MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Moderní nástroje pro měření výkonu v praxi

BAKALÁRSKA PRÁCA

Tomáš Borčin

Brno, jeseň 2014

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Vedúci práce: Mgr. Marek Grác, Ph.D.

Poďakovanie

Rád by som sa poďakoval Mgr. Marekovi Grácovi, Ph.D. za vedenie a rady poskytnuté pri písaní tejto práce. Veľká vďaka patrí aj Mgr. Martinovi Večeřovi a Ing. Pavlovi Macíkovi. V neposlednej rade ďakujem mojej rodine, hlavne mojim rodičom, Emílii a Ladislavovi za ich podporu pri štúdiu.

Zhrnutie

Cílem práce je nastudovat a popsat moderní nástroje na testování výkonu aplikací a velmi detailně porovnat jejich vlastnosti (tj. funkcionalita, použitelnost, podpora v IDE, možnosti nastavení, generování reportů atd.). Student napíše jednoduchou aplikaci, která bude odpovídat na dotazy za použití několika protokolů (především WS, JMS, REST, HTTP), kterou nasadí na JBoss AS 7. Poté navrhne několik testovacích scénářů pro všechny nástroje a spouští měření vyvinuté aplikace. Každé měření musí být provedeno několikrát a bude sledována spolehlivost výsledků. Student by se měl pokusit odhalit příčinu případných rozptylů ve výsledcích. Dále by měla být změřena teoretická propustnost samotného náastroje.

Nástroje pro porovnání:

- Apache jMeter - jmeter.apache.org
- PerfCake - perfcake.org
- Faban - faban.org
- Gatling - gatling-tool.org

Kľúčové slová

performance testy, testovanie, porovnanie nástrojov, Apache JMeter, PerfCake, Faban, Gatling

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Obrovský rozvoj informačných technológií so sebou prináša zvyšovanie nárokov na výpočtový výkon [1]. Tento problém sa dá riešiť dvoma spôsobmi. Jednoduchým riešením by bolo neustále zvyšovanie systémových zdrojov spolu s rastúcimi požiadavkami. Treba podotknúť, že toto riešenie je z dlhodobého hľadiska neefektívne, pretože investície do systémových zdrojov budú s ďalším vývojom a údržbou systému rásť, kým neprekročia únosnú hranicu a systém sa stane neudržateľným [2]. Prácnejším, ale o to efektívnejším riešením je testovanie aplikácií a analýza ich výkonu [3]. Test výkonu aplikácie sa uskutočňuje s cieľom zistiť, ako daná aplikácia pracuje pod určitou záťažou. Taktiež môže slúžiť na vyšetrovanie, meranie, potvrdzovanie alebo overovanie ďalších vlastností systému, ako rozšíriteľnosť a efektívnosť využívania prostriedkov. Najčastejšie používanými testami výkonu sú záťažové testy, únavové testy, testy výdrže, kulminačné testy, konfiguračné testy a separačné testy [3].

Výber správneho nástroja na testovanie je často náročnou úlohou, pričom kriticky dôležitá funkcionalita závisí od rôznych faktorov. Testovací nástroj určený na občasné testovanie jednoduchej aplikácie nemusí poskytovať detailné informácie o priebehu testov, avšak jednoduchosť používania a rýchlosť prípravy testov môže byť kritická. Toto zadanie môže spĺňať aj nástroj, ktorý nepodporuje najnovšie technológie a jeho vývoj prebieha oproti iným nástrojom veľmi pomaly. Naopak tomu môže byť vo firme s desiatkami zamestnancov, ktorá vyvíja nové aplikácie s použitím najmodernejších technológií. Tím skúsených testerov denne vykonávajúci množstvo testov kladie vysoké požiadavky na automatizáciu, dobrú organizáciu a presnosť testov. Výsledky testov generované v užívateľsky čitateľnej forme môžu byť prezentované vedeniu firmy bez nutnosti ďalšej úpravy. Pravidelná aktualizácia, aktívny vývoj a poskytovaná užívateľská podpora zohráva nemenej dôležitú rolu.

Úlohou tejto práce je porovnať nástroje používané k testovaniu výkonu aplikácii v praxi a poksytnúť tak čitateľovi prehľad. V práci sú popísané klady a zápory jednotlivých nástrojov a čitateľ si na základe svojich kritérií môže zvoliť nástroj, ktorý mu najviac vyhovuje.

Kapitola 2

Testy výkonu

Testovanie je nevyhnutnou súčasťou vývoja softvéru. Existuje mnoho testov, ktoré majú za účel odhaliť rôzne typy chýb. Spoločnosti vyvíjajúce softvér si veľmi dobre uvedomujú dôležitosť testovania a preto sú testy nedeliteľnou súčasťou životného cyklu vývoja softvéru. Napriek tomu sú testy výkonu softvéru podceňované a nie je im prikladaný dostatočný dôraz [3]. Testy výkonu sa delia na základné testy, záťažové testy, stresové testy, testy vytrvalosti, testy špičky, konfiguračné testy a izolačné testy.

2.1 Základné testy

Základné testy určujú čas potrebný na vykonanie jednotlivých transakcií jedným užívateľom. Za účelom získania čo najpresnejších údajov je nutné, aby počas vykonávania testov systém nevykonával žiadne iné aktivity. Pomocou týchto informácií bude možné zistiť, v akej miere klesol výkon systému s ohľadom na záťaž systému a počet užívateľov pracujúcich so systémom.

2.2 Záťažové testy

Najpoužívanejším a najjednoduchším testom je záťažový test. Počas tohto testu je systém vystavený stálej záťaži, pričom je monitorovaná dostupnosť služieb, priepustnosť a čas odozvy. Tento test je najvernejšou reprezentáciou reálneho používania systému. Vkladanie a čítanie dát musí byť oneskorené tak, ako je tomu pri práci užívateľa.

2.3 Stresové testy

Stresové testy vystavujú aplikáciu rastúcej záťaži a tým spôsobia, že celá aplikácia, alebo jej časť zlyhajú. Zlyhaním systému môže byť nedostupnosť celej aplikácie, jej služby, alebo veľmi dlhý čas odozvy. Následne je možné určiť maximálnu únosnú záťaž systému. Znalosť limít systému je dôležitá pre budúci vývoj aplikácie. Maximálna záťaž systému by mala poskytovať dostatočnú rezervu v závislosti na spôsobe používania aplikácie1.

