GOPAS
POCITACOVA.SKOLA.CZ

GOLD PARTNER
<embed/it>
deliver with passion

GENERÁLNÍ PARTNER
tieto

HLAVNÍ PARTNER
<epam>

Přechod z Java 8 na Java 11

Jirka Pinkas
Twitter: @jirkapinkas
https://github.com/jirkapinkas

Oracle JDK → OpenJDK

- V Java 11 Oracle změnil licencování Oracle JDK, které je nyní na produkci placené:
  - https://blogs.oracle.com/java-platform-group/oracle-jdk-releases-for-java-11-and-later
- Konec světa? Ani ne …
  - V dávných dobách bylo referenční implementací Java SE Sun JDK
  - Od Java 7 bylo referenční implementací OpenJDK (ale byly rozdíly v OpenJDK a Oracle JDK)
  - Od Java 11 je OpenJDK a dnešní Oracle JDK funkčně identické

Nová kadence vydávání Java verzí

- Java již od Java 9 přešla na novou kadenci vydávání verzí, které jsou navíc rozdělené na dvě skupiny:
  - Non-LTS verze: Java 9, 10, 12, …
    - vychází každých 6 měsíců
  - LTS verze: Java 8, 11, 17, 23, …
    - vychází jednou za 3 roky
  - https://www.oracle.com/technetwork/java/javase/eol-135779.html
- Oracle tvrdí, že Non-LTS verze jsou vlastně to samé jako dřívější „major release“ verze (8u20, 8u40), ale to není úplně pravda:
  - https://blogs.oracle.com/java-platform-group/update-and-faq-on-the-java-se-release-cadence
  - https://blog.joda.org/2018/10/adopt-java-12-or-stick-on-11.html

Starý vs. nový Java SE release model

Model	Old model		New model	
Upgrade	Java major releases	Java update releases	Java release train	Java patches
Frequency	Every 3 years or so	Every 6 months	Every 6 months	Every 3 months
Versions	6 -> 7 -> 8	8 -> 8u20 -> 8u40	11 -> 12 -> 13	11 -> 11.0.1 -> 11.0.2
Language changes	✅	✗	✅	✗
JVM changes	✅	✗	✅	✗
Major enhancements	✅	✗	✅	✗
Added classes/methods	✅	✗	✅	✗
Removed classes/methods	✗	✗	✅	✗
New deprecations	✅	✗	✅	✗
Internal enhancements	✅	✅	✅	✗
JDK tool changes	✅	✅	✅	✗
Bug fixes	✅	✅	✅	✅
Security patches	✅	✅	✅	✅

https://blog.joda.org/2018/10/adopt-java-12-or-stick-on-11.html

Produkce

- Základní otázka zní:
  - Dát na produkci non-LTS verzi a každé tři měsíce aktualizovat (patch) a každých šest měsíců přejít na další non-LTS verzi?
    - https://www.reddit.com/r/java/comments/9swg1z/should_you_adopt_java_12_or_stick_on_java_11/
- Nejbezpečnější je na produkci používat pouze LTS verze. A průběžně aplikaci testovat na non-LTS verzích.
  - Problém s Non-LTS verzemi je ten, že když například použijete v kódu nějakou novinku z Javy 12 a při přechodu na Java 13 se objeví nějaký blocker (je jedno jestli v Javě, ve Vašem kódu, v IDE, ve Vašem build. nástroji, v nástrojích jako SonarQube, nebo v knihovně třetí strany), tak se může lehce stát, že aplikace nějakou dobu bude muset běžet na verzi Javy, ve které nebudou nasazené poslední bezpečnostní aktualizace.

Životnost (bezpečnostní aktualizace, bugfixy)

- OpenJDK:
  - Životnost **každého** OpenJDK (non-LTS i LTS) releasu je 6 měsíců. Každé tři měsíce přijde patch.
- Oracle JDK:
  - Životnost LTS releasu je 4 roky (3 roky je životnost LTS + 1 rok čas na přechod na další LTS release).

- Poznámka k JDK 8:
  - Veřejné aktualizace přestane Oracle vydávat v lednu 2019, pro nekomerční použití budou aktualizace až do konce roku 2020. AdoptOpenJDK (viz. další snímek) plánuje buildy JDK 8 až do konce roku 2023

Další možnosti

- Zulu
  - Certifikované buildy OpenJDK od společnosti Azul
  - https://www.azul.com/downloads/zulu/
  - Nabízí LTS OpenJDK s životností 4 roky (to samé co Oracle), jenom levněji ☺
- AdoptOpenJDK
  - OpenJDK buildy
  - Pokud budou společnosti jako RedHat a IBM backportovat bugfixy (to samé co dělá Zulu), pak by se AdoptOpenJDK mohlo stát novým domovem pro LTS OpenJDK, která bude 100% zdarma.
    - https://www.reddit.com/r/java/comments/9l0mr0/adoptopenjdk_java_11_release/
    - https://adoptopenjdk.net/support.html
- RedHat?, IBM?, SAP?, Amazon? …
  - https://blog.joda.org/2018/09/time-to-look-beyond-oracles-jdk.html

Instalátor Javy

- OpenJDK produkuje pouze ZIP / TAR.GZ balíky
- Instalátor má (vše placené):
  - Oracle JDK
  - Zulu
- Osobně jsem si za posledních pár měsíců zvyknul, že Java nemusí být nainstalovaná. Ze začátku je trochu opruz všude nastavovat JAVA_HOME nebo něco podobného, ale časem si člověk zvykne a nyní mám alespoň lepší přehled o tom, pod jakou verzí Javy co spouštím a mám v tom celkově větší pořádek.
- Také tento nástroj hodně zjednodušuje instalaci Javy (a dalších nástrojů):
  - https://sdkman.io/

OpenJ9

- Společnost IBM před nějakou dobou plně opensourcovala svou implementaci JVM: J9, nyní je v nadaci Eclipse Foundation pod názvem OpenJ9.
- Chlubí se rychlostí a lepší správou paměti, což potvrzují i nezávislí uživatelé.
  - https://www.reddit.com/r/java/comments/9jt9z7/psa_try_out_the_eclipse_j9_variant_of_openjdk_for/
- AdoptOpenJDK nabízí ke stažení OpenJ9 a když nyní budete přemýšlet nad tím jaký build OpenJDK zvolit, můžete dát OpenJ9 šanci ☺

Java SE 11: The Great Removal

- Již v Java 9 zmizelo:
  - Java Visual VM (nyní lze stáhnout z GitHubu)
  - 32 bit verze JDK
- V Java 11 zmizelo:
  - Desktop JRE (viz. dále)
  - Java applety, Applet Viewer, Java Web Start, browser plug-in, Java Control Panel s automatickými aktualizacemi, javaws, javapackager
  - Java EE & Corba moduly (nyní z Centralu)
  - Java Mission Control (lze stáhnout zvlášť)
  - Java FX (nyní z Centralu)
- Brzy zmizí:
  - Nashorn (JavaScript engine)
- https://news.kynosarges.org/2018/09/26/java-se-11-the-great-removal/

JRE

- JRE už oficiálně není (alespoň pro desktop). Místo toho se má použít jlink pro vygenerování custom JRE a tím pádem výsledná aplikace není závislá na tom, aby někde byla nainstalovaná Java.
  - Bohužel výsledný jlink build není multi-platformní, tudíž se musí pro každou platformu udělat build zvlášť.
- Co servery?
  - Pokud používáte Docker, pak nemáte žádný problém:
    - https://store.docker.com/images/openjdk
  - Můžete použít JDK ☺
  - AdoptOpenJDK generuje i JRE

JAXB

- Knihovna JAXB byla v Java 11 odstraněna:
  - https://jaxenter.com/jdk-11-java-ee-modules-140674.html
- Poslední verze JAXB fungují s Java 11, stačí přidat tuto dependency:
  - https://javalibs.com/artifact/org.glassfish.jaxb/jaxb-runtime

JAX-WS

- Knihovna JAX-WS byla v Java 11 odstraněna, to se týká také konzolových nástrojů wsgen & wsimport:
  - http://openjdk.java.net/jeps/320

Přechod z Java 8 na Java 11

- Všechno aktualizujte (dependency, Maven pluginy atd.)
  - Také to můžete vzít jako záminku k tomu, abyste udělali v dependencies a pluginech pořádek ☺
  - Největší problémy způsobují starší verze knihoven, které slouží k manipulaci bytecode: javassist, cglib, asm, byte-buddy
- Není nutné rovnou začít používat Module path. Pokud budete z Classpath přecházet na Module path, tak to udělejte až úplně nakonec.
- Protože byly některé knihovny z Javy odstraněny, je zapotřebí je přibalit k projektu (obvykle se dají získat v Maven Central).
  - https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html
  - https://blog.codefx.org/java/java-11-migration-guide/
  - https://winterbe.com/posts/2018/08/29/migrate-maven-projects-to-java-11-jigsaw/

Hibernate & Java 11

- Hibernate 5.3 fungují s Java 11 + Classpath
  - V současnosti není doporučené používat Module path s Hibernate
- Jenom drobnosti:
  - Hibernate potřebují JAXB dependency ☹
  - Můžete dostat chybu z ByteBuddy, ale to má řešení
- http://in.relation.to/2018/09/13/using-hibernate-orm-with-jdk11/

ZGC

- Po G1 (od Java 9 výchozí Garbage Collector) přibyl v Javě další Garbage Collector: ZGC
  - Zatím v experimentální fázi
  - Cílem je garbage collector, který bude fungovat optimálně při různě velké alokaci paměti (až po terabyty) a GC pauzy nepřekročí 10 ms.
  - https://www.opsian.com/blog/javas-new-zgc-is-very-exciting/

Java Mission Control + Flight Recorder

- Java Mission Control (JMC) + Flight Recorder jsou plně Open Source!
  - Tyto nástroje jsou skvělé pro monitorování produkčních serverů
  - Overhead Flight Recorderu je velice nízký (cca. 1%) a sbírá informace jako: délka vykonávání jednotlivých metod, pauzy Garbage Collectoru, použití paměti, CPU atd.
  - https://springuni.com/using-java-flight-recorder-with-openjdk-11/
  - https://jdk.java.net/jmc/

Module system (jigsaw)

- V Java 11 NEMUSÍTE modularizovat aplikaci. Můžete, ale nemusíte. Stále je možné používat classpath a pro celou řadu projektů module path nemá žádný přínos.
- V Maven Central mají v současnosti explicitně definovaný module name (ať už pomocí module-info.java, nebo pomocí Automatic-Module-Name v MANIFEST.MF) pouze 2% projektů:
  - https://javalibs.com/charts/java9

var

- Od Java 10 je nové klíčové slovo: var.
  - Místo:

```
String text = "Hello Java 9";
```

  - Můžete nyní použít:

```
var text = "Hello Java 10";
```

  - Jedná se jenom o „syntactic sugar“, kompilátor nahradí „var“ příslušným typem (je to přímo vidět v class souborech).
  - Osobně nejsem velkým příznivcem této nové funkcionality, ale může to v některých situacích pomoci k čitelnějšímu kódu:
    - http://openjdk.java.net/projects/amber/LVTIstyle.html

HTTP Client

- V Java 9 byl uveřejněn nový HTTP client (experimentální), v Java 11 byl finalizován:
  - Nový HTTP client umí jak synchronní, tak asynchronní požadavky, umí pracovat jak s HTTP/1.1, tak s HTTP/2 a má hezké API:

```
var request = HttpRequest.newBuilder()
    .uri(URI.create("https://javalibs.com"))
    .build();
var client = HttpClient.newHttpClient();
client.sendAsync(request, HttpResponse.BodyHandlers.ofString())
    .thenApply(HttpResponse::body)
    .thenAccept(System.out::println);
```

  - https://winterbe.com/posts/2018/09/24/java-11-tutorial/
  - https://blog.codefx.org/java/reactive-http-2-requests-responses/

Vylepšení Collection API

- Kolekce jako List, Set a Map mají od Java 9 nové užitečné metody:
  - List.of() – vytvoří immutable list
  - List.copyOf() – vytvoří immutable kopii listu

```
var list = List.of("A", "B", "C");
var copy = List.copyOf(list);
```

Vylepšení Streams API

- V Java 9 bylo vylepšeno Stream API:
  - Stream.ofNullable() vytvoří stream z jednoho elementu:

```
Stream.ofNullable(null).count() // 0
```

  - Intermediate metody dropWhile() a takeWhile() slouží k předčasnému ukončení streamu:

```
Stream.of(1, 2, 3, 2, 1)
      .dropWhile(n -> n < 3)
      .collect(Collectors.toList()); // [3, 2, 1]
```

```
Stream.of(1, 2, 3, 2, 1)
      .takeWhile(n -> n < 3)
      .collect(Collectors.toList()); // [1, 2]
```

Vylepšení Optional

- Optional má od Java 9 a 10 několik velice užitečných metod:

```
Optional.of("foo").orElseThrow(); // foo

Optional.of("foo").stream().count(); // 1

Optional.ofNullable(null)
    .or(() -> Optional.of("fallback"))
    .get(); // fallback

Optional.ofNullable(123)
    .ifPresentOrElse(e -> System.out.println("Value: " + e),
                     () -> System.out.println("Value not present!!!"));
```

Vylepšení String

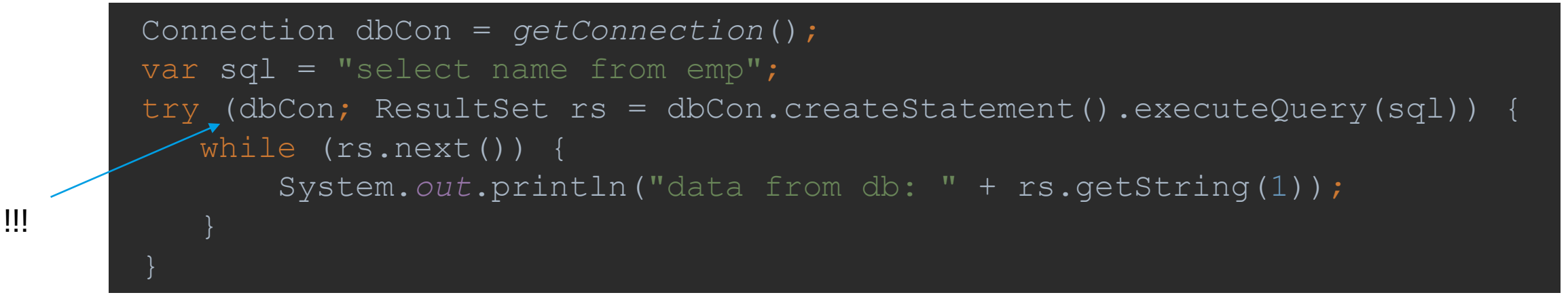
- String má od Java 11 několik nových metod:

```
" ".isBlank(); // true

" Foo Bar ".strip(); // "Foo Bar"

" Foo Bar ".stripTrailing(); // " Foo Bar"

" Foo Bar ".stripLeading(); // "Foo Bar "

"Java".repeat(3); // "JavaJavaJava"

"A\nB\nC".lines().count(); // 3
```

Metoda lines() vrací stream!

Vylepšení try-with-resources

- Try-with-resources je od Java 9 flexibilnější:



Vylepšení InputStream

- Od Java 9 je nová metoda pro transfer dat z InputStream do OutputStream:

```
var classLoader = ClassLoader.getSystemClassLoader();
var inputStream = classLoader.getResourceAsStream("myFile.txt");
var tempFile = File.createTempFile("myFileCopy", "txt");
try (var outputStream = new FileOutputStream(tempFile)) {
   inputStream.transferTo(outputStream);
}
```

Další drobnosti:
https://www.azul.com/90-new-features-and-apis-in-jdk-11/

Co nás čeká dál?

- OpenJDK bude nově na GitHubu ☺
- Blízká budoucnost je v následujících projektech OpenJDK:
  - Amber – vylepšení samotného jazyka, částečně implementované v Java 10 (klíčové slovo var), v Java 12 budou vylepšeny Stringy a switch operátor
    - https://openjdk.java.net/projects/amber/
  - Loom – vylepšení práce s vlákny
    - http://cr.openjdk.java.net/~rpressler/loom/Loom-Proposal.html
  - Panama – lepší spolupráce s C / C++
    - https://openjdk.java.net/projects/panama/
  - Valhalla – opět vylepšení samotného jazyka: možnost používat primitivní datové typy s generics
    - https://openjdk.java.net/projects/valhalla/

Děkuji za pozornost.

www.**JavaDays**.cz

www.**gopas**.cz