Zesium mobile d.o.o
Mičurinova 8
21000 Novi Sad

Anja Bešić
Branislav Manojlović

Praktikum

Alati za testiranje veb aplikacija

JMeter

Novi Sad, 2018

Sadržaj:

1. Uvod

Testiranje predstavlja pokušaj da se pronađu greške u softveru koji je napravljen. Softver je implementiran prema korisničkim zahtevima kojima se rešava neki realni problem ili se kreira neka korisna funkcionalnost koja predstavlja nešto što je potrebno krajnjim korisnicima. Kada se implementira, softver može u većoj ili manjoj meri da odgovara originalnim zahtevima prema kojima je i napravljen.
Svako ponašanje softvera koje se ne slaže sa originalnim zahtevima predstavlja grešku koju je potrebno identifikovati i otkloniti. U užem smislu, testiranje predstavlja upravo proveru da li je određeni softver u potpunosti implementiran prema originalnim korisničkim zahtevima. U širem smislu testiranje predstavlja sistem kontrole kvaliteta (QA – *Quality Assurance)* kojim se ne proverava samo softver već i sve njegove prateće komponente i karakteristike.

Kvalitet softvera se može definisati na različite načine kao na usaglašenost sa zahtevima i potrebama korisnika. Dobri atributi proizvoda kao što su brzina rada, malo zauzeće memorije i prostora na disku, brzina pokretanja, lakoća održavanja i promena u softveru, kao i prenošenja na druge platforme.
Dobar softver je propracen kvalitetnom dokumentacijom, jasno definisanim zahtevima, definisanim dizajnom dizajna, uputstavima za upotrebu i drugim pratećim dokumenatima kojima se opisuje softver. Usaglašenost sa standardima podrazumeva usaglašenost sa organizacionim standardnima pisanja programskog koda, poštovanje opštih standarda (ISO), usaglašenost sa zakonima i slično.

Za svaki softver posebno je vazno da se proveri rad u ekstremnim uslovima sa ogromnim količinama podataka, slabim vezama, ograničenim resursima koji su na raspolaganju i slično. Ovde je posebna paznja usmerena na testiranje performansi aplikacija.

Namena pratikuma je da objasni osnovne pojmove iz oblasti testiranja performansi veb aplikacija i da opiše metode testiranja pomocu alat za testiranj preformansi veb aplikacije JMeter i da opiše način njegove upotrebe.

Poglavlje broj dva, opisuje pojam testiranja performansi, definise osnovne tipove testiranja performansi. Takodje u ovom poglavlju su navedeni i opisani osnovni problemi koji se javljaju u procesu testiranja performansi. Prikazan je digjagram koji opisuje proces testiranja performansi i opisani su koraci procesu testiranja performansi.

Poglavlje broj tri opisuje JMeter softverski alat za testiranje performansi veb aplikacija. Navedene su i opisane osnovne karakteristike i osnovne komponente ovog alata.

Poglavlje broj cetiri, daje pregled osnovnih koraka u kreiranju JMeter testova.

Poglavlje broj pet sadrzi nekoliko primera upotrebe JMeter alata za testiranje veb aplikacije u rezlicitim test scenarijima.

2. Testiranje performansi

Testiranje performansi (Performance testing) je tip testiranja koje treba da proveri odnosno verifikuje da će se softverske aplikacije ponašati dobro pod očekivanim radnim opterećenjem.

Cilj testiranja performansi je da se provere:
1. *Brzina* – provera da li aplikacija odgovara brzo;
2. *Skalabilnost* – utvrđuje koji je maksimalan broj korisnika koji aplikacija može da podnese u isto vreme;
3. *Stabilnost* – utvrđuje da li aplikacija stabilno radi pod velikim opterećenjem.

Razlozi za testiranje performansi leže u tome da treba da utvrdi šta treba da bude poboljšano na aplikaciji pre nego što aplikacija ode na tržište. Ovo testiranje mora da utvdi da li aplikacija zadovoljava zahteve klijenta u pogledu brzine, skalabilnosti, stabilnosti pod planiranim opterećenjem.

2.1.Tipovi testiranja preformansi

Osnovni tipovi testiranja performansi su:

- *Load testing* (testiranje učitavanja) - proverava sposobnost aplikacije da se učita pod pritiskom očekivanog broja korisnika. Ovde je potrebno utvrditi koja su to "uska grla" u pogledu maksimalnog broja korisnika.
- *Stress testing* – testiranje aplikacije pod ekstremnim opterećenjem kako bi se utvrdilo kako se aplikacija nosi s visokim saobraćajem i obradom velike količine podataka. velikim radnim opterećenjem tokom dugog vremenskog perioda.
- *Endurance testing* (testiranje izdržljivosti) – vrši se da bi se proverilo da li softver može da se nosi s velikim radnim opterećenjem tokom dugog vremenskog perioda.
- *Spike testing* – proverava reakciju aplikacije usled naglog povećanja opterećenja koje generišu korisnici.
- *Volume testing* (testiranje opterećenja) – testiranje u kome se veliki broj podataka "pohrani" u bazu podataka aplikacije i tada se posmatra ponašanje čitavog sistema u radu s velikom količinom podataka.
- *Scalability testing* – proverava kako se efikasnost aplikacije ponaša u slučaju porasta kako broja korisnika tako i količine njihovih podataka. Ovo je bitno zbog eventualnog širenja sistema aplikacije u budućnosti.