2.4 Testy vytrvalosti

Testy vytrvalosti podrobujú systém stálej očakávanej záťaži, pričom sú monitorované rôzne časti systému. Vďaka tomu je možné odhaliť problémy s pamäťou, neuvoľnené zdroje a celkovú degradáciu výkonu. Je potrebné zabezpečiť, aby výkon časom neklesal, ale bol rovnaký ako pri spustení, prípadne sa zvyšoval. Cieľom testu je zistiť, ako sa systém správa pri stálom používaní, tak ako tomu bude po jeho nasadení.

2.5 Testy špičky

Počas testu špičky dochádza k náhlemu zvýšeniu záťaže. Úlohou tohto testu je zistiť ako systém reaguje na náhle zmeny záťaže.

2.6 Konfiguračné testy

Konfiguračné testy pozorujú, ako zmeny konfigurácie ovplyvňujú výkon, jednotlivé komponenty a celkové správanie systému. Zmeny konfigurácie môžu systému pomôcť lepšie zvládať zmeny záťaže.

2.7 Izolačné testy

Izolačné testy pomáhajú odhaliť chybné komponenty opakovaným spúšťaním neúspešných testov.

Kapitola 3

Metodika porovnávania nástrojov

1. Určenie a popis hodnotiacich kritérií jednotlivých nástrojov. Z hľadiska používateľa je dôležitý čas odozvy, obchodné záujmy reprezentuje priepustnosť a pre efektivitu systému je dôležité využitie zdrojov. Sem patrí aj určenie obmedzení a nájdenie optimálneho nastavenia systému.

2. Určenie testovacieho prostredia, zdrojov a nástrojov dostupných pre testovanie. Dobrá znalosť systému a jeho častí je kľúčová pri identifikácii problémov v počiatočnej fáze testovania a taktiež výrazne zvyšuje efektivitu testov.

3. Naplánovanie a navrhnutie testov, identifikácia vzorových užívateľov, simulácia ich rozmanitosti v testoch, definovanie testovacích dát.

4. Konfigurácia testovacieho prostredia. Sem patrí príprava nástrojov, zdrojov potrebných k vykonaniu testovacích scenárov a monitorovacích prostriedkov.

5. Implementácia testov v súlade s testovacím dizajnom.

6. Spustenie testov. Sem patrí vykonanie a monitorovanie testov, overenie testov a zbieranie výsledkov.

7. Analýza výsledkov, opätovné spúšťanie testov. Posledný bod zahŕňa opakovanú analýzu, vylepšovanie a spúšťanie testov. To vedie k zlepšovaniu výsledkov až po určitý konečný bod, ktorý určuje úzke miesto ďalšieho vývoja aplikácie.

8. Zhodnotenie a porovnanie nástrojov na základe hodnotiacich kritérií.

Kapitola 4

Nástroje

4.1 Apache JMeter

Apache JMeter je písaný v jazyku Java a dostupný pod open-source licenciou. Nástroj poskytuje podporu pre testovanie statických, ako aj dynamických zdrojov a podporuje širokú škálu protokolov [4]. Najnovšia verzia 2.11 je dostupná na stránke `www.jmeter.apache.org/download_jmeter.cgi`.

4.1.1 Inštalácia

Keďže je Apache JMeter aplikácia písaná v jazyku Java, vyžaduje k svojmu behu systém s nainštalovaným Java behovým prostredím JRE[1] verzie 6 a vyššej. V operačnom systéme Ubuntu je možné Apache JMeter nainštalovať aj cez softvérové centrum, alebo pomocou príkazu *sudo apt-get install jmeter*. Inštalácia cez softvérové centrum, alebo pomocou príkazu *sudo apt-get install jmeter* je jednoduchšia, ale obsahuje dva roky starú verziu 2.8.1.

4.1.2 Používanie

Spustenie programu je možné z príkazového riadka pomocou príkazu *jmeter.bat* pre Windows a *./jmeter.sh* pre Linux a MacOS, prípadne univerzálnym príkazom *java -jar ApacheJMeter.jar*. V nasledujúcej tabuľke je zoznam argumentov, ktoré sa dajú použiť pri spúšťaní programu.

1. Java Runtime Environment(JRE) je prostredie umožnujúce spúšťať Java aplikácie. Aplikácia je skompilovaná do štandardizovaného a prenosného binárneho formátu. Vo výsledku je možné túto aplikáciu spustiť v akomkoľvek systéme, ktorý obsahuje JRE.

Argument	Popis	Príklad
-h	Zobrazenie informácií o spustení programu.	./jmeter.sh -h
-v	Zobrazenie verzie aplikácie.	./jmeter.sh -v
-p	Použije sa zadaný property súbor.	./jmeter.sh -p subor
-q	Doplnkové property súbory k použitiu.	./jmeter.sh -q subor1 subor2
-t	jMeter testovací súbor, ktorý sa má spustiť.	./jmeter.sh -t test
-l	Logovací súbor pre vzorky.	./jmeter.sh -l sampleLog
-j	Súbor na logovanie behu jMeter programu.	./jmeter.sh -j jMeterLog
-n	Spustenie jMeter v textovom režime.	./jmeter.sh -n
-s	Spustenie jMeter servera.	./jmeter.sh -s server
-H	Nastavenie proxy servera pre jMeter.	./jmeter -H proxy
-P	Nastavenie portu proxy servera pre jMeter.	./jmeter.sh -P port
-N	Nastavenie zoznamu hostov bez proxy.	./jmeter.sh -N localhost *apache.org
-u	Nastavenie loginu pre proxy server, ktorý bude používať jMeter.	./jmeter.sh -u username
-a	Nastavenie hesla pre proxy server, ktorý bude používať jMeter.	./jmeter.sh -a password
-J	Definuje ďalšie property.	./jmeter.sh -J property=1
-G	Definuje globálne property, ktoré sa posielajú serveru.	./jmeter.sh -G port=123
-D	Definuje systémové property.	./jmeter.sh -D property=1
-S	Definuje systémové property súbory.	./jmeter.sh -S property1 property2
-L	Určuje úroveň logovania.	./jmeter.sh -L jmeter.util=INFO
-r	Spustenie vzdialených serverov, ktoré sú definované v remote_hosts.	./jmeter.sh -r
-R	Spustenie zadaných vzdialených serverov. Nahradzuje remote_hosts.	./jmeter.sh -R server1 server2
-d	Nastaví domáci adresár pre jMeter.	./jmeter.sh -d /home/user/adresar
-X	Ukončí vzdialené servery na konci testu.	./jmeter.sh -X

