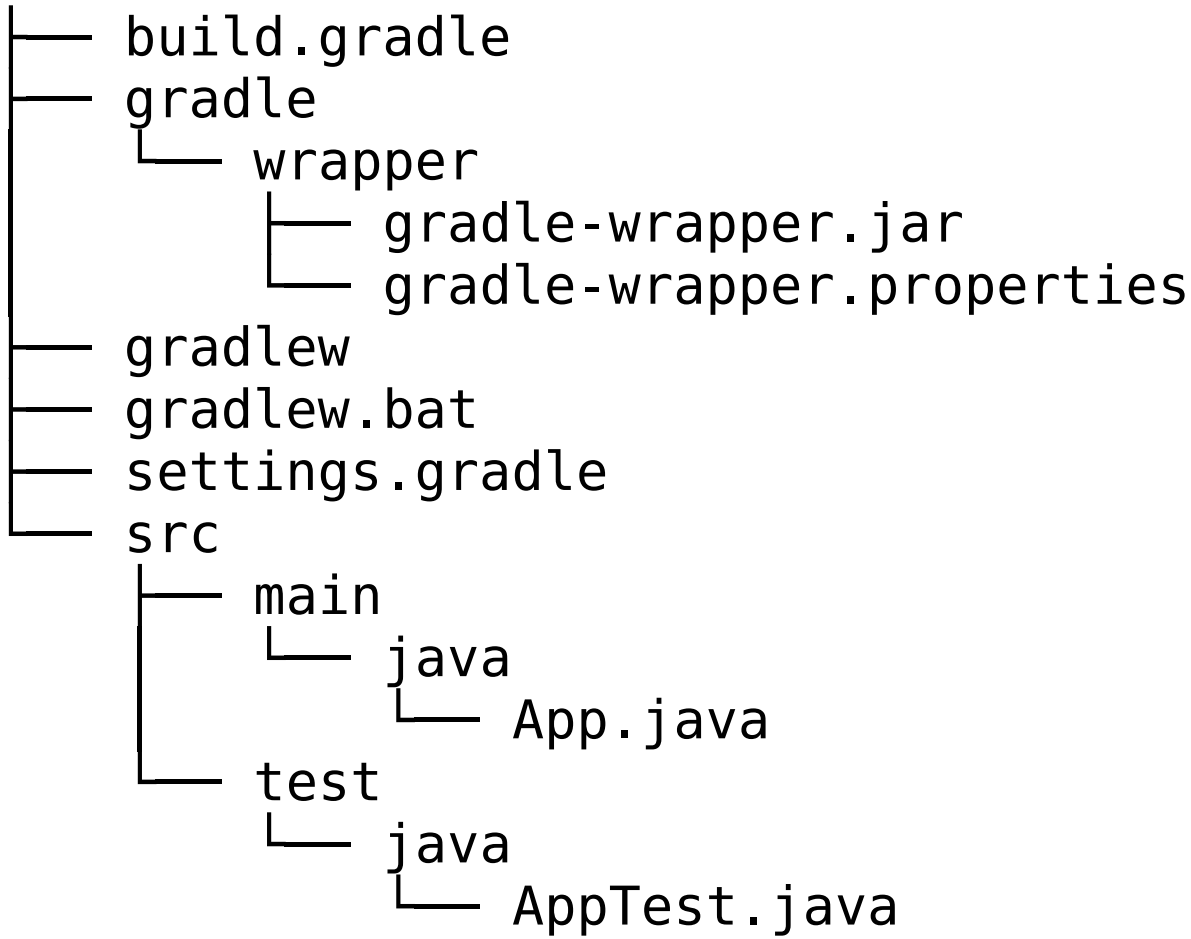
- https://gradle.org/

- podobné jako Maven
  - stejné repozitáře pro závislosti
  - ale vlastní jazyk(y) pro specifikaci projektu
    - DSL v Groovy
    - DSL v Kotlinu

- podpora pro různé jazyky/prostředí
  - Java, Android, Groovy, Scala, Kotlin, C++

Struktura projektu

- gradle init --type java-application



Gradle

- gradle build
- gradle run
- ...

- gradle tasks
  - seznam, co vše lze provádět