# PerfEval

Nástroj pro vyhodnocování
výsledků benchmark testů

# Co PerfEval dělá?

Uživatel změří pomocí nějakého benchmarku svůj software

Systém PerfEval výsledky strojově porovnává a zjišťuje zda došlo ke zhoršení výkonu

PerfEval je tedy systém pro strojové porovnávání výsledků výkonnostních testů

# Používání nástroje PerfEval

| |
|---|
| perfeval init |
| perfeval index-new-result <path-to-file> |
| perfeval index-all-results <path-to-dir> |
| perfeval evaluate |
| perfeval list-undecided |
| perfeval evaluate –graphical |
| perfeval config [--option=value] |

# Základní struktura kódu

# Třída Main

- metoda *main* se stará o nalezení správného příkazu a spuštění jemu příslušejícího kódu
- ostatní metody ve třídě se starají o částečnou kontrolu argumentů a nastavování správného exit kódu

# Package resultDatabase

Obsahuje rozhraní *IDatabase*, které definuje chování databáze s výsledky testů

- Získání nejnovějších N záznamů
- přidání souboru (1 záznamu)
- přidání složky se záznamy

Implementace *CacheDatabase*

Záznam *DatabaseItem*

- Cesta k souboru
- Datum vytvoření
- Verze SW ke které patří

# Package perfevalConfig

Třída *ConfigManager*

Stará se o přepisování konfiguračního souboru

# Package perfevalInit

Balíček zodpovědný za inicializaci systému

PerfEvalInitializer

- logika vytváření konfiguračního souboru a konfigurace

IniFileData

- obsah, zapisování, čtení konfiguračního souboru

# Package measurementFactory

| | |
|---|---|
| Balíček zodpovědný za vytvoření vhodného parseru na výsledky | |
| ParserFactory | • factory třída, která vrací instanci IMeasurementParser |
| UniversalTimeUnit | • každý benchmark umí měřit v jiných jednotkách<br>• tato třída se je snaží sjednotit v podobě nanosekund |
| IMeasurementParser | • definuje vlastnosti parseru |
| Measurement | • představuje výsledky měření jednoho konkrétního testu |

# Package evaluation

Balíček zodpovědný za prezentování výsledků vyhodnocování

| | |
|---|---|
| IMeasurementComparisonResult | • definuje vlastnosti objektu, které jsou zapotřebí k jeho prezentování jako výsledku vyhodnocení |
| MeasurementComparisonResult | • implementuje výše zmíněné rozhraní<br>• konstruktor mezi sebou vyhodnotí 2 objekty typu Measurement a vrátí výsledek porovnání |
| MeasurementComparisonResultView | • serializovatelná verze IMeasurementComparisonResult |

ResultPrinter – stará se o tisk výsledků do zadaného objektu PrintStream

# Package performanceComparatorFactory

| | |
|---|---|
| Balíček určený ke konstrukci správného komparátoru testů | |
| Enum ComparisonResult | • enum, který popisuje výsledek porovnání |
| IPerformanceComparator | • definuje chování komparátoru<br>• 2xList<UniversalTimeUnit> => ComparisonResult |
| AlwaysXXXComparator | • výsledkem porovnání je vždy XXX |
| Basic/Normal/BootstrapPerformanceComparator | • Skutečné komparátory pro porovnání testů s různou úrovní přihlédnutí ke statistice |
| Bootstrap | • třída starající se o bootrapování na vstupních listech s časy |
| ComparatorFactory | • konstruuje vhodný komparátor |

# Package perfevalCLIEvaluator

- Balíček, který se stará o zpracování příkazu *evaluate* a *list-undecided*
  1. Načtení souborů z databáze
  2. Naparsování souborů
  3. Porovnání testů
  4. Vytisknutí výsledků

# Package perfevalGraphicalEvaluator

- Balíček, který zpracovává příkaz *evaluate --graphical*
- Připravuje data pro vykreslení webové stránky
- Po řádném ukončení aplikace data smaže