Tabuľka 4.1: Apache JMeter argumenty

4.1.3 Dokumentácia

Užívateľská dokumentácia je dostupná na stránke http://jmeter.apache.org/usermanual/index.html. Apache JMeter je pomerne robustný nástroj, čo sa odrazilo aj na rozsahu dokumentácie. Ako pozitívum hodnotím wiki stránku http://wiki.apache.org/jmeter/, ktorá obsahuje tutoriály, popisuje rozšírenia a tiež poskytuje informácie pre vývojárov. Nachádza sa tu aj tutoriál popisujúci nastavenie Apache JMetra pre používanie v Eclipse IDE. Pozitívom je aj FAQ stránka http://wiki.apache.org/jmeter/JMeterFAQ ponúkajúca odpovede na najčastejšie problémy, na ktoré môže užívateľ pri používaní Apache JMetra naraziť. Dokumentácia je navzdory svojej rozsiahlosti prehľadná a pre každý typ testu poskytuje samostatnú kapitolu popisujúcu vytvorenie testu od úplného začiatku.

4.1.4 Použiteľnosť vo vývojovom prostredí

Eclipse
Tutoriál http://wiki.apache.org/jmeter/JMeterAndEclipseHowTo z wiki stránky čitateľa hneď v úvode presmeruje na stránku http://people.apache.org/~mkostrze/jmeter-eclipse/jmeter-eclipse.html, ktorá bola naposledy upravená 10.03.2004. Stránka odkazuje na neexistujúce zdroje a nepodarilo sa mi podľa nej nastaviť Apache JMeter tak, aby sa dal jednoducho použiť v Eclipse IDE verzie 4.4.1.

NetBeans
Apache JMeter doplnok je možné do NetBeans IDE nainštalovať v menu Nástroje (Tools) → Doplnky (Plugins). V okne doplnkov je potrebné vybrať panel Dostupných doplnkov (Available Plugins), vyhľadať JMeter doplnok a spustiť inštaláciu. Následne je možné do projektu pridať nový JMeter plán. Doplnok vytvorí v adresári projektu nový priečinok "jmeter", ktorý obsahuje jednoduchý návrh testu. V okne služieb vznikne nová služba Generátory záťaže (Load Generators) obsahujúca generátor JMeter, ktorý spustíme tak, že mu priradíme testovací plán, ktorý sa následne spustí automaticky.

IntelliJ IDEA
Doplnok pre IntelliJ IDEA je nutné stiahnuť zo stránky https://plugins.jetbrains.com/plugin/7013. V IntelliJ IDEA v menu Nastavenia (Settings) → IDE nastavenia (IDE Settings) → Doplnky (Plugins) vyberieme možnosť Nainštalovať doplnok z disku (Install plugin from disk). Po dokončení inštalácie a reštartovaní IDE je nutné nastaviť v menu Nastavenia → Nastavenia projektu (Project settings) → JMeter domovský adresár pre Apache JMeter. Následne je možné do projektu vložiť JMeter súbor a v IDE ho aj upravovať. Po spustení testovacieho súboru sa otvorí okno Apache JMeter, v ktorom je možné spustiť testy.

Apache Maven[2]
Chýbajúci doplnok pre Eclipse IDE je možné nahradiť Maven doplnkom. Návod je k dispozícii na wiki stránke Apache JMeter https://wiki.apache.org/jmeter/JMeterMavenPlugin. Po nastavení doplnku a vytvorení JMeter testovacích scenárov sa tieto testy spúšťajú automaticky pri kompilácii projektu.

4.1.5 Generovanie reportov

Apache JMeter poskytuje širokú škálu spôsobov, pomocou ktorých je možné monitorovať priebeh testov. Monitor zaznamenávajúci využitie procesora a pamäte RAM nie je súčasťou Apache JMeter, ale je poskytovaný zdarma ako rozšírenie. Inštalácia doplnku prebieha skopírovaním knižnice JMeterPlugins-Standard.jar do adresára lib/ext, ktorý sa nachádza v domovskom adresári nástroja. Následne je nutné nastaviť aj Apache agenta, ktorý bude na strane servera monitorovať využitie procesora a pamäte. Adresár obsahujúci agenta je nutné priložiť k serveru tak, aby bolo možné pri štarte servera spustiť aj agenta, ktorý sa štartuje spustením skriptu startAgent.sh. Agent používa na komunikáciu port číslo 4444.

2. Apache Maven je softvér určený na automatizáciu tvorby aplikácií.

4.2 Perfcake

PerfCake je podobne ako JMeter open-source nástroj písaný v jazyku Java. Uprav casy a verzie V súčasnosti je k dispozícii verzia 2.0. Dátum vydania verzie 3.0 je naplánovaný na 23.9.2014. Stránka projektu: www.perfcake.org/. Github: www.github.com/PerfCake/PerfCake. Na stránke www.github.com/PerfCake sú dostupné pluginy pre Maven, IntelliJ IDEA, Eclipse a rôzne ukážkové testovacie scenáre.

4.2.1 Inštalácia

Základným predpokladom pre používanie PerfCake je systém s Java Runtime Environment verzie 7 alebo vyššej. Tiež je nutné mať správne nastavenú $JAVA_HOME premennú prostredia. Pre operačné systémy Windows a Linux je na stránke www.perfcake.org/download/ pripravená binárna distribúcia, ktorú je možné používať hneď po stiahnutí a rozbalení archívu. Užívatelia operačného systému Mac si musia stiahnuť zdrojový kód a skompilovať ho. Týmto je program pripravený na používanie. Inštalácia je veľmi jednoduchá, prebehla bez komplikácií a zaberie len pár minút.

4.2.2 Používanie

Práca s programom prebieha prostredníctvom príkazoveho riadka. V systéme Windows spustíme PerfCake príkazom *perfcake.bat*, v systémoch Linux a MacOS *./perfcake.sh*. Po spustení týchto príkazov sa zobrazia argumenty, ktoré je možné používať pri spúšťaní programu.

Argument	Popis	Príklad
-s	Názov spúšťaného scenára. Scenár je uložený v adresári /resources/scenarios.	./perfcake.sh -s http-echo
-D	Názov vlastnosti a jej hodnota. Píšeme v tvare názov=hodnota	./perfcake.sh -s http-echo -D thread.count=100
-sd	Adresár, z ktorého má byť spustený scenár. Cesta k adresáru musí byť absolútna.	./perfcake.sh -s http-echo -sd /home/user/scenare
-md	Adresár pre správy. Cesta k adresáru musí byť absolútna.	./perfcake.sh -s http-echo -md /home/user/spravy
-pd	Adresár pre pluginy. Cesta k adresáru musí byť absolútna.	./perfcake.sh -s http-echo -pd /home/user/pluginy
-pf	Názov property súboru.	./perfcake.sh -s http-echo -pf property

Tabuľka 4.2: PerfCake argumenty

4.2.3 Dokumentácia

Užívateľská dokumentáciahttp://faban.org/docs/QuickStartTutorial.html sa nachádza na stránke www.perfcake.org/docs/perfcake-user-guide.pdf. Skladá sa zo siedmich kapitol, v ktorých popisuje architektúru, vytváranie scenárov, generovanie záťaže, posielanie správ, generovanie reportov, validáciu a rozšírenia PerfCake. Pre každú komponentu testu obsahuje jednoduchý príklad. Návod je jednoduchý, prehľadný a dobre sa v ňom orientuje.

4.2.4 Použiteľnosť vo vývojovom prostredí

je mozne pouzit v Eclipse Kepler a je tam aj GUI

4.2.5 Generovanie reportov

V prípade nástroja Perfcake sú všetky informácie vypisované do príkazového riadku v časových intervaloch, ktoré sa nastavujú v testovacích scenároch. Reporty som vytvoril presmerovaním výstupu z terminálu do súboru. Všetky informácie sú prehľadne umiestnené v jednom súbore.

4.3 Faban

Faban je napísaný v jazyku Java a jeho licencia je open-source. Skladá sa z dvoch hlavných komponent: Faban Driver Framework a Faban Harness. Driver Framework slúži na definovanie záťaže, riadenie životného cyklu a nastavenie reportovania. Faban Harness poskytuje užívateľské rozhranie k spúšťaniu a kontrolovaniu testov, generovaniu výsledkov a ukladaniu naplánovaných testov do fronty [5]. Aktuálna verzia Fabanu je 1.2 a dá sa získať na stránke projeku `www.faban.org/`. Faban sa nachádza aj na GitHub-e: `www.github.com/akara/faban`.

4.3.1 Inštalácia

Faban na rozdiel od predchádzajúcich testovacích nástrojov vyžaduje plnú JDK[3] inštaláciu verzie 1.5 a vyššej. Nutnosť inštalácie JDK hodnotím ako negatívum. Najnovšia verzia programu sa nachádza na stránke `www.faban.org/download.html`.

4.3.2 Používanie

Práca s programom prebieha prostredníctvom prehliadača. Najskôr je nutné spustiť Faban Master, ktorý sa spúšťa v Linuxe pomocou príkazu *./startup.sh*, pre Windows príkazom *startup-using-launcher.bat*. Súbory sa nachádzajú v priečinku master/bin a je nutné ich spustiť cez príkazový riadok, pretože po spustení program vypíše adresu, na ktorej beží Faban server. Túto adresu je nutné zadať do prehliadača. Otvorí sa rozhranie, v ktorom je nutné nastaviť vytvorený ovládač definujúci základné parametre testu. Následne je možné upravovať, spúšťať, ukončovať testy a prezerať si vygenerované výsledký testov. Syntax pre spúšťanie testov z príkazového riadku je *FABAN_HOME/bin/fabancli submit benchmark profile /configuration/run.xml*, kde FABAN_HOME je adresár obsahujúci Faban, fabancli je skript umožňujúci interakciu s Faban Harness. Prvý argument spúšťaného skriptu definuje príkaz, ktorý sa má vykonať. V našom prípade sa má zaregistrovať nový test. Druhý argument určuje ovládač použitý k spúšťaniu testov, nasledujúci argument určuje profil použitý k spusteniu testov a posledným argumentom je konfiguračný súbor, ktorý obsahuje základné informácie o teste, teda dĺžku testu, počet klientov a verziu Javy použitú v testoch.

4.3.3 Dokumentácia

Stručný tutoriál popisujúci inštaláciu a spustenie prvých testov je dostupný na stránke `http://faban.org/docs/QuickStartTutorial.html`. Návod popisujúci vytváranie záťaže je k dispozícii na stránke `http://faban.org/docs/CreatingWorkloadTutorial.html`. Ďalšie návody sú k dispozícii na stránke `http://faban.org/1.2/docs/index.html`.

3. Java Development Kit(JDK) je vývojová platforma pre programovací jazyk Java. Súčasťou balíka JDK sú komponenty a nástroje umožňujúce vývoj Java aplikácií a JRE umožňujúce ich spúšťanie.

4.3.4 Použiteľnosť vo vývojovom prostredí

Žiadne doplnky pre vývojové prostredia Eclipse, Netbeans, IntelliJ IDEA a nástroj Maven neboli nájdené. Faban je možné spúšťať len v príkazovom riadku.

4.3.5 Generovanie reportov

Vytváranie reportov a logovacích súborov prebieha automaticky. Pri štarte testu dostane každý test identifikačný reťazec, ktorý sa skladá z názvu profilu použitého pri testovaní, bodky, čísla a velkého písmena. Číslo a písmeno slúži ako jednoznačný identifikátor testu a Faban ho generuje automaticky. Výsledky testu je možné zobraziť v grafickom prostredí Faban Harness. V menu View Results je nutné vybrať test identifikovaný identifikačným reťazcom, čím sa zobrazí stručný report. Detailnejšie výsledky sú k dispozícii v hornom menu v záložkách Detailed Results, Run Log a Statistics. Výsledky testov sa ukladajú do adresára output, ktorý sa nachádza v adresári Fabanu. Pre každý test Faban vytvára adresár, ktorého názov je zhodný s identifikačným reťazcom testu a do tohto adresára ukladá všetky reporty a logy, ktoré patria k danému testu. Faban automaticky zaznamenáva využitie procesora počas testu, ale záznam využitia operačnej pamäte chýba a popis nastavenia chýba aj v dokumentácii.

4.4 Gatling

Gatling je open-source, podobne ako všetky predchádzajúce nástroje. Kým predchádzajúce nástroje boli písané výhradne, alebo z veľkej časti v jazyku Java, Gatling je vyvíjaný v jazyku Scala. Gatling ako jediný nástroj nepoužíva jazyk XML[4] pre písanie testovacích scenárov. Scenáre sú písané v jazyku Scala. Jadro nástroja nie je závislé na konkrétnom protokole a je ľahko rozšíriteľné. V súčasnosti poskytuje podporu pre HTTP a JMS protokol [6]. Stránka projektu `www.gatling-tool.org/` je veľmi jednoduchá a obsahuje len pár odkazov, väčšinou na GitHub stránku projektu `www.github.com/excilys/gatling`.

4.4.1 Inštalácia

Staré verzie programu sa nachádzajú na stránke `www.github.com/excilys/gatling/wiki/Downloads`. Najnovšia stabilná verzia 2.1.1 je dostupná na stránke `http://gatling.io/download/`. Nachádza sa tu aj verzia 2.2.0-SNAPSHOT, ktorá je vo vývoji. Základným predpokladom pre používanie programu Gatling je JDK verzie 6, pričom sa odporúča mať nainštalovanú najnovšiu verziu JDK.

4.4.2 Používanie

Program je možné spustiť v príkazovom riadku príkazom *./gatling.sh* pre Linux a príkazom *gatling.bat* pre Windows. Spúšťacie skripty sa nachádzajú v priečinku bin. Nasledujúca tabuľka popisuje zoznam argumentov, s ktorými je možné spustiť program.

4. Extensible Markup Language je obecný značkovací jazyk vyvinutý a štandardizovaný konzorciom W3C.

Argument	Popis	Príklad
-nr	Spustenie testov bez generovania reportu.	./gatling.sh -nr
-ro	Generovanie reportov do zadaného adresára.	./gatling.sh -ro /home/user/reporty
-df	Určuje adresár, z ktorého majú byť načítané testovacie dáta. Cesta k adresáru musí byť absolútna.	./gatling.sh -df /home/user/data
-rf	Adresár pre ukladanie výsledkov. Cesta k adresáru musí byť absolútna.	./gatling.sh -rf /home/user/vysledky
-bf	Adresár pre requesty. Cesta k adresáru musí byť absolútna.	./gatling -bf /home/user/requesty
-sf	Vyhľadá spustiteľné scenáre v zadanom adresári.	./gatling.sh -sf /home/user/scenare
-sbf	Vyhľadá skompilované scenáre v zadanom adresári.	./gatling.sh -sbf /home/user/skompilovane
-s	Spustí zadaný scenár.	./gatling.sh -s scenar
-on	Zadaný argument sa použije ako východiskový názov pre výstupný adresár.	./gatling.sh -on nazov
-sd	Krátky popis spúšťaného testu, ktorý bude zahrnutý do reportu.	./gatling.sh -sd Popis

Tabuľka 4.3: Gatling argumenty

4.4.3 Dokumentácia

Zoznam užívateľských manuálov je k dispozícii na stránke http://gatling.io/docs/2.0.2/. Nachádza sa tu stručný a prehľadný návod popisujúci inštaláciu a spustenie prvých testov, ako aj pokročilejšie tutoriály detailne popisujúce nastavenie nástroja a vytváranie testov. Dokumentácia je prehľadná, dobre sa v nej orientuje a pozitívom je aj vzhľad, ktorý je kvalitne spracovaný.

4.4.4 Použiteľnosť vo vývojovom prostredí

Eclipse, Netbeans, IntelliJ IDEA
Gatling neposkytuje podporu pomocou samostatných doplnkov pre jednotlivé vývojové prostredia. Integrácia Gatlingu je realizovaná pomocou Maven buildovacieho nástroja, ktorý tieto vývojové prostredia podporujú.

Maven
Dokumentácia obsahuje sekciu Extensions, ktorá popisuje nastavenie Maven Plugin a Maven Archetype.
Maven Plugin je možné použiť v každom projekte, ktorý využíva Maven ako buildovací nástroj. Maven je možné použiť v Eclipse, IntelliJ IDEA, ako aj Netbeans. Pri nastavovaní Maven konfiguračného súboru podľa stránky http://gatling.io/docs/2.0.2/extensions/maven_plugin.html bolo nutné zmeniť hodnotu *io.gatling* v elemente *groupId* na hodnotu *io.gatling.highcharts*, pretože závislosť (Dependency) s hodnotou *io.gatling* sa v maven repozitári nebola k dispozícii.

Gatling je možné pomocou Maven Archetype iba do vývojových prostredí podporujúcich jazyk Scala a obsahujúcich Maven. Aj keď testované vývojové prostredia neobsahujú natívnu podporu jazyka Scala je možné túto funkcionalitu implementovať pomocou doplnkov dostupných pre každé vývojové prostredie. Manuál popisujúci integráciu Gatling do vývojového prostredia pomocou Maven Archetype sa nachádza na stránke http://gatling.io/docs/2.0.2/extensions/maven_archetype.html.

4.4.5 Generovanie reportov

Po spustení testu sú výsledky priebežne vypisované do terminálu a zároveň sa vytvorí adresár, ktorý umožňuje zobraziť výsledky v prehľadnej grafickej podobe. Adresár obsahuje aj logovací súbor, ktorý zaznamenáva výsledok každej pre každý vykonaný dotaz. Výsledky som z terminálu presmeroval do súboru a pri spúšťaní testu nastavujem umiestnenie adresára s výsledkami na adresár, v ktorom sa spúšťa daný test. Dokumentácia obsahuje návod, pomocou ktorého je možné nastaviť zaznamenávanie využitia operačnej pamäte. Napriek veľkej snahe sa mi podľa tohto návodu nepodarilo nastaviť zaznamenávanie využitia pamäte. Nastavenie vyžadovalo zmenu konfiguračného súboru Gatlingu a inštaláciu programov tretích strán. Po zmene konfiguračného súboru sa stal nástroj nepoužiteľným a nástroj som musel inštalovať odznova. V dokumentácii navyše chýbal návod k inštalácii programov tretích strán. Návod je k dispozícii na stránke: http://gatling.io/docs/2.1.1/realtime_monitoring/index.html.

Kapitola 5

Testy

5.1 Testovaná aplikácia

K vytvoreniu testovanej aplikácie som použil programovací jazyk Java vo verzii 7 a ako vývojové prostredie NetBeans 8.0.1. Aplikácia sa skladá z dvoch častí: webových služieb a triedy obsluhujúcej Java Message Services (ďalej len JMS). Webové služby som naimplementoval pomocou Java aplikačného programovacieho rozhrania pre REST[1] webové služby, ktoré sa v skratke označuje ako JAX-RS. Aplikácia pozostáva zo siedmich webových služieb, ktoré sú volané pomocou HyperText Transfer Protocol (ďalej len HTTP) protokolu. HTTP je protokol umožňujúci prenos dokumentov vo formáte značkovacieho jazyka HyperText Markup Language (HTML).

Druhá časť aplikácie implementuje JMS rozhranie pre posielanie správ medzi dvomi klientami. Testovacie nástroje posielajú správy do fronty a tieto správy sú preposielané do druhej fronty, odkiaľ správu vyberie testovací nástroj.

Webová služba main prijíma na vstupe reťazec znakov a za použitia zvyšných webových služieb vypíše na výstup pôvodný reťazec, otočený reťazec, dĺžku reťazca, počet samohlások, spoluhlások, číslic a nakoniec veľkosť pôvodného reťazca v bytoch. Pre účely testov som službu upravil tak, aby po zadaní reťazca "generate: x"vygenerovala reťazec o dĺžke x znakov, pričom každý vygenerovaný znak má veľkosť jeden bajt. Veľkosťou správy rozumieme veľkosť vygenerovaného reťazca.

Služba memoryLeak sa od služby main líši len v tom, že simuluje únik pamäte. Simuláciu som naprogramoval pomocou mapy, ktorá nemá naimplementované metódy equals a hashCode. Správne naprogramovaná mapa musí mať tieto metódy korektne naimplementované, pretože pomocou nich rozhoduje, či sa vkladaná hodnota v mape už nachádza. Bez týchto metód moja mapa nedokáže detekovať duplicity a preto do mapy vkladá stále nové a nové hodnoty, a tým sa vytvorí únik pamäte. Službu používam len pri vytrvalostných testoch, ktoré sa skladajú z dvoch častí: Bez memory leaku a S memory leakom. Pri testovaní Bez memory leaku používajú testy službu main a pri testovaní S memory leakom volajú službu memoryLeak.

Poslednou priamo volanou službou je echo, ktorú používam pri testovaní teoretickej priepustnosti jednotlivých nástrojov. Táto služba vracia jednoduchý reťazec "echo", vďaka čomu dokáže server veľmi rýchlo vrátiť odpoveď a tým sa teoretické hrdlo fľaše

1. Representational state transfer je abstraktný architektúrny vzor pre distribuované prostredie.

presunie od služby k testovaným nástrojom. Hrdlo fľaše v softvérovom inžinierstve označuje tú časť systému, ktorá výrazne limituje priepustnosť celého systému.

5.2 Testovacie prostredie

K testovniu sú použité dva samostatné počítače. Aplikácia je nasadená na aplikačnom serveri JBoss Application Server 7.1.1.Final "Brontes". Server beží na počítači Acer Aspire 4830TG s procesorom Intel Core i5-2410 a 8GB RAM pamäťou. Na počítači je nainštalovaná 64-bitová verzia operačného systému Ubuntu 14.04 LTS. Generovanie záťaže prebieha na počítači RED HAT PC, alebo moj PC.

5.3 Testy

Pre potreby porovnania nástrojov je navrhnutých 7 typov testov. Medzi jednotlivými testami je JBoss server reštartovaný. Výsledky testov sú zaznamenávané pomocou generátorov výsledkov, ktoré ponúkaju jednotlivé nástroje. Medzi dotazmi klientov nie je nastavené oneskorenie.

1. **Základný test s jedným klientom**
   Prvý test má za úlohu simulovať maximálnu možnú záťaž systému jedným klientom, pričom je vykonávaný po dobu piatich minút. Na základe tohto testu je možné sledovať efektivitu generovania maximálnej záťaže, generovanie výsledkov a spotrebu zdrojov pri nízkej záťaži.

2. **Testy s rastúcim počtom klientov**
   K simulácii stresového typu testu slúži postupné zvyšovanie nárokov na systém. Test sa preto skladá zo série testov, pričom rastie množstvo klientov využívajúcich systém. Klienti zasielajú dotazy po dobu piatich minút. Je spustených päť druhov testov, z nich každý päťkrát, celkovo je spustených dvadsaťpäť testov pre každý nástroj. Množstvo klientov v testoch je: desať, päťdesiat, sto, stopäťdesiat a dvesto. Zvyšovanie záťaže spôsobí v istom bode zlyhanie systému. Okrem toho test slúži k identifikácii optimálneho množstva klientov využívajúcich systém. Tým je určené množstvo klientov potrebné pre vytrvalostný test.

3. **Testy s rastúcou veľkosťou správ**
   Stresový typ testu je možné nasimulovať aj pomocou rastúcej veľkosti generovaných správ pri fixnom počte klientov. Podobne ako v predchazajúcom prípade sa test skladá zo série testov, pričom sa postupne zväčšuje veľkosť správ. Test je takmer totožný s predchádzajúcim testom. Namiesto počtu klientov sa v tomto teste mení veľkosť prijatej správy. Dĺžka každého testu je päť minút. Zasielaných je päť druhov správ a každá z nich päťkrát pre každý testovaný nástroj. Počet klientov v každom teste je 50 a veľkosti správ sú: 5 znakov, 1024 znakov, 5120 znakov, 51200 znakov a 512000 znakov. Podobne ako v predchádzajúcom prípade spôsobí zvyšovanie záťaže zlyhanie systému. Na základe testu je možné určiť optimálnu veľkosť prenášanej správy pre vytrvalostný test.

4. **Základný JMS test so 100 klientmi**
   Tento test takmer identický, ako prvý test. Líši sa od neho množstvom užívateľov využívajúcich systém a protokolom používaným pre posielanie správ. Hlavným účelom testu je overiť schopnosť nástoroja pracovať s JMS protokolom. Samozrejmosťou je sledovanie a porovnanie výkonu nástrojov.

5. **Základný test so 100 klientmi**
   Test je identický s predchádzajúcim testom. Jediný rozdiel je v spôsobe zasielania správ, ktorý neprebieha pomocou JMS, ale pomocou HTTP protokolu. Úlohou tohto testu je poskytnúť testovacie dáta zo základných testov aj u tých nástrojov, ktoré nepodporujú protokol JMS. Vďaka tomu je možné porovnať výsledky základných testov u všetkých nástrojov.

6. **Vytrvalostný test**
   Dlhodobé monitorovanie systému je realizované pomocou vytrvalostných testov. Systém je vystavený záťaži generovanej sto klientmi po dobu jednej hodiny. Test sa skladá z dvoch častí. V prvej časti je testovaná služba použitá v predchádzajúcich testoch a v druhej časti upravená služba simulujúca únik pamäte. Druhá časť testu ukáže rastúcu spotrebu pamäte a v porovnaní s prvou časťou tak bude možné odhaliť vzniknutý únik pamäte.

7. **Test teoretickej priepustnosti nástroja**
   Teoretická priepustnosť nástroja určuje, ako rýchlo dokáže nástroj generovať záťaž. Vysoká priepustnosť nástroja umožňuje generovať vyššiu záťaž testovaného nástroja za využitia menšieho objemu zdrojov. Test prebieha generovaním maximálnej možnej záťaže na aplikáciu, ktorá na výstupe vracia vždy rovnaký reťazec. Aplikácia je schopná odpovedať na dotaz v čase kratšom, ako je čas potrebný na vygenerovanie dotazu. Celková priepustnosť systému je tak limitovaná samotným nástrojom.

5.4 Automatizácia testov

Pre účely testu bola vytvorená architektúra obsahujúca zložená z klientskej a serverovej časti. Architektúra je vytvorená tak, aby bolo možné spúšťať testy v Linuxových operačných systémoch.

5.4.1 Klient

Klient obsahuje adresáre so spúšťacími skriptami a testami, symbolické linky adresárov, testovacie nástroje a skripty spúšťajúce testy a upravujúce testy. Názov adresára obsahuje číslo a popis spúšťaného testu. V každom adresári sú umiestnené podadresáre Apache JMeter, Faban, Gatling a Perfcake, ktoré obsahujú testy pre daný nástroj a skript, ktorý riadi spúštanie testu. V adresároch druhého, tretieho a šiesteho testu sa nachádzajú podadresáre obsahujúce modifikácie daného testu, tak ako je to uvedené v ich definícii. Tieto testy navyše obsahujú skript, ktorý spúšťa jednotlivé modifikácie daných testov. Nasleduje popis skriptov, ktoré sa nachádzajú v koreňovom adresári.

- **change_runtime.sh**
  Slúži na nastavenie dĺžky testov. Prijíma dva prepínače: ”-b” a ”-t”. Prepínač -t prijíma ako argument číslo, ktoré reprezentuje dĺžku trvania vytrvalostných testov v minútach a argument prepínača -b reprezentuje dĺžku trvania zvyšných testov v sekundách. Implicitná hodnota pre prepínač -t je 60, čiže jedna hodina a pre prepínač -b 300, teda 5 minút. Tieto hodnoty sa použijú v prípade, ak je skript spustený bez argumentov. Skript sa automaticky volá v skripte run_all.sh, ktorým sa spúšťajú všetky testy. Tým sa zabezpečí korektné nastavenie času v testoch.
  Použitie: *./change_runtime.sh -b 10 -t 120*

- **change_server_and_port.sh**
  Skript nastavuje adresu servera, číslo portu pre webové služby a číslo JMS portu. Na vstupe berie tri prepínače: ”-s”, ”-p” a ”-m”. Argumentom prepínača -s je adresa servera, ktorá môže byť reprezentovaná IP adresou. IP adresa môže byť reprezentovaná číslom, pomocou ktorého je možné jednoznačne identifikovať server v počítačovej sieti alebo aliasom, ktorý je k danej IP adrese pridelený. Prepínač -p prijíma ako argument číslo portu, pomocou ktorého je možné komunikovať s webovými službami aplikácie. A nakoniec prepínač -m, ktorý nastavuje port pre JMS. V prípade, že nie sú pri spustení skriptu použité argumenty, použije sa pre server implicitná hodnota ”localhost”, pre port webových služieb hodnota ”8080” a port JMS sa nastaví na hodnotu ”4447”. Použité čísla portov sú východiskovými portmi JBoss AS servera pre dané komunikačné technológie.
  Použitie: *./change_server_and_port.sh -s 192.168.0.2 -p 18080 -p 14447*

- **open_scenarios.sh**
  Po spustení skriptu sa otvoria všetky testovacie scenáre v textovom editore gedit.

Skript často používam pri kontrole správneho nastavenia scenárov.
Použitie: *./open_scenarios.sh*

- **remove_all_logs.sh**
  Tento skript slúži na vymazanie všetkých záznamov a logovacích súborov, ktoré vznikli pri testovaní. Vygenerované záznamy a logovacie súbory majú značnú veľkosť, čo spôsobuje problémy pri kopírovaní testov.
  Použitie: *remove_all_logs.sh*

- **remove_big_logs.sh**
  Podobne ako v predchádzajúcom prípade odstraňuje skript súbory, ktoré vznikli pri testovaní. Na rozdiel od predchádzajúceho prípadu odstraňuje len najväčšie súbory. V prípade nástroja Apache JMeter sú to súbory s príponou ".jtl" a v prípade nástroja Gatling súbory "simulation.log". Tieto súbory obsahujú záznam pre každú prijatú odpoveď, čo je užitočné v prípade výskytu chyby, pretože je možné presne identifikovať kedy chyba nastala. Súbory neobsahujú informácie dôležité pre vyhodnotenie a porovnanie testov, preto je žiadúce ich zmazanie po úspešnom ukončení testov.
  Použitie: *remove_big_logs.sh*

- **restart_faban.sh**
  Nástroj Faban sa skladá z dvoch hlavných komponent: Faban Driver Framework a Faban Harness. Kým Faban Driver Framework slúži k vytváraniu testov a definovaniu testovacej záťaže, Faban Harness slúži k samotnému spúšťaniu a monitorovaniu testov. Komponenta beží na Apache serveri, ktorý je reštartovaný pred každým spustením Faban testu. Cieľom tohto reštartu je minimalizovanie vplyvu predchádzajúcich testov.
  Použitie: *restart_faban.sh*

- **run_all.sh**
  K spusteniu všetkých testov slúži tento skript, ktorý prijíma 7 prepínačov: "-s", "-p", "-m", "-t", "-b", "-u" a "-d". Pred samotným sputstením všetkých testov sú spustené skripty *change_runtime.sh* a *change_server_and_port.sh*, ktorým sú predané argumenty prepínačov pre nastavenie dĺžky trvania testov, adresy servera, čísla portu webových služieb a čísla portu pre JMS. Prepínač -u prijíma ako argument meno užívateľa, cez ktorého klient pomocou príkazu *ssh* reštartuje JBoss AS server, ktorý beží na inom stroji. Pokiaľ tento atribút nie je definovaný, skript skončí s chybou. Argument posledného prepínača reprezentuje adresár, v ktorom je umiestnený koreňový adresár obsahujúci server JBoss AS a skript potrebný na jeho obsluhu. V prípade, že adresár nie je definovaný, použije sa implicitná hodnota "/home/user", kde user je meno užívateľa nastavené prepínačom -u.
  Použitie: *run_all.sh -s 192.168.0.2 -p 18080 -m 14447 -b 10 -t 120 -u tomas -d /home/tomas/server*

5.4.2 Server

Na strane servera sa nachádza adresár obsahujúci server JBoss AS, skript potrebný k jeho reštartovaniu a agenti Apache JMeter a Perfcake, ktorí slúžia k monitorovaniu využitia pamäte na strane servera.

- **ssh_restart_jboss.sh**
  Skripty spúšťajúce testy na strane klienta reštartujú JBoss AS server pred každým spustením testu. K tomu dochádza spustením skriptu ssh_restart_jboss.sh príkazom ssh na strane klienta. Tento skript prijíma dva argumenty. Prvým argumentom je adresár, v ktorom je umiestnený koreňový adresár. Ako druhý argument je potrebné uviesť IP adresu servera. Adresa sa používa pri spustení servera a server pomocou nej komunikuje s klientom. Server sa vždy spúšťa na porte 8080.
  Použitie: *ssh user@192.168.0.2 "/user/dir/ssh_restart_jboss.sh /user/dir 192.168.0.2"*

Kapitola 6

Záver

6.1 Zhrnutie

Apache Jmeter

Plusy

+ Podpora pre IDE
+ Wiki a FAQ stránky
+ Je súčasťou ubuntu softvérového centra

Mínusy

- Chýbajúce informácie o ďalšom vývoji
- Miestami zastaralá dokumentácia

Perfcake

Plusy

+ Aktívny vývoj nových verzií
+ Prehľadná dokumentácia
+ Strmá krivka učenia

Mínusy

- Neexistuje grafické prostredie
- Potrebná základná znalosť značkovacých jazykov

Faban

Plusy

+ SPECS

Mínusy

- Chýbajúca podpora pre IDE
- Neaktívny vývoj
- Port 9999 používa Faban, aj JBoss AS
- Najaktuálnejšia verzia bola vydaná 26.9.2013
- Zložitá tvorba a upravovanie scenárov

Gatling

Plusy

+ Dokumentácia
+ Výsledky zobrazené v grafickom prostredí

Mínusy

- Chýbajúce informácie o ďalšom vývoji
- Úprava a tvorba scenárov vyžaduje znalosti jazyka Scala
- Neexistuje grafické prostredie

Literatúra

[1] W. Garside. Computing power: capacity on demand. www.computerweekly.com/feature/Computing-power-capacity-on-demand. Navštívené: 22.6.2014.

[2] J. Sochor. ÚVT MU zpravodaj. www.ics.muni.cz/bulletin/articles/61.html. Navštívené: 22.6.2014.

[3] Ian Molyneaux. *The Art of Application Performance Testing*. O'Reilly Media, Inc., Sebastopol, first edition edition, 2009.

[4] Apache Software Foundation. Apache jmeter. http://jmeter.apache.org/index.html. Navštívené: 19.12.2014.

[5] Faban. http://faban.org/about.html. Navštívené: 19.12.2014.

[6] Gatling. http://gatling.io/docs/2.1.1/. Navštívené: 19.12.2014.

Dodatok A

Obsah priloženého CD

Súčasťou tejto práce je aj CD obsahujúce všetky materiály, ktoré boli vytvorené v rámci písania práce. Adresárová štruktúra CD vyzerá nasledovne:

```
/
├── Bakalárska práca
│   ├── img ..... Adresár s obrázkami použitými v práci
│   ├── bibliografie.bib ..... Bibliografická databáza práce
│   ├── bp.tex ..... Zdrojový kód práce
│   └── bp.pdf ..... Hotová práca
├── klient.zip ..... Archív obsahujúci klientskú časť testov
│   ├── 1, 2, 3, 4, 5, 6, 7 ..... Symbolické odkazy k adresárom
│   ├── 1.Zakladny test s jednym klientom ..... Adresár s testom
│   ├── 2.Rastuce mnozstvo klientov ..... Adresár s testom
│   ├── 3.Rastuca velkost spravy ..... Adresár s testom
│   ├── 4.JMS zakladny test so 100 klientmi ..... Adresár s testom
│   ├── 5.Zakladny test so 100 klientmi ..... Adresár s testom
│   ├── 6.Vytrvalostny test ..... Adresár s testom
│   ├── 7.Test teoretickej priepustnosti nastrojov ... Adresár s testom
│   └── Nastroje ..... Adresár obsahujúci nástroje
│       ├── apache-jmeter-2.11 ..... Nástroj Apache JMeter
│       ├── faban ..... Nástroj Faban
│       ├── gatling-charts-highcharts-2.0.1 ..... Nástroj Gatling
│       ├── change_runtime.sh ..... Skript meniaci čas testov
│       ├── change_server_and_port.sh ..... Skript meniaci server a port
│       ├── open_scenarios.sh ..... Skript otvárajúci scenáre
│       ├── remove_all_logs.sh ..... Skript odstraňujúci všetky logy a reporty
│       ├── remove_big_logs.sh ..... Skript odstraňujúci velké logy a reporty
│       ├── restart_faban.sh ..... Skript reštartujúci Faban server
│       └── run_all.sh ..... Skript spúšťajúci testy
└── server.zip ..... Archív obsahujúci serverovú časť testov
    ├── Nastroje ..... Adresár obsahujúci JBoss AS server
    │   └── jboss-as-7.1.1.Final ..... JBoss AS server
    └── ssh_restart_jboss.sh ..... Skript reštartujúci JBoss AS server
```