2.2.Problemi u testiranju performansi

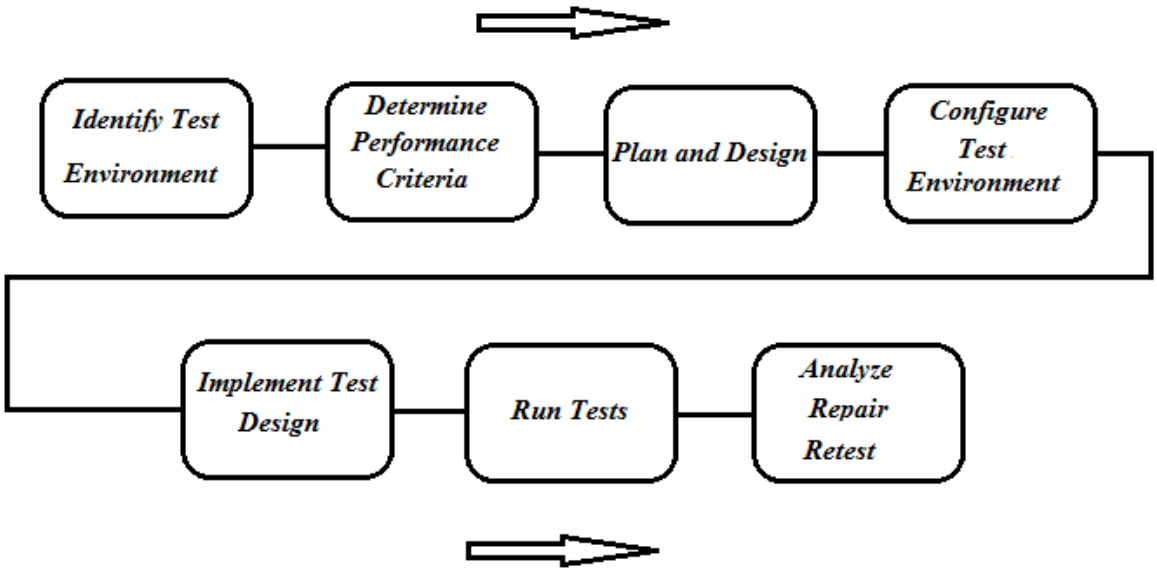
Problemi koji se najčešće pojavljuju kada su u pitanju performanse se tiču brzine, vremena odziva, vremena i dužine učitavanja. Zato je testiranje performansi bitno, jer ono treba da osigura da će aplikacija raditi dovoljno brzo.

Najčešći problemi su:

- *Long load time* – dugo vreme učitavanja. Vreme učitavanja je inicijalno vreme potrebno da se aplikacija startuje. Ono treba da bude što je kraće moguće (<5s).
- *Bad response time* – vreme odgovora je vreme koje protekne od trenutka kada korisnik unese neke svoje ulazne podatke u aplikaciju pa do trenutka kada aplikacija vrati odgovor na njih. Ono takođe treba da bude što kraće.
- *Bad scalability* – za aplikaciju se kaže da ima lošu skalabilnost kada ne može da podnese veliki broj korisnika ili kada ne može da usluži dovoljno širok broj korisnika.
- *Bottleneck* – uska grla su ograničenja u sistemu koja degradiraju sveukupne performanse sistema. Ona najčešće nastaju usled loše urađenih delova koda u aplikaciji. Neki od čestih uskih grla u sistemima su: opterećenje procesora (CPU utilization), opterećenje memorije (memory utilization), opterećenje mreže (network utilization), ograničenja operativnog sistema, upotrebljenost hard diska.

2.3.Proces testiranja performansi

Sastoji se od sledećih koraka:



Slika 1. Dijagram procesa testiranja performansi

Koraci su:

1. *Identifikovanje test okruženja* – potrebno je utvrditi kakvo je fizičko okruženje u kome će se izvršavati testovi i koji su test alati dostupni. Potrebno je prepoznati karakteristike hardvera, softvera i mreže koje će se koristiti pre nego šro se počne s procesom testiranja.
2. *Definisati kriterijume performansi* – potrebno je definisati koji su to ciljevi koje treba dostići, odnosno koje su to vrednosti u pogledu performansi koje aplikacija treba da ispuni kako bi zadovoljila klijenta. Često je u pogledu kriterijuma moguće uporediti svoju aplikaciju koju testiramo sa istom takvom ili sličnom koja se nalazi na tržištu.
3. *Isplanirati i dizajnirati testove* – potrebno je shvatiti koliko će upotreba aplikacije varirati od strane krajnjih korisnika, identifikovati koji su to ključni test scenariji i test slučajevi koje treba napraviti kako bismo obuhvatili sve moguće slučajeve korišćenja. Treba isplanirati šta će i kakvi će biti test podaci, zatim utvrditi kakva metrika će biti sakupljena i upotrebljena.
4. *Konfigurisanje test okruženja* – potrebno je pripremiti test okruženje u kome će se raditi testovi, instalirati potrebne alate i setovati kapacitet i protok ostalih resursa (baza, mreža…)
5. *Implementiranje test dizajna* – potrebno je napraviti testove u skladu s test dizajnom. To znači napisati skripte ili setovati parametre u alatima kao što su Jmeter, Webserver stress tool 8…
6. *Izvršavanje testova* – podrazumeva konkretno izvršavanje testova u skladu sa test planom i potrebno je pratiti i beležiti izlazne rezultate.
7. *Analiza, Dorada i Retestiranje* – potrebno je prikupiti, srediti i analizirati sve ostvarene rezultate, izvršiti neophodne korekcije na sistemu ili aplikaciji a nakon toga retestirati.

2.4.Parametri u testiranju preformansi

Osnovni parametri koji se prate tokom testiranje performansi:

- Upotreba procesora (*Processor usage)* – je količina vremena koje procesor utroši izvršavanjem zadataka.
- Upotreba memorije (*Memory usage*) – količina fizičke memorije raspoložive procesima aplikacije, tj. koliko se memorije zauzme dok aplikacija izvršava zadatke.
- Vreme rada hardvera (**Disk time**) – vreme koliko je disk zauzet dok izvršava read & write zahteve.
- Protok (*Bandwith*) – pokazuje broj bita po sekundi (bit/sec) koje se koriste od strane mrežnog interfejsa.
- Privatni bajti (*Private bytes*) – broj bajta koje je proces koji se izvršava dodelio sebi i ne može podeliti među ostalim procesima. Ovde se obično nađe memory leak u aplikacijama.
- Dodeljena memorija (*Committed memory*) – količina upotrebljene virtuelne memorije.

- Broj memorijskih stranica po sekundi (*Memory pages/second*) – broj strana koje su učitane istovremeno na disk i koje su pri tome odrađene od strane procesora.
- Broj grešaka po sekundi (*Page faults/second*) – prosečan broj grešaka u procesiranju stranice od strane procesora.
- Prekid procesora po sekundi (*CPU interrupts per second*) – prosečan broj hardverskih prekida u radu procesora u primanju i slanju zahteva po sekundi.
- Dužina upita po sekundi (*Disk queue length*) – prosečan brij upita za čitanje I pisanje na određeni disk tokom odabranog vremenskog intervala.
- Dužina izlaznog mrežnog upita (*Network output queue length*) – dužina paketa podataka koji se šalje kroz upit u mreži; sve više od 2 po upitu znači da može doći do pojave "uskog grla".
- Broj mrežnih bajta po sekundi (*Network bytes total per second*) – tempo po kome su bajtovi poslati i primljeni kroz mrežni interfekjs uključujući i frame characters.
- Vreme odgovora (*Reponse time*) – vreme koje protekne od trenutka kada korisnik unese zahtev do trenutka kada se prvi karakter iz odgovora primi.
- Propusnost (*Throughput*) – tempo po kom računar ili mreža prima zahteve po sekundi.
- Broj zahteva za konekcijom (*Amount of cnnection pooling*) – broj korisničkih zahteva koji moraju biti ispunjeni uspostavljanjem konekcije.
- Maksimalan broj aktivnih sesija (*Maximum active sessions*) – maksimalan broj sesija koje mogu biti aktivne u isto vreme.
- Razmera pogodaka (*Hit ratios*) – broj SQL upita hendlovanih od strane keširanih podataka umesto od strane skupih ulazno/izlaznih operacija. Ovo je mesto gde se stvaraju "uska grla".
- Broj pogodaka po sekundi (*Hits per second*) – broj pogodaka na serveru tokom svake sekunde load testa.
- Povratni segment (*Rollback segment*) – količina podataka koja može ponovo da se pozove i iščita u bilo kom trenutku.
- Zaključavanje baza podataka (*Database locks*) – praćenje zaključavanja baza podataka I tabela u njima.
- Najduža čekanja (*Top waits*) – se prate kako bi se utvrdilo koja sve vremena čekanja možemo skratiti kada su u pitanju pozivanja podataka iz memorije.
- Broj aktivnih niti (*Thread Counts*) – kvalitet i stanje aplikacije se može izmeriti prema broju niti koje se moraju aktivirati u isto vreme radi izvršavanja zadataka.
- Skupljanje smeća (*Garbage Collection*) – ima veze sa vraćanjem neupotrebljene memorije sistemu. Radi bolje efikasnosti aplikacije potrebno je pratiti količinu smeća koja nastaje usled neupotrebljenih podataka.

2.5.Alati u testiranju performansi

Najpopularniji alati za testiranje performansi su: *Neoload, WebLoad, LoadView Testing, HP LoadRunner, JMeter*.

3. JMeter alat za testiranje

JMeter (Apache JMeter) je open-source softver napisan na programskom jeziku Java i namenjen je za testiranje performansi veb aplikacija.

Koristi se za testiranje performansi statičkih i dinamičkih resursa veb aplikacija. Takođe, koristi se i za simuliranje velikog opterećenja servera, grupe servera ili mreže, kako bi se analizirale njihove performanse pod različitim tipovima opterećenja.

3.1. Karakteristike JMeter alata.

Osnovne karakteristike JMeter-a su:

- omogućava testiranje performansi i opterećenja različitih aplikacija, servera, protokola: veb (HTTP, HTTPS, ...), SOAP/REST veb servisa, FTP, Baza podataka, LDAP, TCP, Java objekata, ... i slično.
- poseduje Test IDE koje omogućava snimanje test plana, build-ovanje i debagovanje,
- omogućava pokretanje testova i bez GUI-a, kroz CMD (windows) ili Terminal (linux)
- pruža kompletne dinamičke HTML izveštaje za prikazivanje i analizu rezultata,
- izvlačenje i testiranje podataka koji se razmenjuju u veb aplikacijama u raznim formatima odgovora (*response*), kao što su HTML, JSON, XML, ... i slično.
- omogućava *multi-threading*, tj. izvršavanje testova simultano sa više različitih korisničkih grupa,
- čuvanje i analizu test rezultata offline,
- omogućava korišćenje raznih plugin-ova i open-source biblioteka.

3.2. Osnovne komponente JMeter alata

Test Plan je kontejner koji sadrži sve elemente i komponente koje su potrebne da se izvrši test.

WorkBench je komponenta u kojoj privremeno čuvamo svoje test elemente, da bismo ih kasnije kada nam ne trebaju mogli odatle da obrišemo.

Thread Group je komponenta koja predstavlja korisnike. Ona sadrži sledeće:

- *Action to be taken after a sample error* – to je izbor aktivnosti koje se izvršavaju nakon što Jmeter pronađe grešku. Postoje: Continue (Nastavi), Start Next Thread Group (Pokreni novu iteraciju), Stop Thread (Zaustavi određenog korisnika), Stop Test (Zaustavi test), Stop Test Now (Zaustavi test odmah).

- *Thread Properties* – određuje se broj korisnika, vreme potrebno da se svi korisnici pokrenu i pristupe stranici i broj ponavljanja njihovog pristupanja. Ako u Ramp Up Period setujemo 20s za 10 korisnika, to znači da će na svake 2s biti dodat 1 korisnik.
- *Scheduler Configuration* – polje za setovanje trajanja testa, eventualnog pauziranja testa, početnog i krajnjeg vremena testa.

HTTP request je komponenta za slanje zahteva na stranici koju testiramo. Potrebno je uneti URL koji gađamo našim zahtevom. Pregled i tumačenje ostvarenih rezultata u vrši se u **Listener** komponentama.

3.3.Assertions komponente

Assertions komponente predstavljaju komponente za proveru odgovora (response-a). Na primer, proveravaju da li se kao response code dobio broj 200 ili da smo dobili neku predefinisanu poruku kao response message, npr. OK i sl. Možemo testirati i format odgovora kao i trajanje odgovora u bajtima i sl. Sve ove provere se zovu Assertions (tvrdnja, navod).

Assertions se dodaju pomoću:

Thread Group → Add → Assertion → odabrati onu koja je potrebna.

Najčešće korišćene Assertions komponente su:

- Response Assertion,
- Duration Assertion,
- Size Assertion,
- HTML Assertion,
- XML Assertion,
- Xpath Assertion.

➢ **Response Assertion** – je najčešće korišćena assertion komponenta. Kada se Response Assertion setuje, potrebno je polju *Apply* odabrati na šta se Assertion želi primeniti:

  - Main sample and sub-samples;
  - Main sample only;
  - Sub-sample only;
  - Jmeter variable.

  U polju *Response Field to Test* bira se koje polje u response-u želimo da testiramo, tj. koji nam podatak treba.
  Možemo odabrati:
  - Text Response;
  - Document (text);
  - URL sampled;

- Response code;
- Response Headers;
- Ignore Status.

Takođe, treba da odaberemo obrazac po kojem će se vršiti poređenje.

Obrasci su:

- Contains – da li response sadrži navedenu reč, kôd ili tekst.
- Matches – da li se response podudara sa navedenom rečju, kodom ili tekstom.
- Equals – da li je response jednak navedenoj reči, kodu ili tekstu.
- Substring – da li je response podskup reči, koda ili teksta.
- Not – da li je odgovor različit od navedene reči, koda, teksta.
- Or - ????

U polju Patterns to test se piše koji šablon (pattern) želimo da dobijemo u odgovoru. Npr, 200.

Princip rada: Komponenta Response Assertion proverava da li u odgovoru od servera dobija response code 200, i ukoliko dobija – test prolazi (passed), a u supritnom test pada (failed). Ona nam omogućava da se prikaže tačno kakav je odgovor dobijen i može direktno da ukaže na grešku.

- **Duration assertion** je komponenta koja omogućava proveru vremena trajanja odgovora (response-a).
  Ova vrednost se poredi s vrednostima u koloni *Sample time (ms)* u *View Results in Table* komponenti.
  U polju *Duration in miliseconds* napišemo koliko trajanje odgovora očekujemo i svaki response koji bude veći od njega smatraće se failed testom. Dakle, u ovo polje se upisuje gornja vrednost (maksimum) koju očekujemo za trajanje odgovora od servera.
  Bolje opisane rezultate za ovu Assertion se može pogledati u *Assertion Results* komponenti.

- **Size Assertion** služi za proveru veličine odgovora u bajtima. Ova vrednost se poredi sa vrednostima iz kolone *Bytes* iz *View Results in Table* komponente. Na Size Assertion komponenti u polju *Size in Bytes* se unese koliko bajta želimo da poredimo, tj. sa kojom vrednošću poredimo veličinu odgovora. A na polju *Type of Comparison* biramo opcije: =, !=, <, >, <=, >=.

- **HTML Assertion** služi za proveru formata response-a, tj. da li je odgovor bio u željenom formatu. Bira se format: HTML, XHTML, XML. U polju *Error Threshold* se navodi neki broj koji treba da bude graničan slučaj, tj. ako ima više grešaka u HTML odgovoru od servera smatra se da je test pao. Isti princip važi i za *Warning Threshold* polje. Ako želimo sve greške iz HTML Assertion komponente da sačuvamo, potrebno je u polju *Write JTidy report to file* navedemo direktnu putanju do njega, npr. C:\Users\Bane\Desktop\dokumenti_izvrštaj.txtŠtiklira se *Errors only* opcija. Kasnije, kada se otvori aplikacija vidi se da je sve okej upisano.

- **XML Assertion** služi za proveru rezultata, da li su dobijeni u odgovarajućem XML formatu.

- **XPath Assertion** se koristi za proveru API-ja (biće posebno objašnjeno). Bitno je da se XPath polje napiše koji XPath očekujemo kao odgovor. Jmeter proverava da li ćemo ga dobiti.

3.4.Listener komponente

Listener-i su komponente koje prikupljaju i prikazuju rezultate i metriku svih izvršenih testova.

Metrika koja se prikuplja i analizira pomoću Listener-a, obuhvata:

- *Latency* je vreme do prvog bajta poslatog kao odgovor od servera. Ako npr. imamo opseg vremena od 0ms do 2000ms i ako pretpostavimo da je naš zahtev trajao od 0ms do 1000ms, a recimo da smo počeli da dobijamo odgovor nakon 1010ms i on je trajao do 2010ms. Ovo znači da nam je Latency vreme bilo 1010ms jer smo baš tada počeli da dobijamo prve bajte odgovora od servera.
- Vrednost 2010ms je *Response time (ms)*, odnosno to je vreme koje protekne dok u potpunosti ne stigne odgovor do korisnika.
- *Connect Time (ms)* je vreme koje je bilo potrebno da naš zahtev (request) dođe do servera , tj. da se poveže sa serverom. Na osnovu ovoga možemo optimizovati aplikaciju.

Neke od najčešće korišćenih Listener komponenata su:

- **Aggregate report** je komponenta koja omogućava prikaz rezultata i metrike u jednoj liniji za svaki HTTP request posebno i pri tome daje prosečne vrednosti metrike. Rezultati su:
  - Samples – broj uzoraka;
  - Average – prosečno vreme trajanja request-a;
  - Median – srednja vrednost trajanja request-a. To znači da je 50% request-ova trajalo više od ove vrednosti a 50% request-ova je trajalo kraće od ove vrednosti.
  - 90% line, 95% line, 99% line – oznaka za maksimalno vreme trajanja request-ova;
  - Min i Max – najkraće i najduže vreme trajanja request-ova;
  - Error % - procenat grešaka koje su se pojavile;
  - Throughput – broj request-ova po sekundi;
  - KB/s – veličina odgovora po sekundi.

- **Graph Results** je kompomenta koja grafički prikazuje metriku koja se dobija u toku testiranja.
  Podaci koji se prikazuju:
  - Data – veličina podataka;

- Average – prosečna veličina podataka;
- Median – srednja vrednost;
- Deviation – odstupanje;
- Throughput – broj zahteva po sekundi.

➢ **View Results in Table**
- *Sample Time (ms)* – je vreme potrebno za upućivanje zahteva (request-a) ka serveru.
- *Status* – može biti Passed (npr. kada je response code = 200) ili Failed (npr. response code = 500).
- *Bytes* – veličina vraćenog odgovora (response-a) u bajtima.
- *Sent bytes* – veličina request-a u bajtovima.
- *Latency* – vreme koje je protekne do pojavljivanja prvog bajta poslatog od servera kao odgovor.

➢ **View Results Tree –** ako se klikne na jedan od uzoraka (sample-ova) u Results Tree, na desnoj strani radne površine se pregledaju sledeći rezultati:
- *Sampler result* – sadrži opis tipova zadataka koji su poslati (data type), kod odgovora (response code), heder odgovora (response headers), oznaku servera, datum i vreme, tip sadržava (content type) i sve one podatke iz prethodno opisane tabele.
- *Request* – detalji upućenog zahteva obično sadrže koja metoda je upotrebljena u zahtevu (GET/POST), koji su podaci traženi u zahtevu (GET data:), poslate kukije (cookies), zahtevane hedere (request headers), koja vrsta konekcije je zahtevana (Connection:), koji je host koji se gađa (HOST – mora da se slaže s onim što je navedeno u Web server sekciji na HTTP Request komponenti).
- *Response data* – podaci dobijeni u odgovoru sa servera. Ovo može da obuhvati sve moguće podatke koje server može poslati a nalaze se na web stranici URL-a koji gađamo. Tip podataka u odgovoru se može selektovati i može biti: text, HTML, CSS/Jquery, JSON, XML… i drugi.

➢ **Summary Report** je sličan Aggregate Report-u. Za svaki listener važi da ga možemo upotrebiti da upišemo rezultate u csv i txt fajlove, samo u Search polju pomoću Browse dugmeta upišemo adresu do fajla na računaru.

➢ **Simple Data Writer** je komponenta koja se koristi za generisanje izveštaja testa. Klikom na dugme Configure moguće je odabrati koje sve podatke želimo sačuvati u fajlovima. Preporučuje se da ovaj listener koristimo uvek, dok ostale ne.

Napomena: Listeneri zauzimaju dosta resursa mašini pa ih treba koristiti samo pri setovanju testa a onda kada testiranje počne, treba ih disable-ovati.

Napomena: Najbolje je koristiti Simple Data Writer za čuvanje rezultata u csv fajlu i na taj način ne treba koristiti ostale listener-e koji troše resurse mašine koja vrši testiranje.

3.5.Upotreba JMeter Plugin menadžera

Za instalaciju i uklanjanje Plugin-a u JMeter-u, koristimo JMeter Plugin Menadžer. Njega je potrebno dodati u JMeter GUI u sledećim koracima:

- Instalaciju skidamo sa: https://jmeter-plugins.org/wiki/PluginsManager/. Odavde skidamo JAR fajl, iz pasusa pod naslovom *Installation and Usage*.
- Sledeće, idemo u folder u kome se nalazi JMeter: jmeter → apache-jmeter-3.0 → lib → ext. Ovde u ext folder dodamo jar fajl koji smo skinuli.
- Nakon što smo dodali jar fajl, restartujemo JMeter, da bi promene bile prihvaćene.

Instalirani Plugin Manager, možemo pronaći na JMeter GUI na: Options → Plugins Manager. Dodavanje novih plugin-ova vršim klikom na Available Plugins → odaberem koji hoćemo plugin da instaliramo i kliknemo na dugme *Apply Changes and Restart JMeter*.

Plugin Menadžer nam omogućava:

- Instaliranje novih plugin-ova,
- Uklanjanje starih plugin-ova,
- Nadogradnju (Upgrade) postojećih plugin-ova,
- Dobijanje informacija o plugin-ovima koji nas interesuju.

3.6.Rad sa funkcijama i varijablama

Funkcije se u JMeter-u definišu kao bilo koji metod koji se može nalaziti u polju u bilo kom elementu test plana.

Sintaksa funkcije izgleda ovako:

${__imeFunkcije} – funkcija bez argumenata

${__imeFunkcije(varijabla1, varijabla2,…)} – funkcija sa argumentima

Varijabla u JMeter-u je kontejner koji čuva vrednosti na koju se može referencirati bilo koji element u niti. Varijable su uvek u lokalnoj niti.

Sintaksa varijable je:

${imeVarijable}

Napomena: Funkcije i varijable su case-sensitive i uvek se za pisanje njihovg imena treba koristiti „Camel Case notacija.

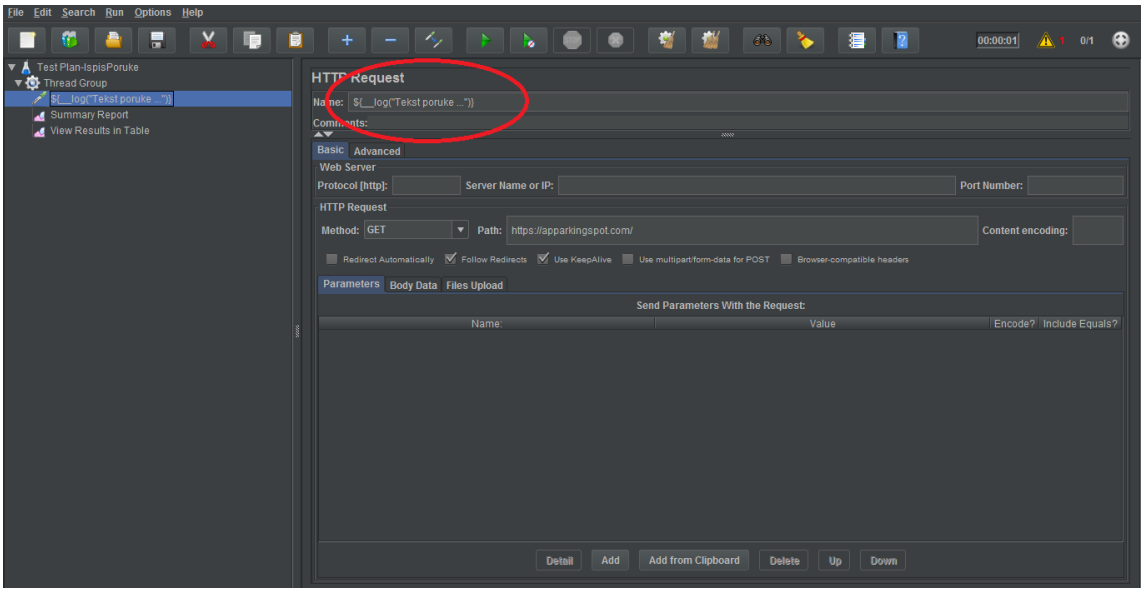
Postoji više funkcija u JMeteru koje možemo upotrebiti. Sve one se mogu pronaći i pregledati klikom na: **Options → Function Helper Dialog**. Ovde u polju Choose a function trazimo funkcije koje nam trebaju.

Primer 1: Primena funkcije za ispisivanje poruke/teksta.

Resenje: Funkcija izgleda ovako: ${__log("Tekst poruke")}

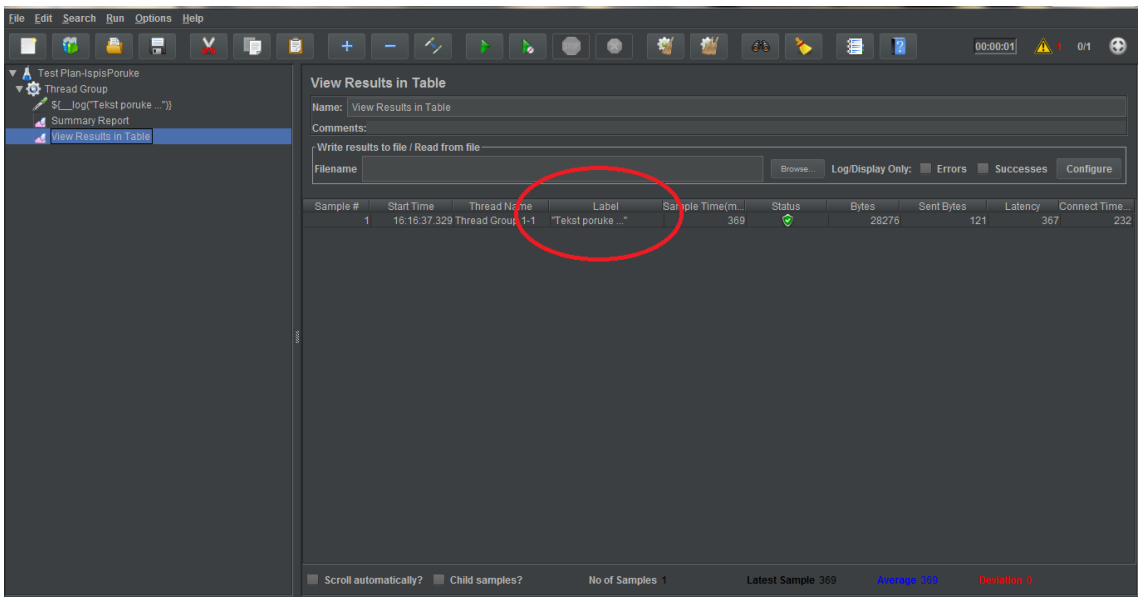
Kreiranje testa:

1. U test dodamo Thread Group-u, HTTP Request,
2. Dodamo Listenere: Summary Report i View Reults in Table,
3. U Path unesemo URL veb aplikacije koju testiramo,
4. Sada HTTP Request u polju *Name* preimenujem u ${__log("Tekst poruke ...")},



Slika 2. Izgled testa za ispis teksta

5. Pokrenemo test,
6. U View Results in table komponenti a u koloni Label dobijamo ispisan tekst koji smo naveli u HTTP Request komponenti.



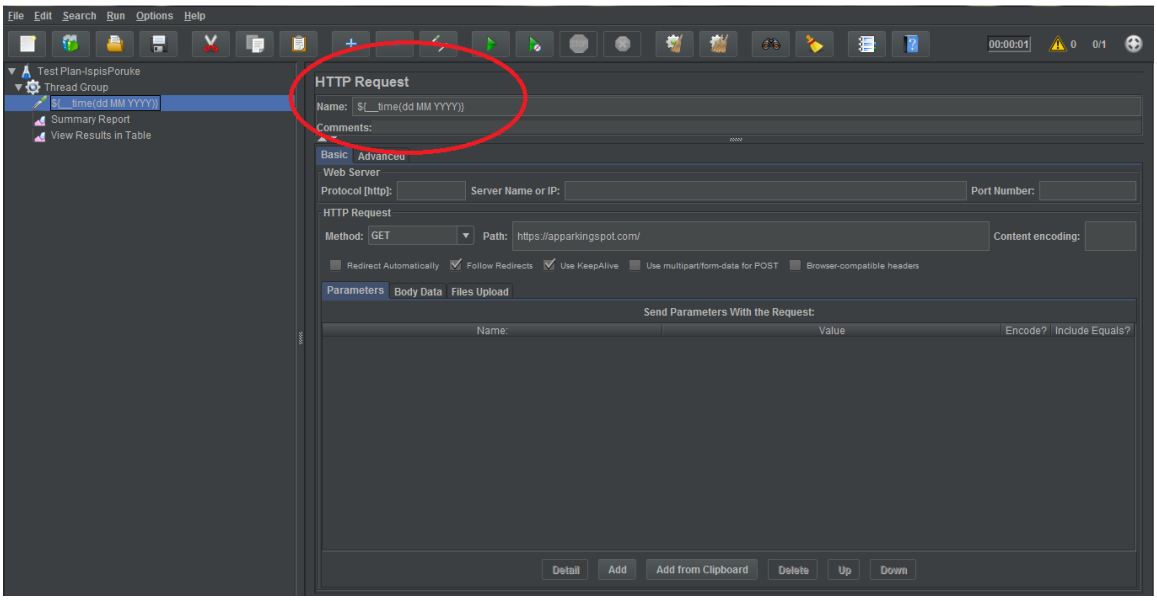
Slika 3. Rezultat testa za ispis teksta

Primer 2: Primena funkcije za ispisivanje datuma.

Resenje: Funkcija izgleda ovako: ${__time(dd MM YYYY)}

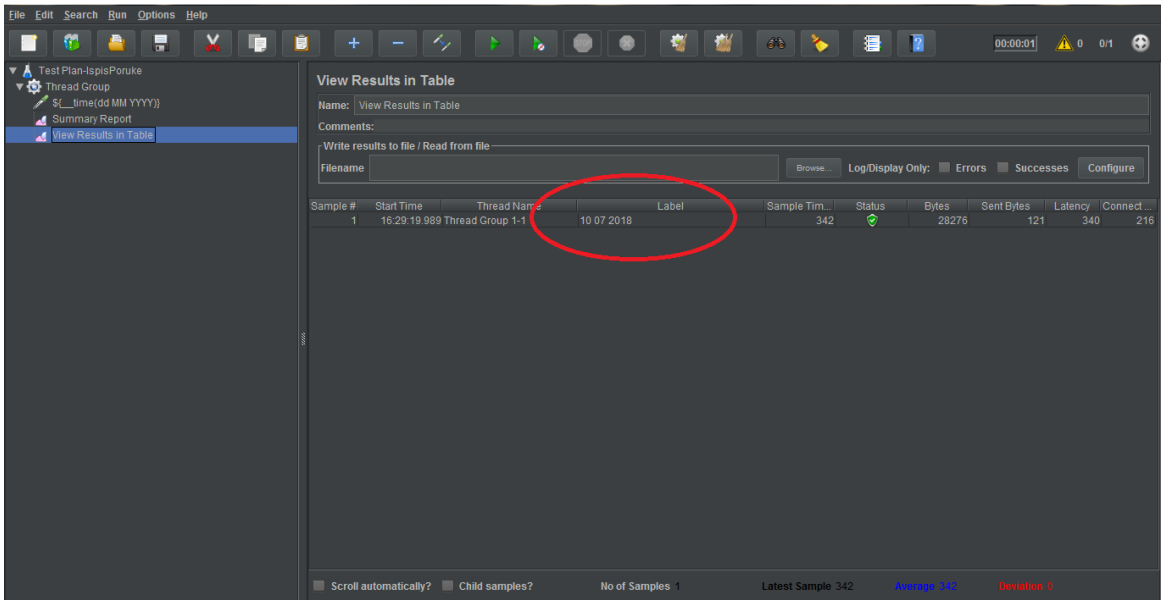
Kreiranje testa:

1. U test dodamo Thread Group-u, HTTP Request,
2. Dodamo Listenere: Summary Report i View Reults in Table,
3. U Path unesemo URL veb aplikacije koju testiramo,
4. Sada HTTP Request u polju *Name* preimenujem u ${__time(dd MM YYYY)},



Slika 4. Izgled testa za ispis datuma

5. