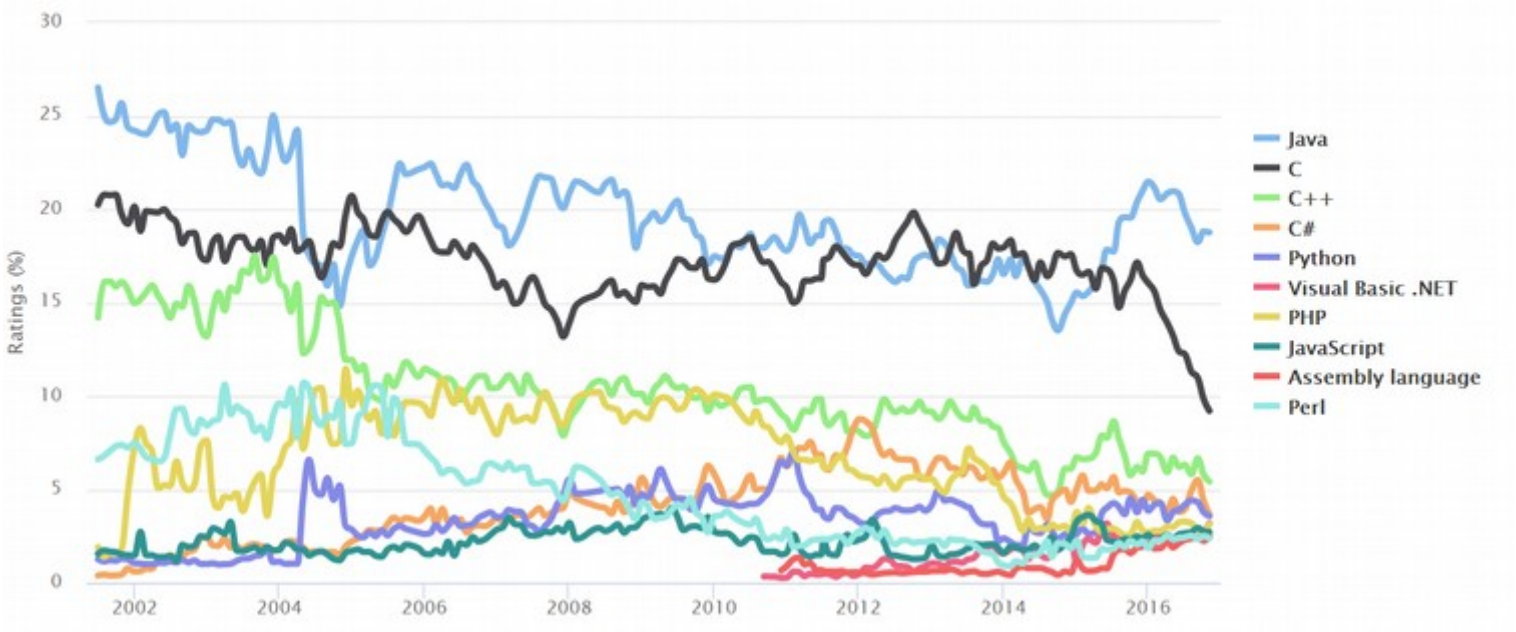
Novinky v Java světě, co je in a co out

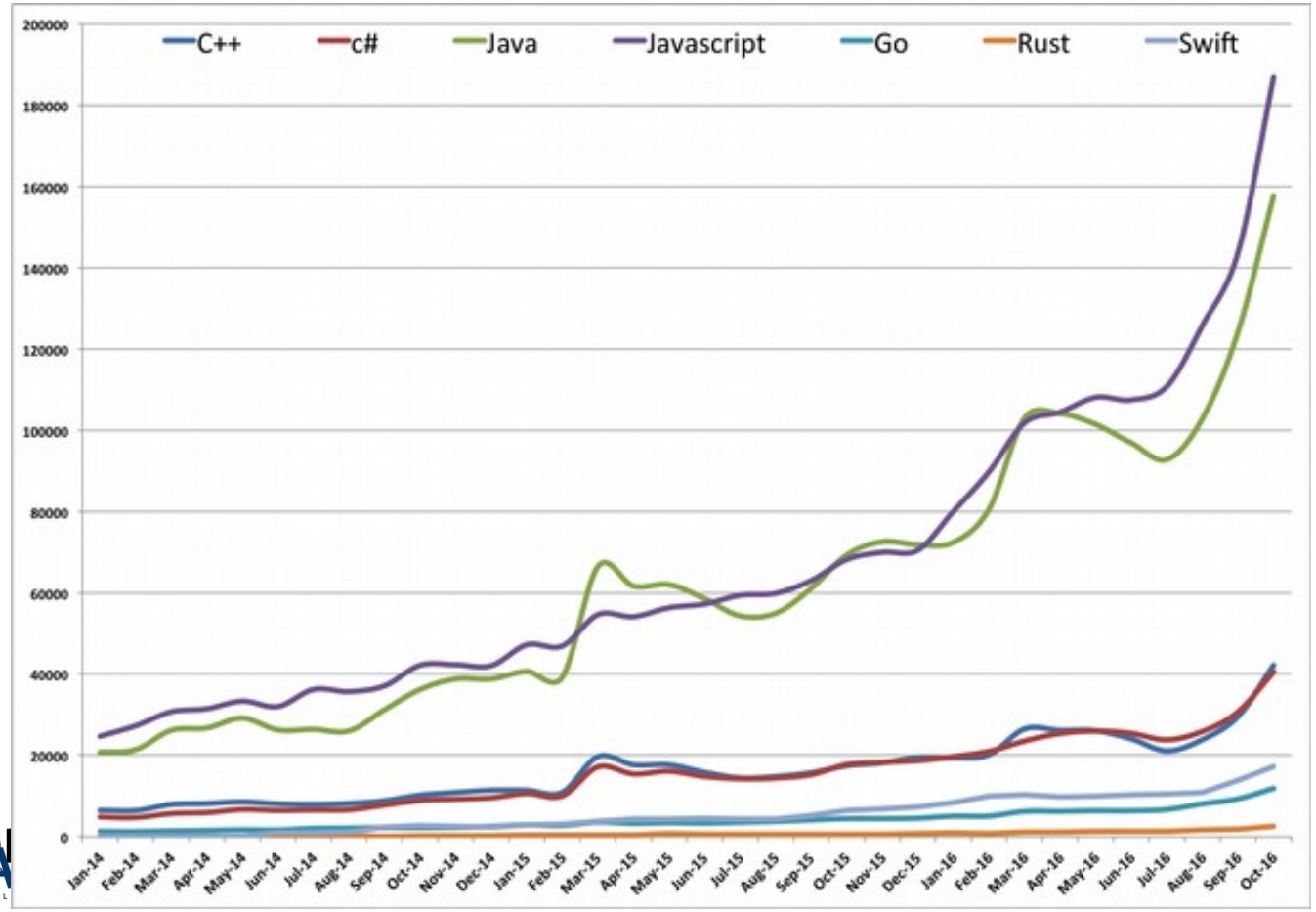
Jirka Pinkas @jirkapinkas

- Jak hodně je Java populární?
  - Podle všech měřítek HODNĚ :-)

- Tiobe index
  - http://www.tiobe.com/tiobe-index/
  - http://www.tiobe.com/tiobe-index/programming-languages-definition/



- GitHub – počet aktivních projektů per language per month
  - https://github.com/emmanuel-keller/github-language-statistics/blob/master/README.md



Podívat se na aktuální data, algoritmus se mění!

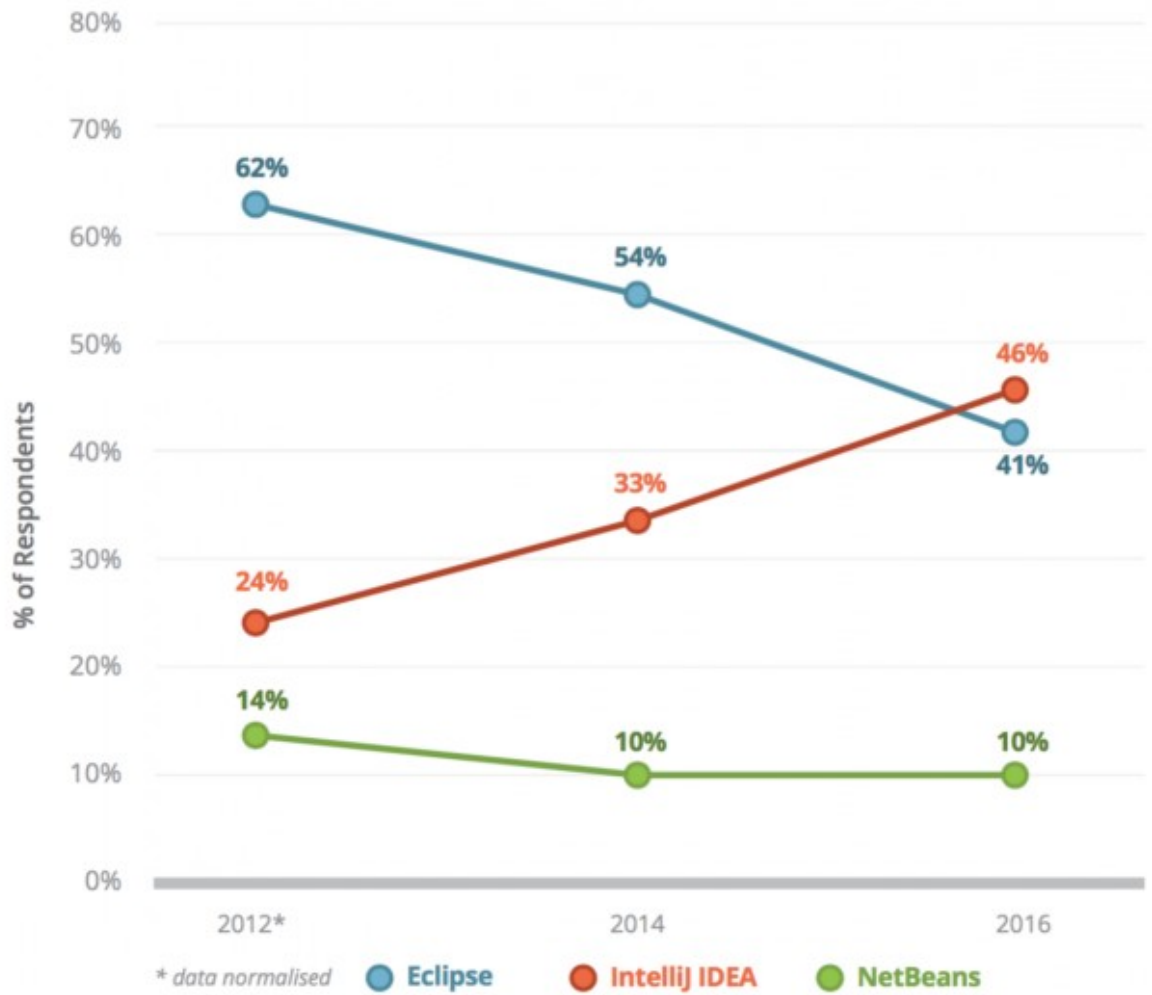
- The computer industry is the only industry that is more fashion-driven than women's fashion.
  - https://en.wikiquote.org/wiki/Larry_Ellison

Eclipse vs. IntelliJ Idea vs. NetBeans

- Velice „populární“ otázka na kterou není jednoduchá odpověď … každé vývojové prostředí má své klady a zápory, žádné není 100% nejlepší.
  - https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/
  - https://blogs.oracle.com/geertjan/entry/adding_some_color_to_the

GOPAS
POCITACOVA.SKOLA.CZ

Figure 3.3 IDE Usage Since 2012



REBELLABS by ZEROTURNAROUND

Java Days

Java Days

Reactive programming I.

- Navrhovaná součást Java EE 9:
  - https://jaxenter.com/a-glimpse-at-java-ees-future-129340.html
- Ve Spring 5 použitelné již dnes:
  - https://projectreactor.io/
  - https://spring.io/blog/2016/07/28/reactive-programming-with-spring-5-0-m1
- V Java SE pomocí RxJava:
  - https://github.com/ReactiveX/RxJava (mimochodem – toto je 2. nejpopulárnější Java projekt na GitHubu)

Reactive programming II.

- Problém: když budeme mít reaktivní prezentační a servisní vrstvu, tak to pro některé aplikace může být super, ale většina aplikací pracuje s databází a ta by taky musela být reaktivní. Pokusy již existují, ale v dohledné době se musíme hodně snažit abychom je využili:
  - http://reactivemongo.org/
  - https://www.reddit.com/r/java/comments/54janw/nonblocking_jdbc_api_slides_from_talk_at_javaone/
  - https://github.com/mauricio/postgresql-async

Java 8 – Streams, Lambdas & Optional

- Rozhodně IN a HOT, HOT, HOT i přes řadu nedodělků, které vyřeší Java 9 (až někdy vyjde … http://www.java9countdown.xyz/):
  - http://blog.codefx.org/java/dev/java-9-stream/
  - http://blog.codefx.org/java/dev/java-9-optional/

… a něco se asi bohužel už nikdy nevyřeší (Lambda & checked exceptions), což nejlépe vystihuje tento obrázek:

Lambda – tak takhle ne

```
1466      -            for (final Indexes idx : indexes) {
1467      -                if (idx.value().length > 0) {
     1464 +            indexes.stream()
     1465 +                    .filter(idx -> idx.value().length > 0)
     1466 +                    .forEach(idx -> {
1468 1467                            for (final Index index : idx.value()) {
1469 1468                                if (index.fields().length != 0) {
1470 1469                                    ensureIndex(mc, dbColl, index.fields(), index.options(), background, parentMCs, parentMFs);
1471 1470                                } else {
1472      -                             LOG.warning(format("This index on '%s' is using deprecated configuration options.  Please update t
     1471 +                                LOG.warning(() -> format("This index on '%s' is using deprecated configuration options.   P
1473 1472 *                                      + "update to use the fields value on @Index: %s", mc.getClazz().getName(), index.to
1474 1473                                  final BasicDBObject fields = parseFieldsString(index.value(), mc.getClazz(), mapper,
1475 1474                                          !index.disableValidation(), parentMCs, parentMFs);
1476 1475                                  ensureIndex(dbColl, index.name(), fields, index.unique(), index.dropDups(),
1477 1476 *                                        index.background() ? index.background(): background, index.sparse(), index.expireAf
     1477 +                                                          : background, index.sparse(), index.expireAfterSeconds());
1478 1478                              }
1479 1479                          }
1480 1480 *                    });
1481      -            }
```

Method Reference > Lambda

```
23 23           IntStream.range(nowMinute, nowMinute + 10)
24      -               .forEach((minute) -> {
25      -                   dataSeries.getData().add(new Data<>(String.valueOf(minute), 0.0));
26      -                   minuteToDataPosition.put(minute, dataSeries.getData().size() - 1);
27      -               });
    24  +               .forEach(this::initialiseBarToZero);
28 25       }
                                                                    }
43 40       }
44 41
    42  +   private void initialiseBarToZero(int minute) {
    43  +       dataSeries.getData().add(new Data<>(String.valueOf(
    44  +       minuteToDataPosition.put(minute, dataSeries.getData
    45  +   }
    46  +
```

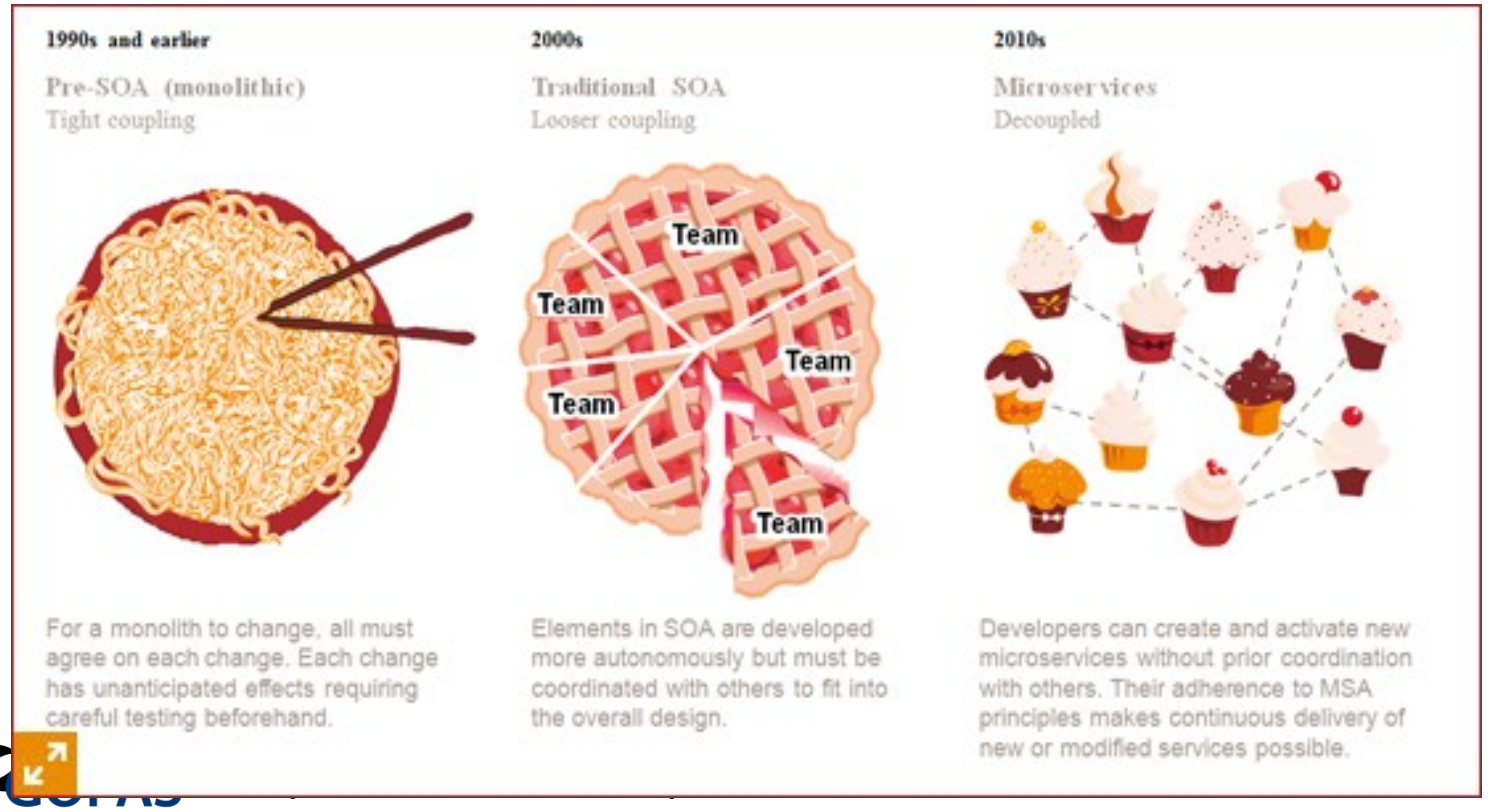
JAR vs. WAR

- Dropwizard, Spring Boot vs. Java EE server (první vlaštovky TomEE a Wildfly Swarm).
- V Java EE 8 by měly být výše uvedené způsoby tvorby uber-jar standardizované.
- A Java EE 9 možná zruší Java EE server kompletně:
  - https://jaxenter.com/a-glimpse-at-java-ees-future-129340.html

Poznámka: Proto jsem se ani nezabýval srovnáním Java EE serverů, vypadá to, že zmizí do pozadí a budou z nich knihovny.

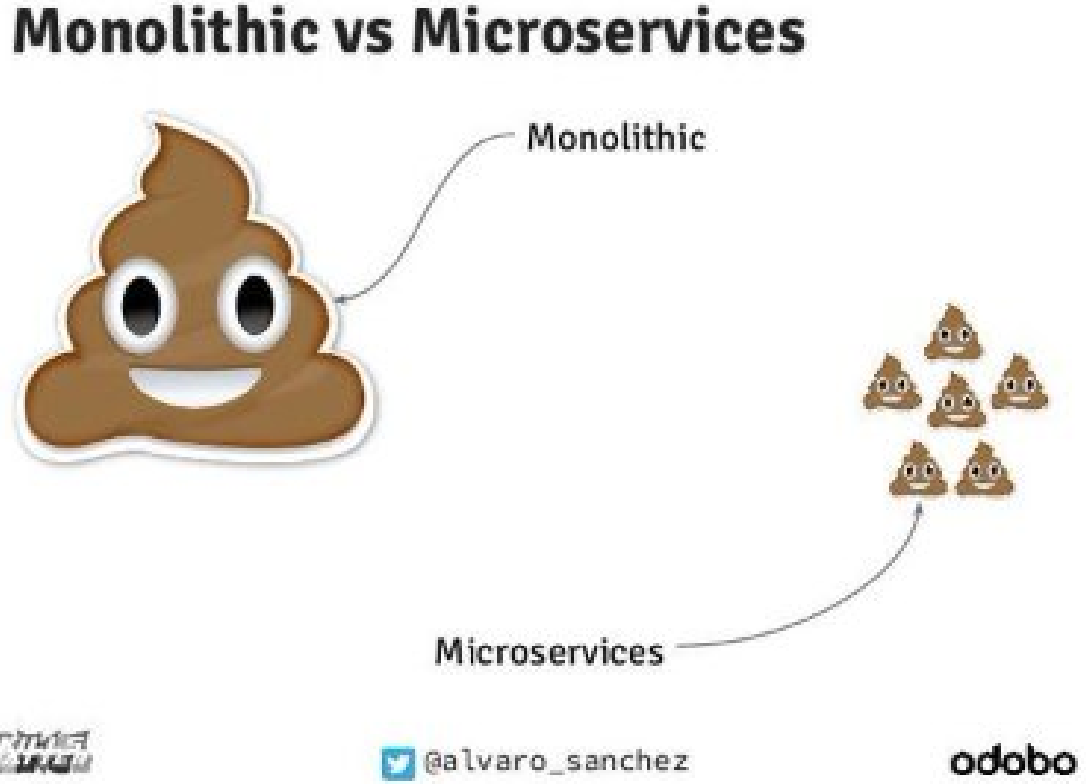
Microservices I.

- Před cca. jedním až dvěma roky se zdálo, že se všechno přepíše do architektonického stylu zvaném „Microservices“. A zdálo se, že budoucnost bude růžová:

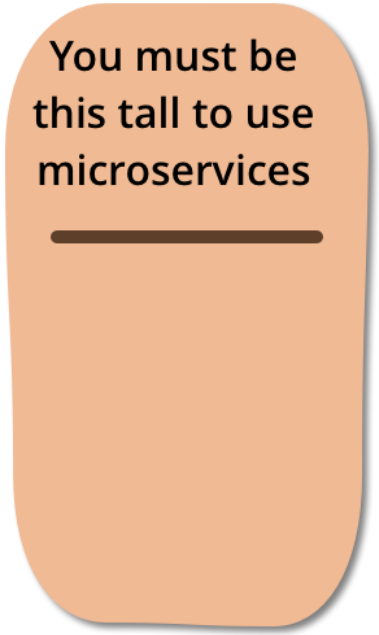


Microservices II.

- Nicméně vše je jenom o lidech a tak to také může velice lehce dopadnout takto:



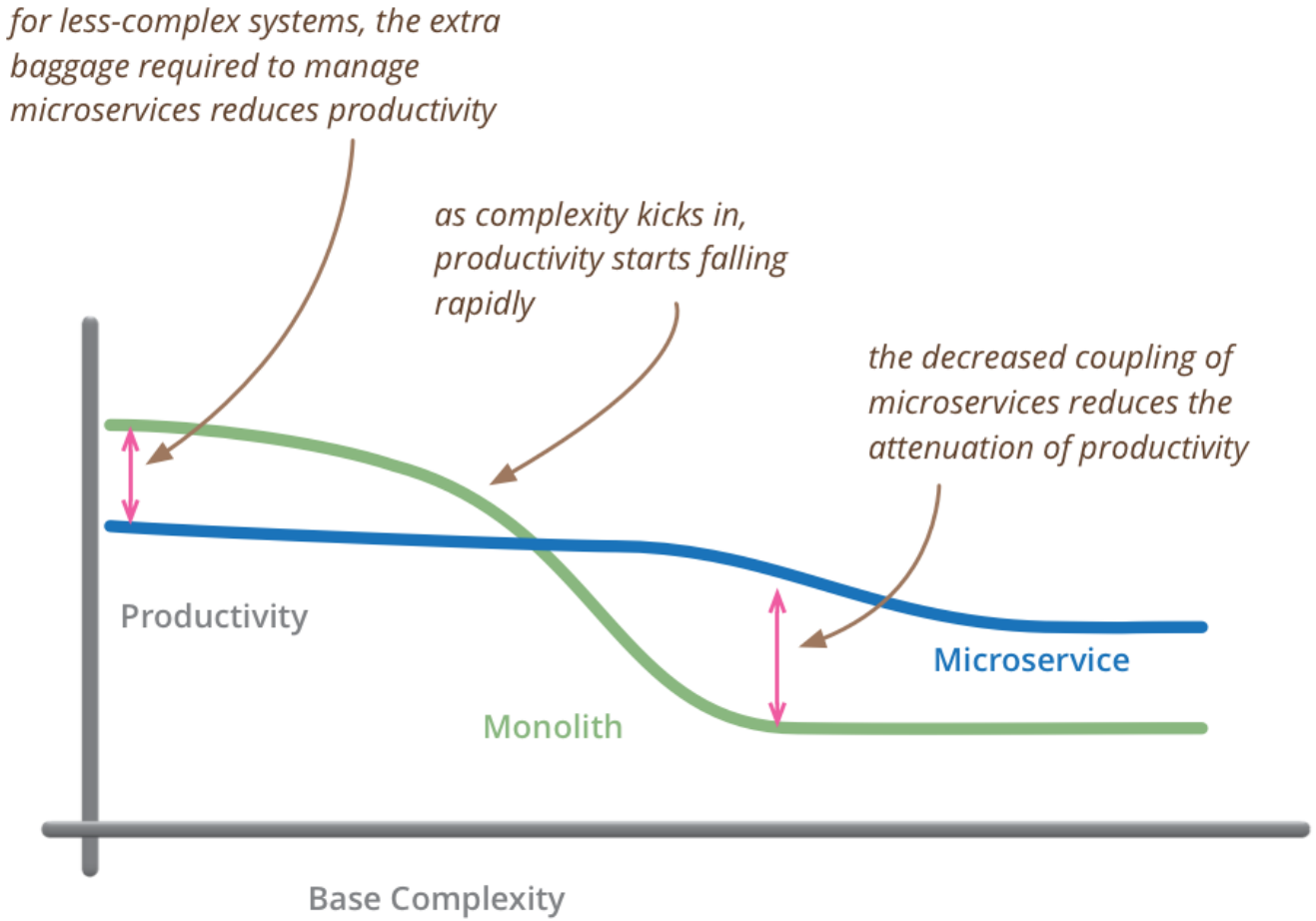
Microservices III.



Microservicy nejsou pro každého.

- Musíte vyřešit:
  - Monitorování
  - Distribuované logování
  - Mít kompletně zautomatizovaný
  - proces od vývoje po nasazení
  - Mít kompletně zautomatizované
  - vytvoření nového serveru a jeho
  - zapojení do Vaší Microservices architektury.

Microservices IV.



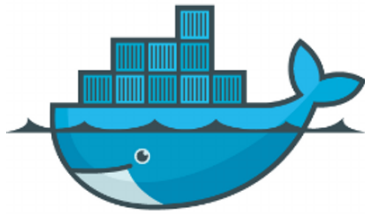
but remember the skill of the team will outweigh any monolith/microservice choice

GOPAS
POCITACOVA.SKOLA.CZ

Microservices V.

- http://martinfowler.com/bliki/MicroservicePrerequisites.html
- http://martinfowler.com/bliki/MicroservicePremium.html
- https://www.stavros.io/posts/microservices-cargo-cult/
- http://basho.com/posts/technical/microservices-please-dont/
- https://news.ycombinator.com/item?id=12508655

Docker



Při používání Microservices je nutné mít vyřešené vytváření

- prostředí, ve kterém bude Microservice běžet. K tomu je možné použít řadu přístupů. Docker (a nástroje, které jsou součástí širšího Docker ekosystému) je jednou z populárních možností.
- Jenom pozor na to, že Docker není žádné magické udělátko, které za nás udělá všechnu práci. V celé řadě situací té práce spíš přidá (zejména když ho chcete rozjet v produkci):
  - https://thehftguy.wordpress.com/2016/11/01/docker-in-production-an-history-of-failure/
  - http://racknole.com/blog/running-docker-in-production-for-6-months/

Moje vlastní poznámky I.

- Abych nebyl tolik negativistický:
  - „Microservices“ používám posledních 10 let, nový je jenom název, samotná technologie je stará snad od doby vynalezení počítačové sítě.
  - V jednom mém projektu plánuji (budoucí novoroční plán pro rok 2017), že z hromady Microservices udělám jeden Monolith. U druhého mého projektu zase plánuji jeho rozdělení do více Microservices.
  - Na Dockeru jedu několik měsíců v produkci a po počátečních porodních bolestech si ho nemohu vynachválit.
  - A všechny servery pohání Spring Boot uber-jary.

Moje vlastní poznámky II.

- A když Microservices, Docker, uber-jar a vše ostatní je nastavené dohromady a správně, pak je to nepřemožitelná kombinace:

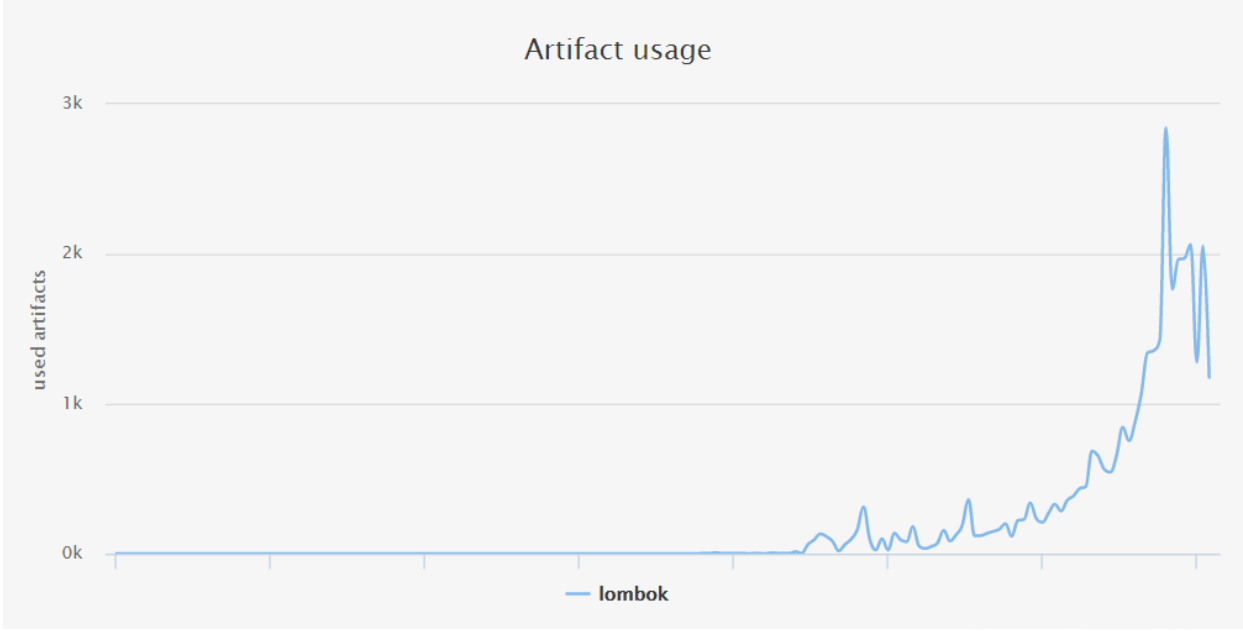
SonarQube, FindBugs, PMD, CheckStyle

- Doufám že každý používá některý z nástrojů pro hlídání kvality kódu (SonarQube, FindBugs a PMD) a CheckStyle pro vynucení standardů psaní kódu (zejména v týmu lidí je to nedocenitelné).
- Dřív byl SonarQube pouze hezký interface nad uvedenými analyzátory kódu, ale v posledních verzích jejich funkcionalitu přeprogramovali k obrazu svému:
  - http://www.sonarqube.org/sonarqube-java-analyzer-the-only-rule-engine-you-need/
- Navíc například vývoj FindBugs prakticky ustal:
  - https://www.reddit.com/r/java/comments/5bg0ye/findbugs_project_in_its_current_form_is_dead/

Lombok

- Nebaví vás psát gettery / settery a další „boilerplate code“? Od toho je tady Lombok:
  - https://projectlombok.org/
- Zejména v tomto roce statistiky jeho použití vyskočily do závratných výšin:
  - Zdroj: https://javalibs.com/custom-chart

Poznámka: Doufejme,
- že ho Java 9 nerozbije.

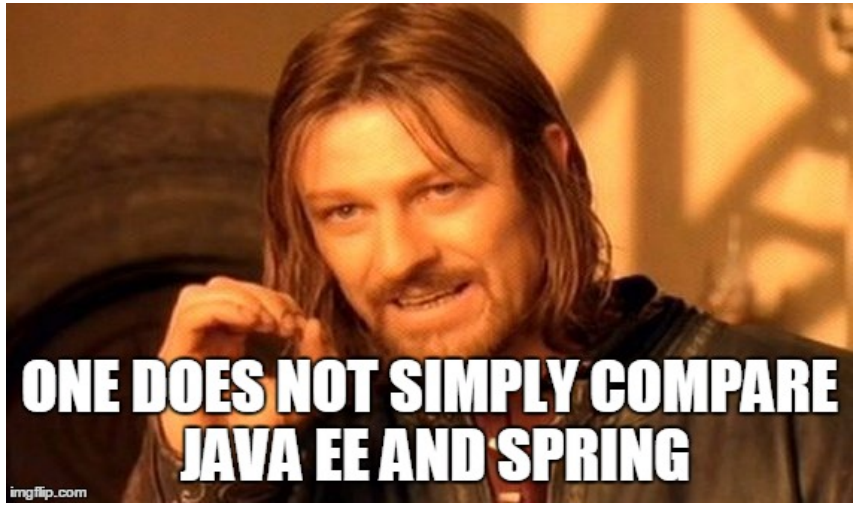


Profiling Java aplikací

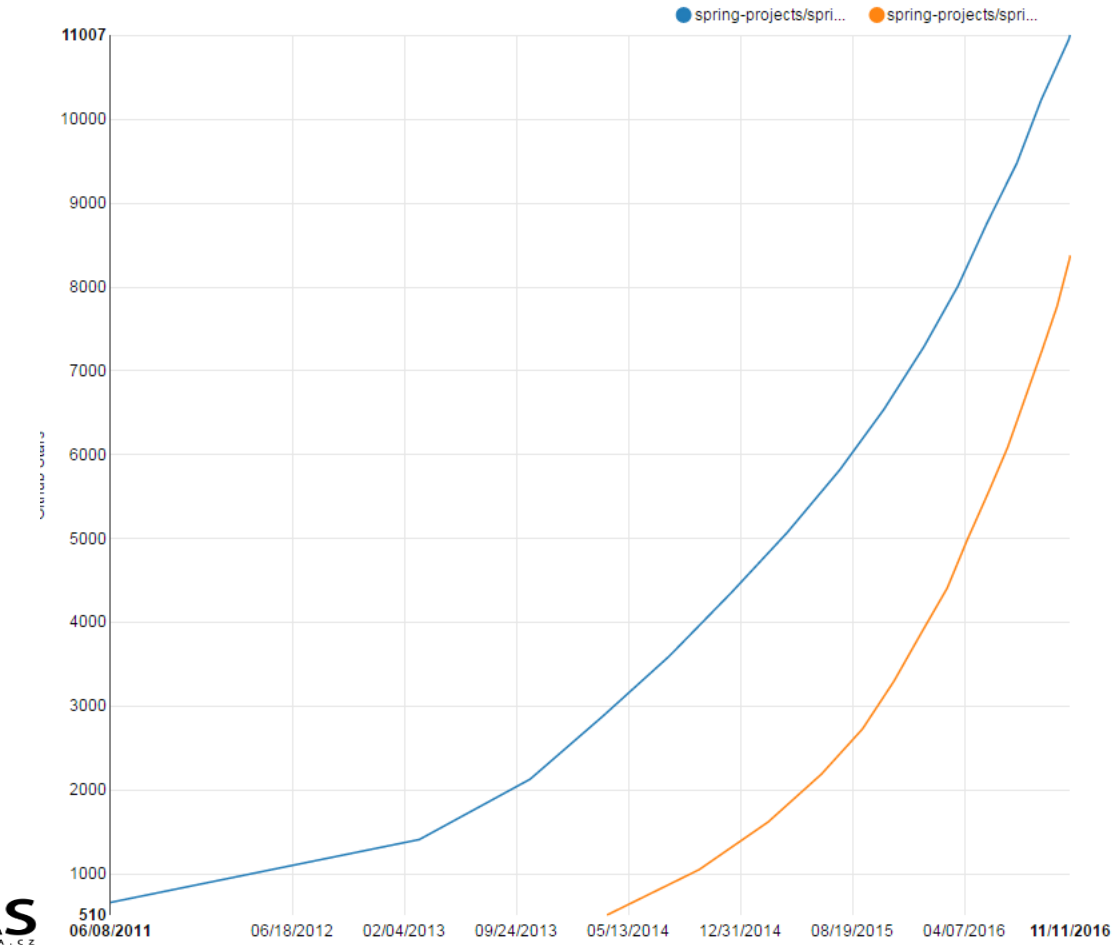
- Populární nástroje:
  - Zdarma (součást JDK):
    - JVisualVM (nově na GitHubu: https://visualvm.github.io/)
    - JMC (Java Mission Control)
      - Na produkci vyžaduje licenci
  - Placené
    - JProfiler
    - New Relic
    - YourKit

Spring vs. Java EE

- Další „populární“ otázka. Vzhledem k tomu, že Spring často kooperuje a staví nad Java EE, pak ani není moc férová.
- Nicméně díky tomu, že je Spring na GitHubu, pak se můžeme podívat jak hodně je populární tam:
  - http://www.timqian.com/star-history/

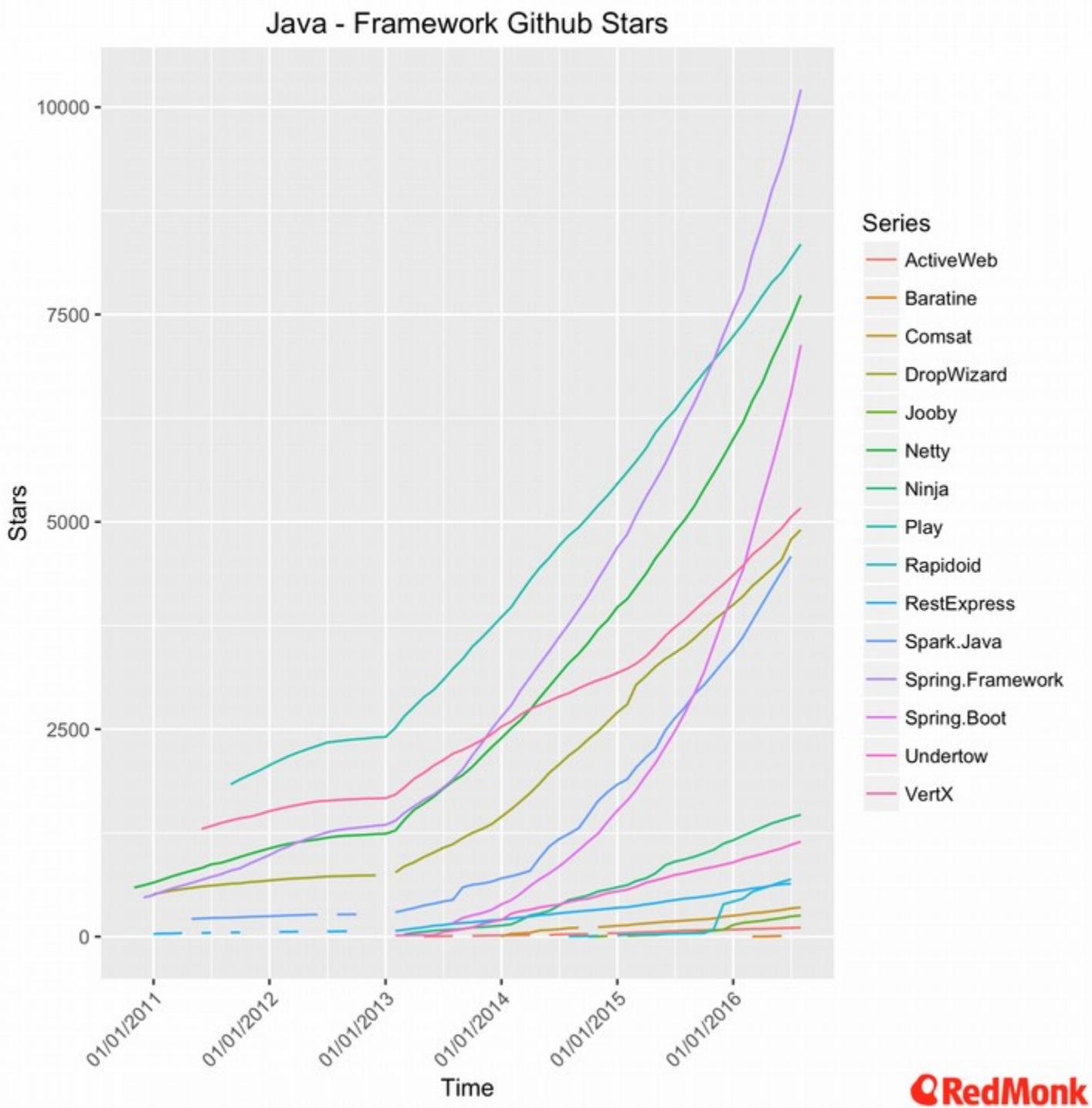


- Vývoj počtu hvězdiček v čase (kumulativní součet) pro Spring Framework a Spring Boot:

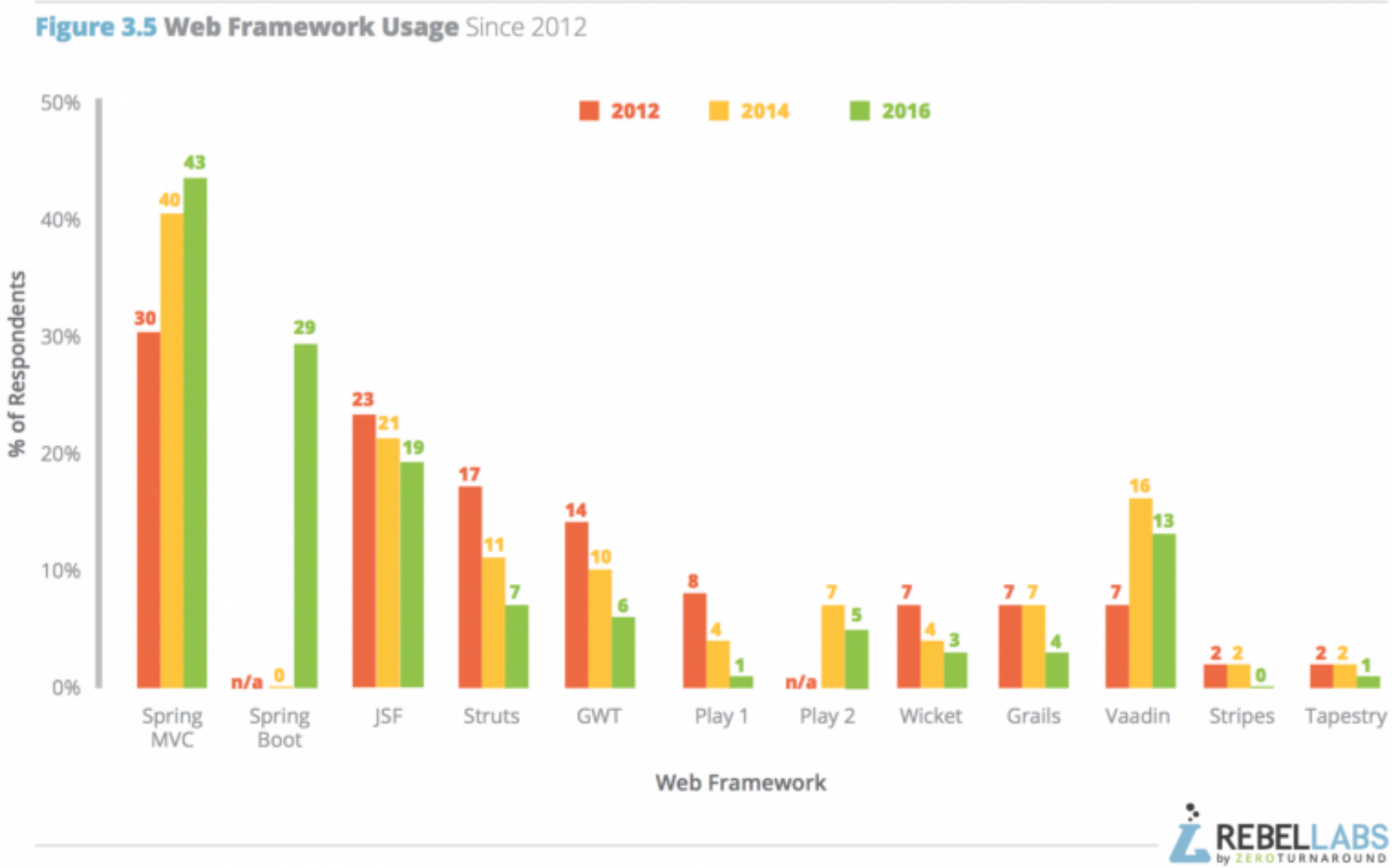


Další srovnání GitHub stars populárních Java web projektů od společnosti

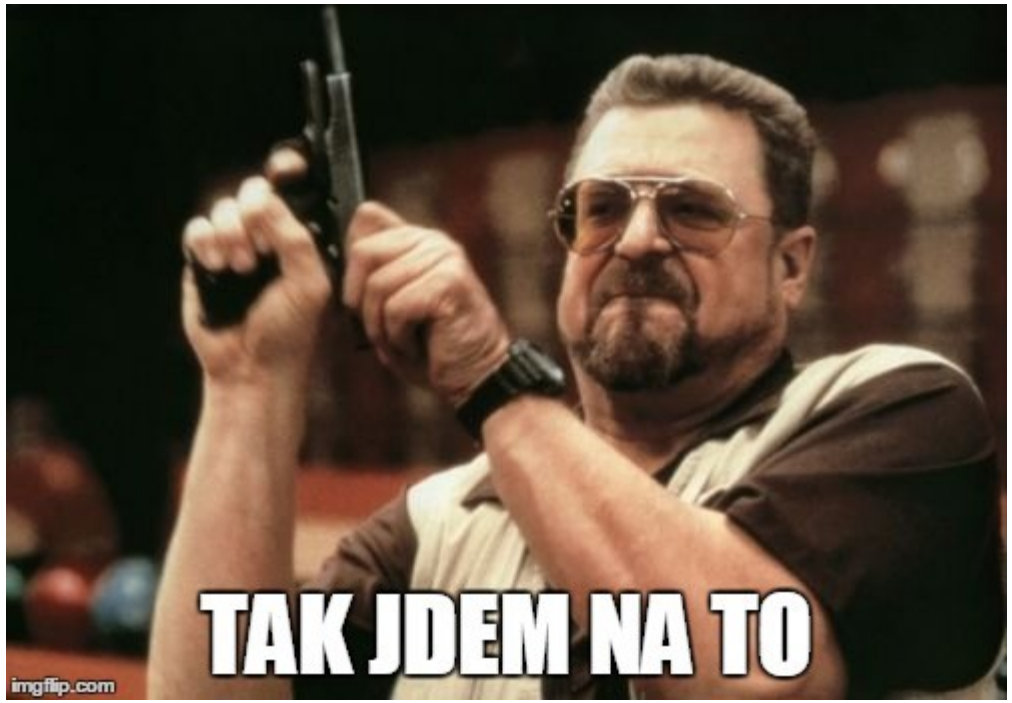
- Redmonk:
  - https://www.infoq.com/news/2016/09/redmonk-java-frameworks



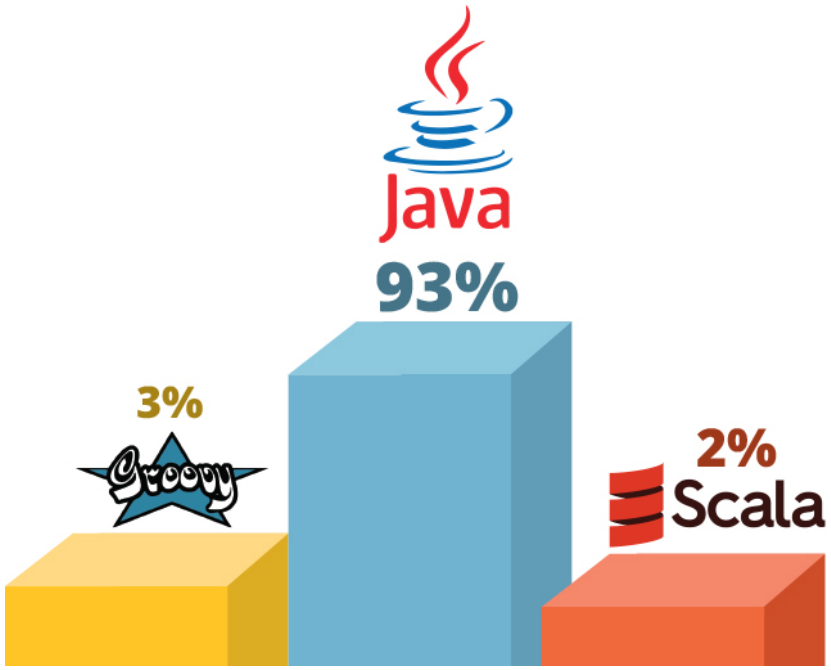
Populární web. frameworky podle RebelLabs

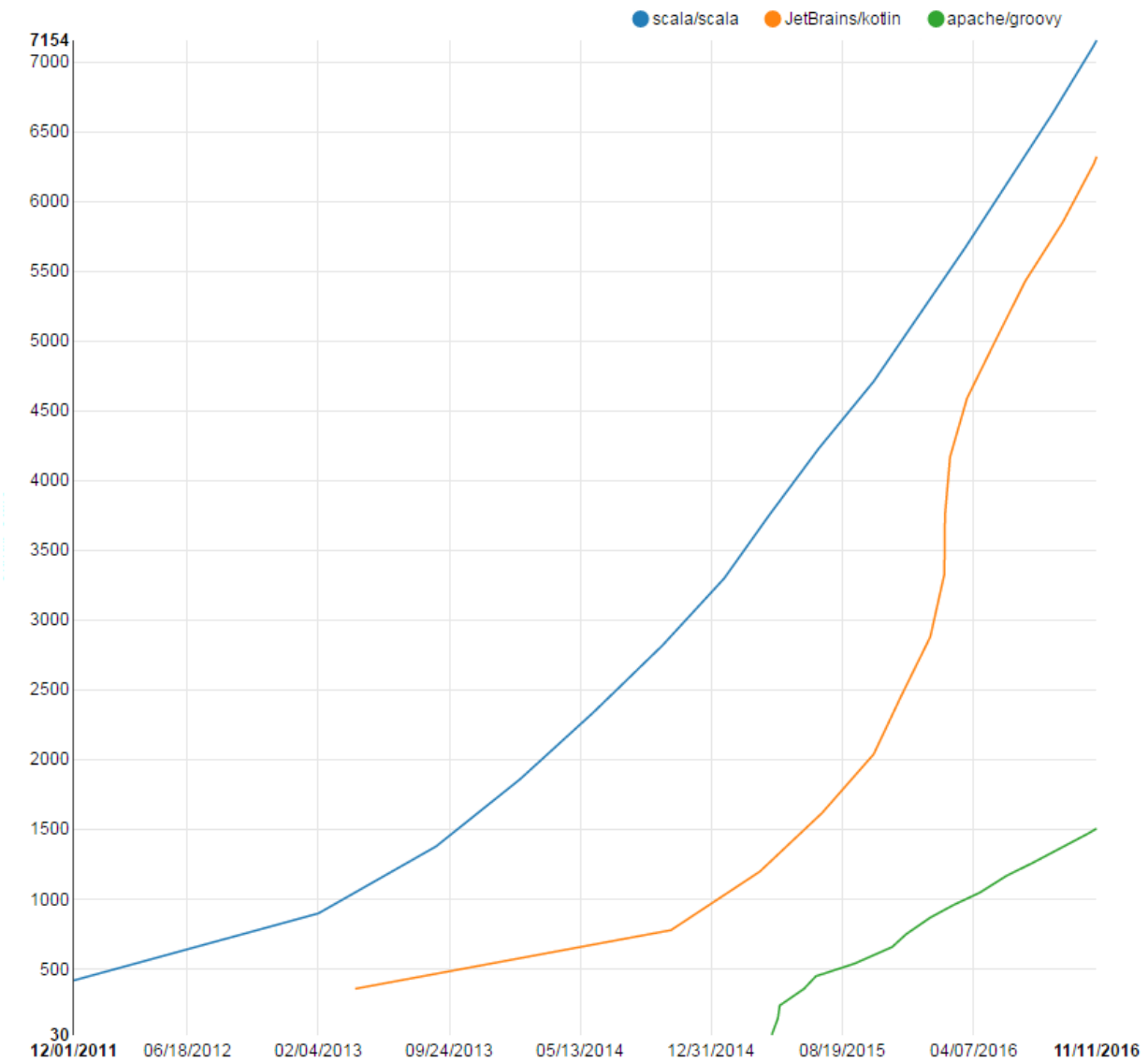


- GitHub data mining:
  - Java knihovny na GitHubu s největším počtem hvězdiček:
    - https://github.com/search?l=Java&p=1&q=stars%3A%3E1&s=stars&type=Repositories



- Java vs. Scala vs. Groovy vs. Kotlin



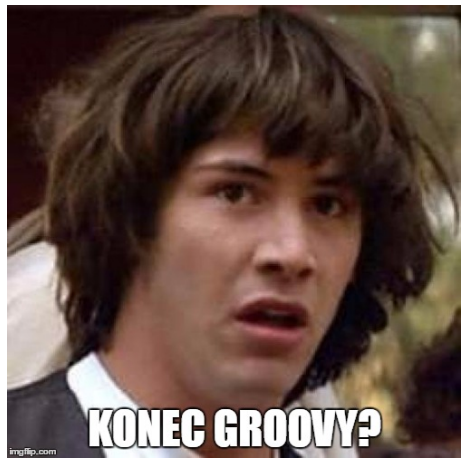


Proč Scala, Groovy, Kotlin, … ?

- Mají alternativní jazyky v JVM vůbec význam? ANO, ale nemá význam je porovnávat s Javou a myslet si, že Javu nahradí. Nenahradí.
  - https://dzone.com/articles/scala-is-dead-long-live-scala - Scala je super pro funkcionální programování zejména ve spojení s big data (Apache Spark)
  - Kotlin v současnosti našel využití v Androidu, protože umí Lambdy a Android „umí“ Java 8 až od Android 7.0. Obrovský rozdíl Kotlinu oproti Groovy a Scale je jeho prvotřídní podpora v IntelliJ Idea (JetBrains jsou tvůrci Kotlinu).

Proč Scala, Groovy, Kotlin, … ?

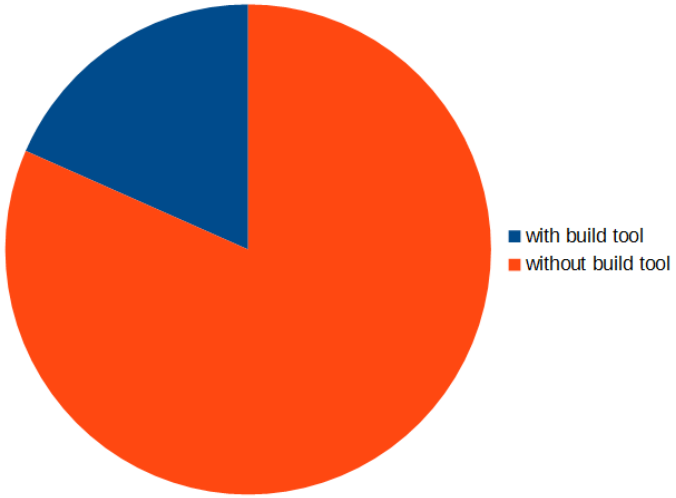
- Groovy je super pro DSL (našlo využití v Gradle, ale tam je postupně nahrazováno Kotlinem). Originální tým (glaforge, blackdrag a melix) byl před dvěma lety zlikvidován Pivotalem a Groovy už moc nevyvíjí, ale pár dalších dobrovolníků se našlo a v současnosti Groovy víceméně přežívá:
  - https://github.com/apache/groovy/graphs/contributors
  - http://melix.github.io/blog/2016/05/gradle-kotlin.html

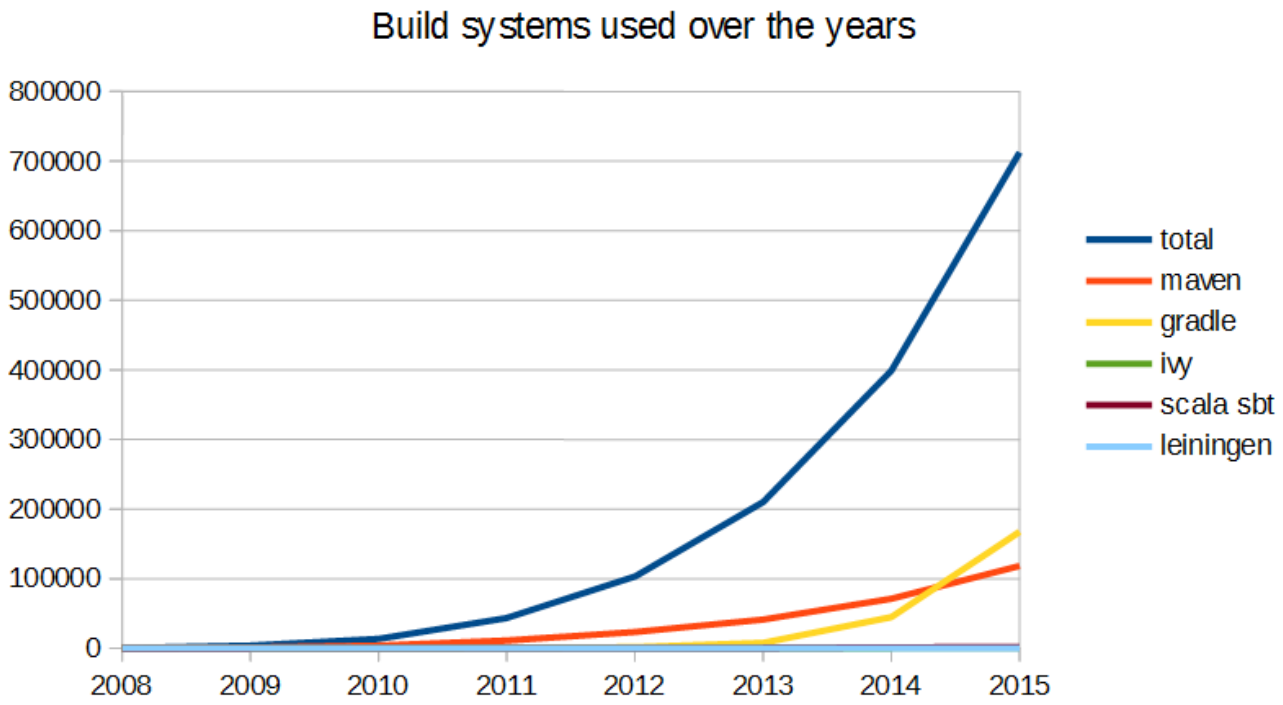


- Maven vs. Gradle
  - Gradle vítězí u Android projektů (a projektů se složitým a customizovaným build procesem).
  - Maven vítězí všude jinde.

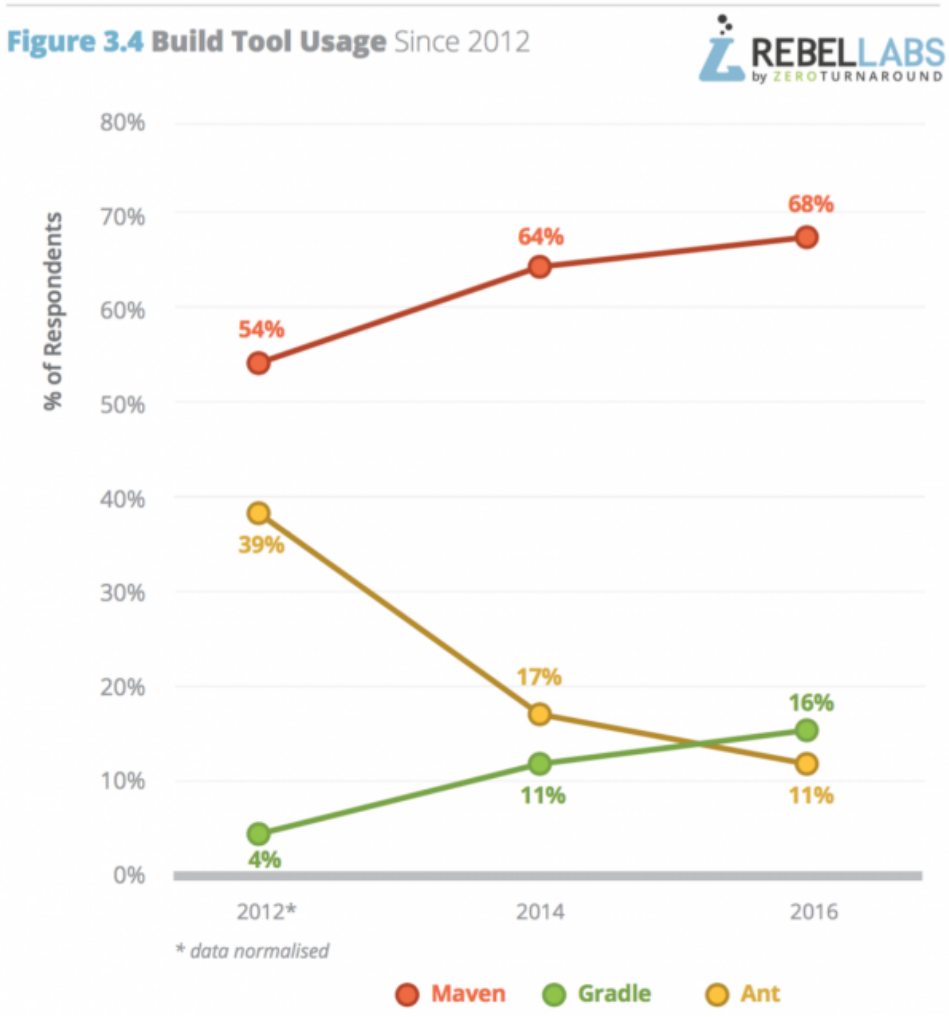
Použití build nástrojů na GitHubu

- Poznámka: graf NEZOBRAZUJE kumulativní součet hodnot, GitHub jenom exponenciálně roste) :-)
  - Zdroj: vlastní data-mining GitHub API





Použití build nástrojů podle RebelLabs



Poznámka: Řekl bych, že tady nebudou zahrnuti Android vývojáři. RebelLabs report se totiž vytváří dotazníkovým šetřením a k Android vývojářům se nemusí dostat (společnost ZeroTurnaround – tvůrce nástroje JRebel znají zejména Java EE vývojáři).

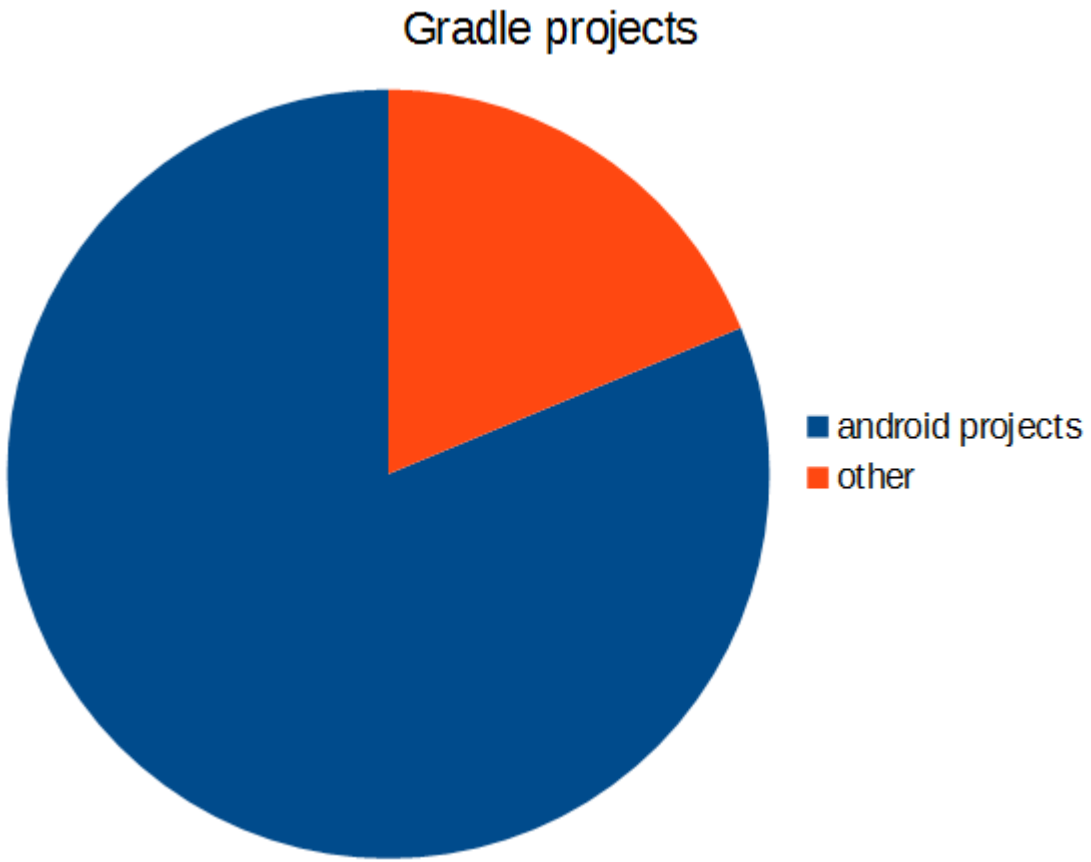
Projekty bez build nástrojů

- V čem jsou projekty bez build nástrojů naprogramovány?
  - Zdroj: vlastní data-mining GitHub API

```
• Idea:    330 052
• Eclipse: 290 236
• NetBeans: 83 721
```

Kde se používá Gradle?

- Převážně na Androidu :-)



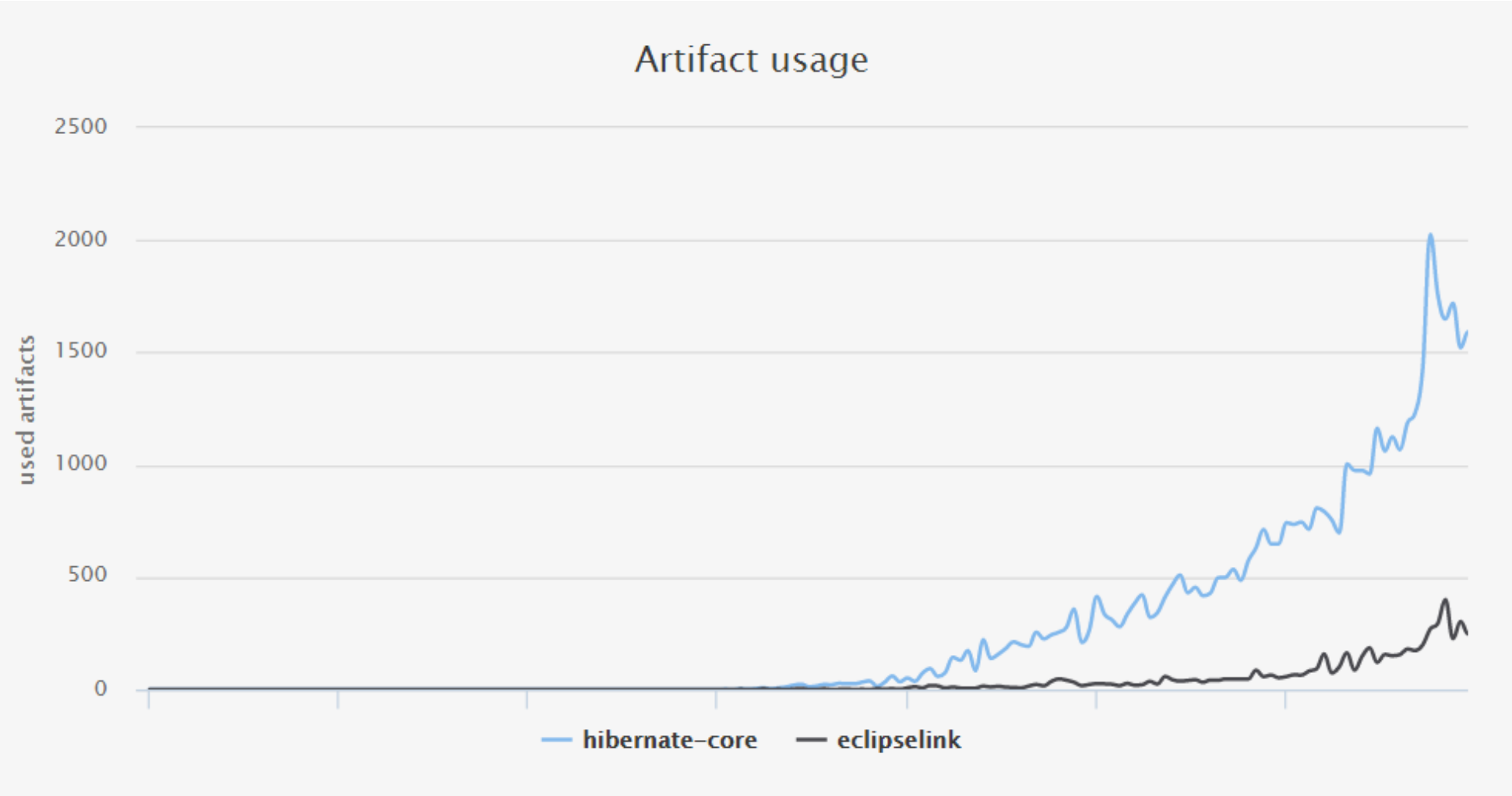
GitHub alternatives

- Spoustu statistik mám z GitHubu, protože je to zdaleka největší online služba pro publikování a sdílení aplikací. A také ho extenzivně využívám. Nicméně existuje pár alternativ:
  - https://bitbucket.org/ - umí neomezený počet privátních repozitářů
  - https://gitlab.com/ - takový lepší GitHub i BitBucket … ale moc se nepoužívá

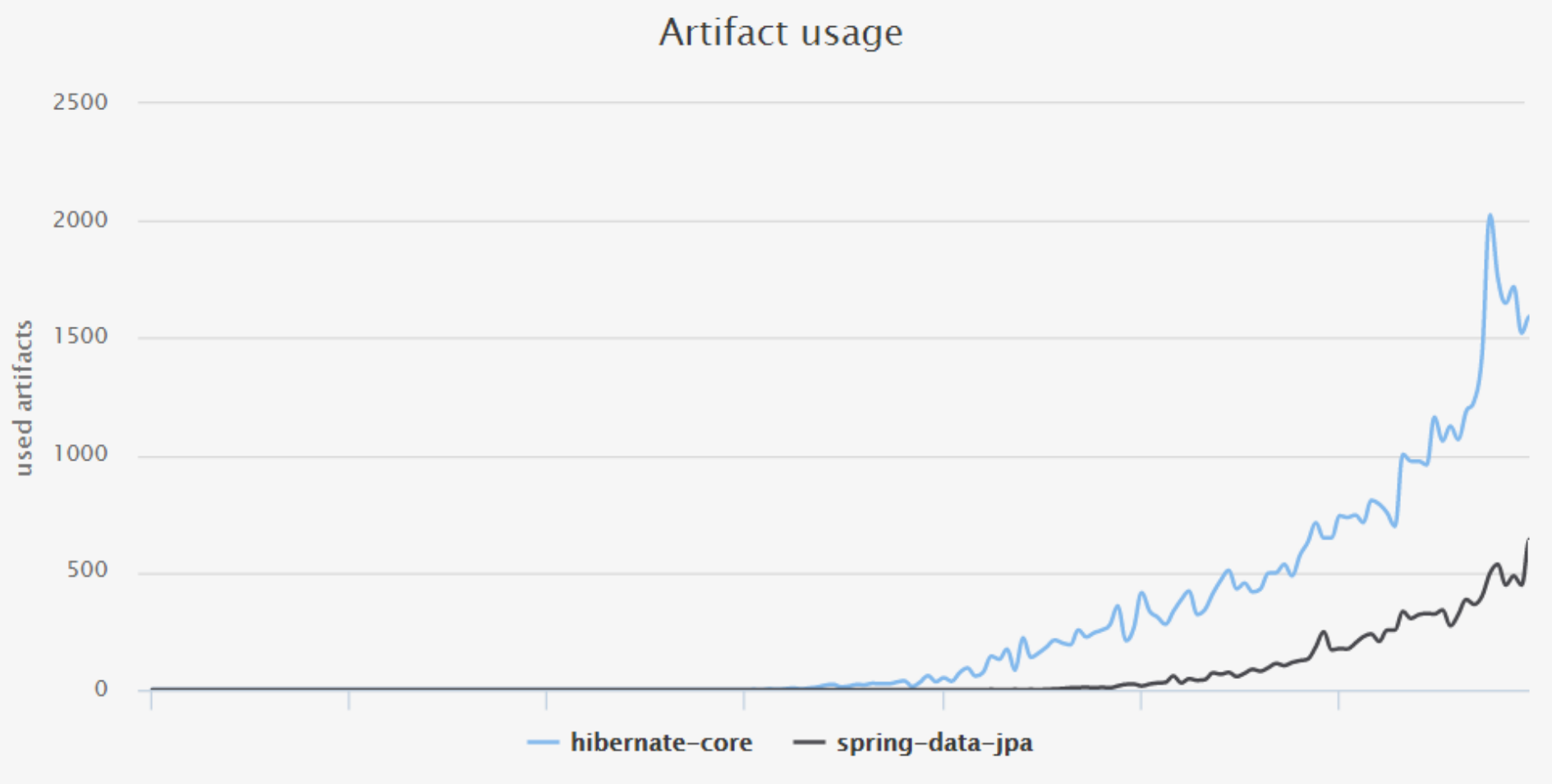
Hibernate vs. MyBatis vs. XYZ

- Nejpoužívanější frameworky pro práci s databází:
  - Hibernate, Spring Data JPA
  - MyBatis
  - JOOQ
  - QueryDSL
  - Třída JdbcTemplate ze Springu
  - Následující statistiky jsou získané převážně
  - z GitHubu a NEJEDNÁ se o kumulativní součet:
  - https://javalibs.com/custom-chart

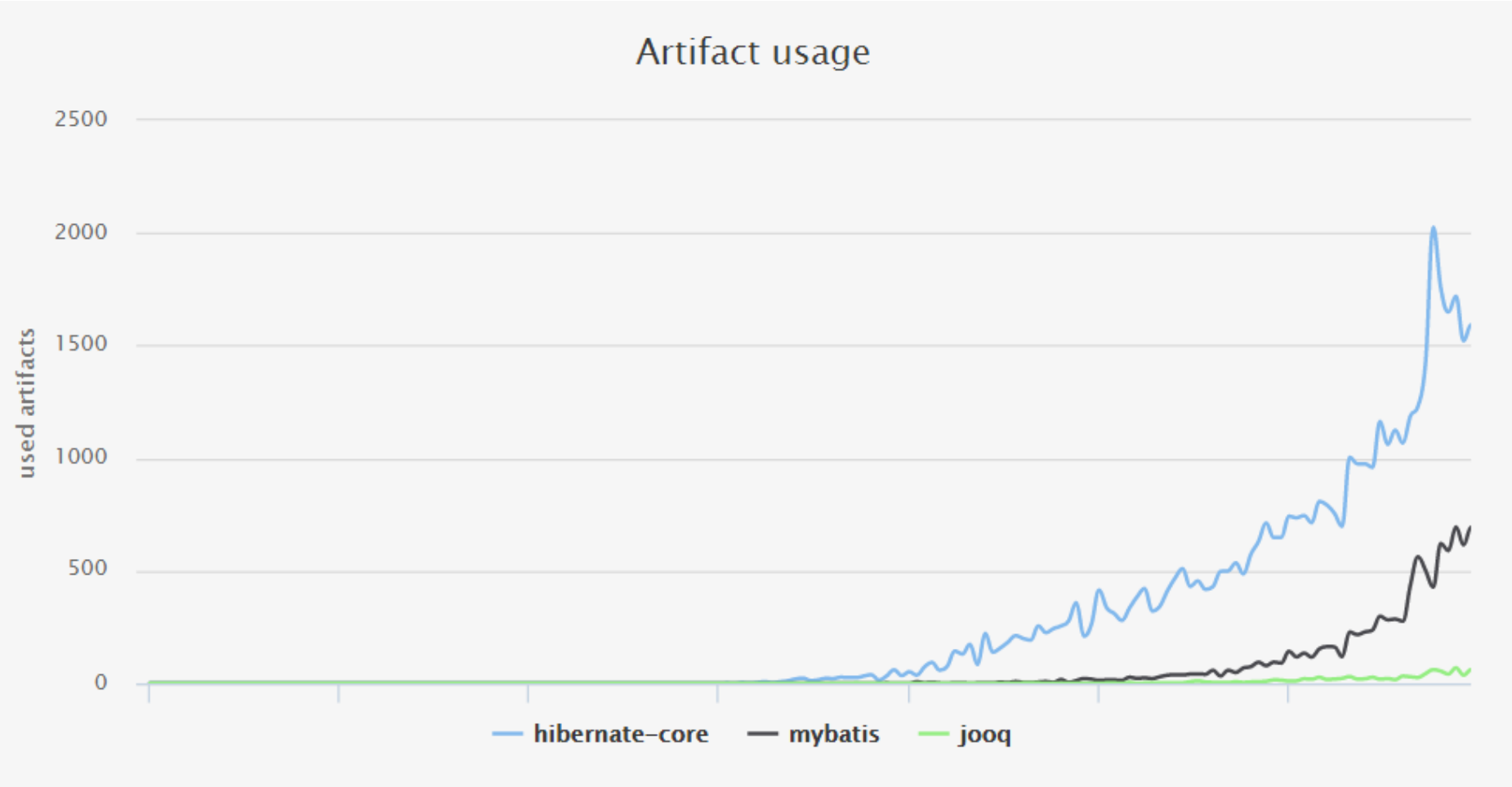
Hibernate vs. EclipseLink



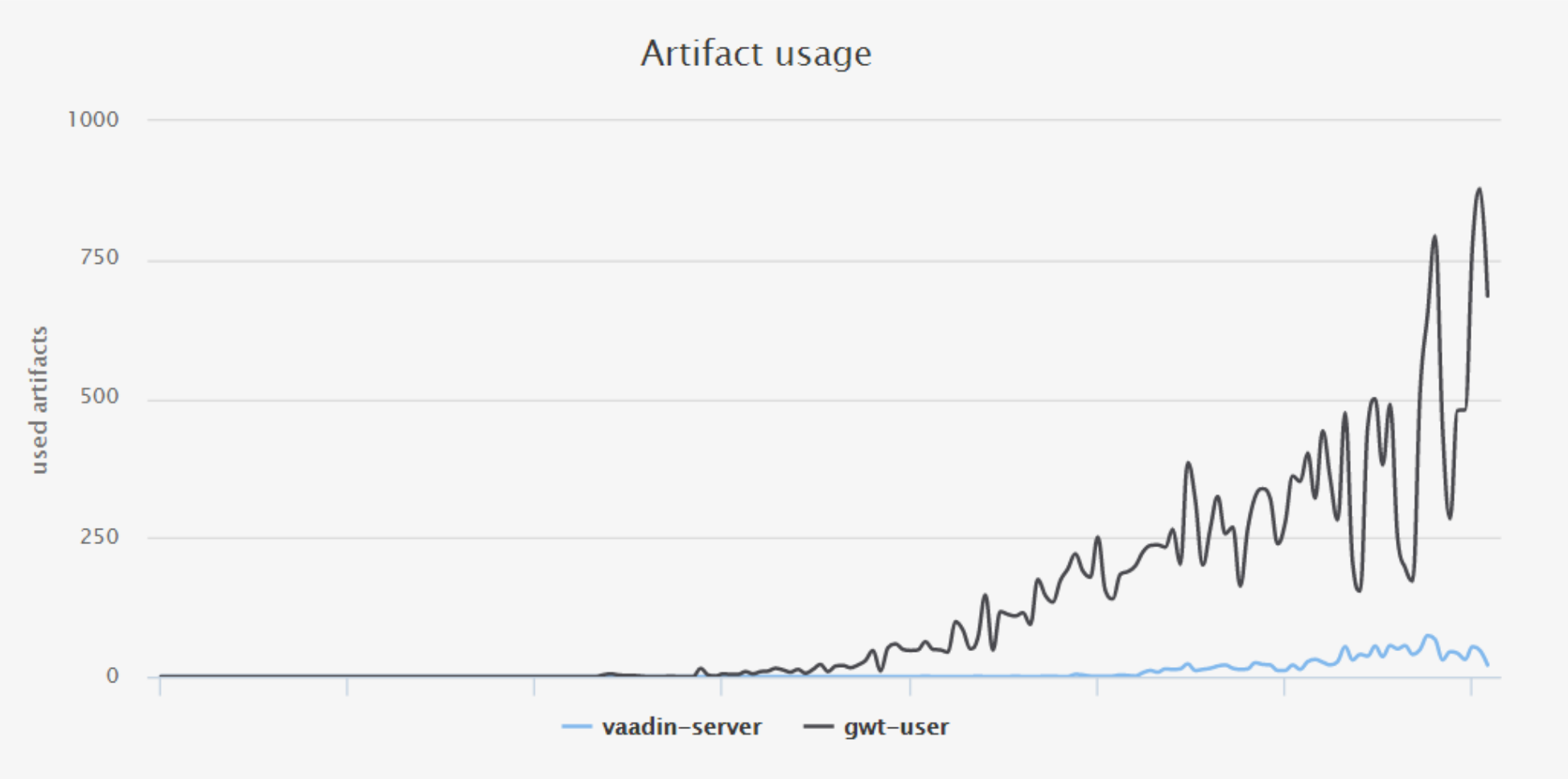
Hibernate vs. Spring Data JPA



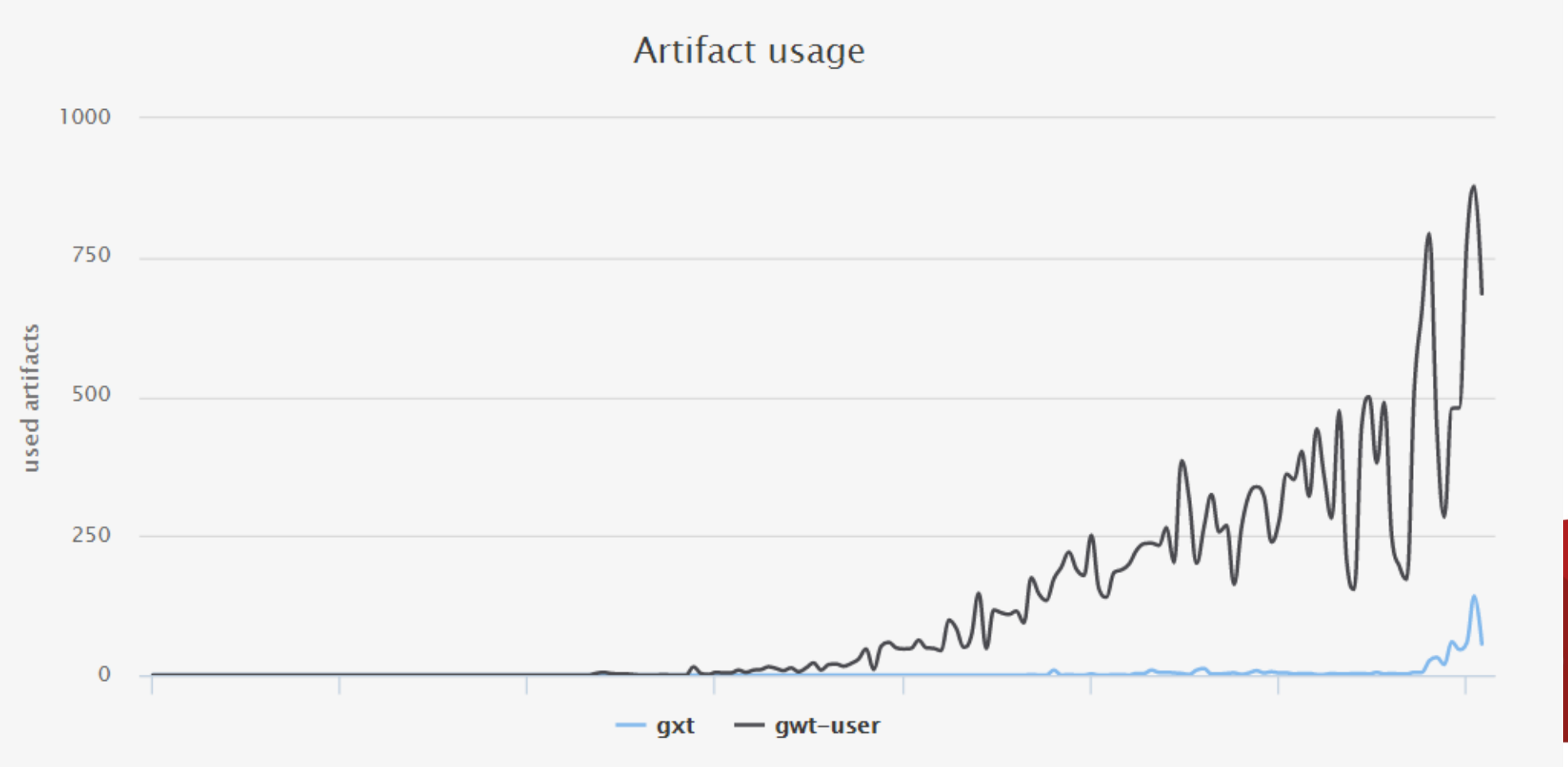
Hibernate vs. MyBatis vs. JOOQ



Vaadin vs. GWT

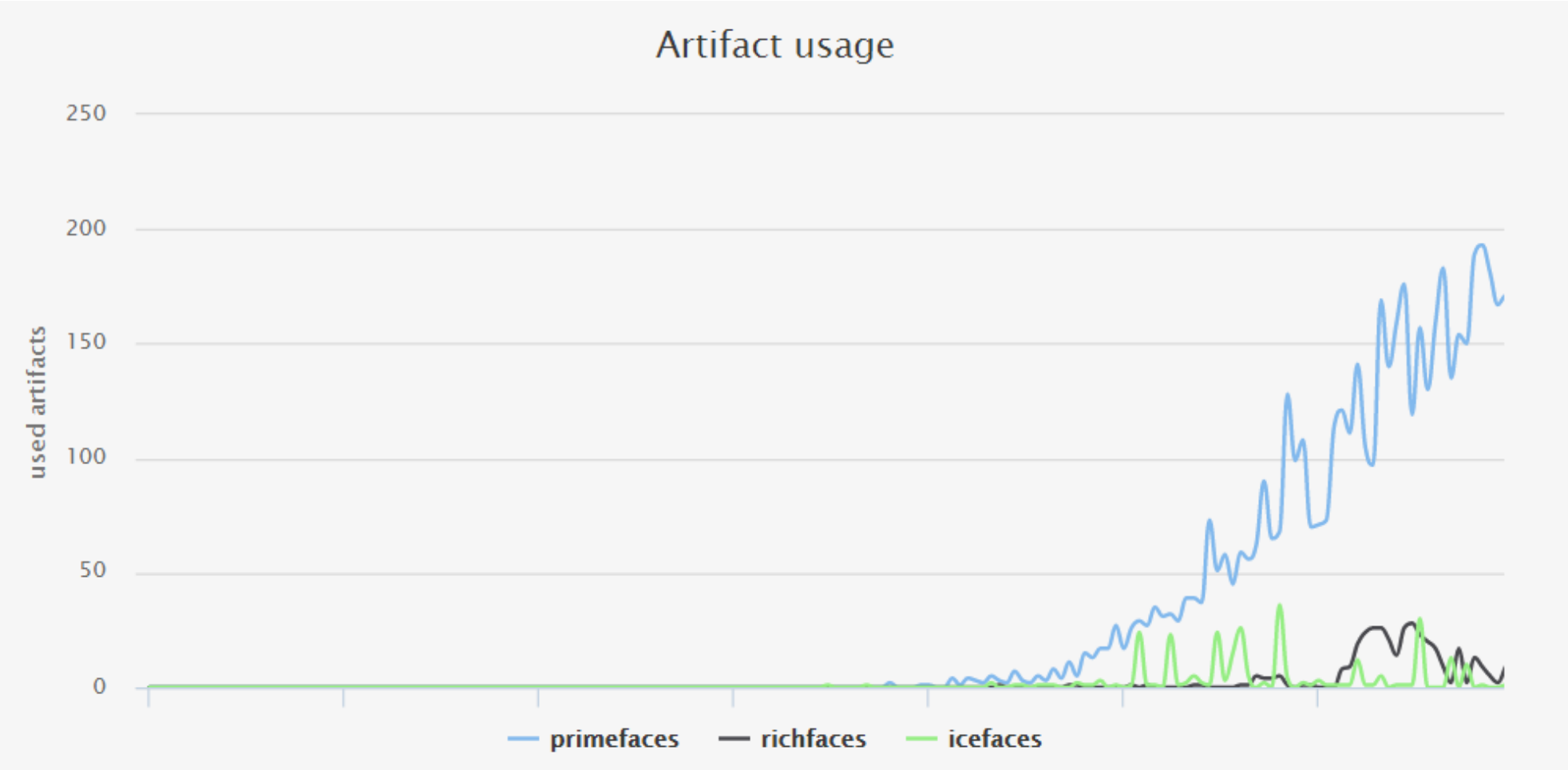


Sencha vs. GWT



*Faces

Poznámka: RichFaces jsou oficiálně mrtvé, kdyby o tom náhodou někdo nevěděl :-)



Další porovnání

- Kupa zajímavých informací je zde:
  - https://javalibs.com/charts

Jak zůstat „v obraze“?

- https://www.reddit.com/r/java/
- https://news.ycombinator.com/news (Hacker News)
- https://www.topjavablogs.com (moje veřejná RSS čtečka)
- https://github.com/akullpp/awesome-java

- Jirka Pinkas Twitter: @jirkapinkas

Děkuji za pozornost.

www.JavaDays.cz

www.gopas.cz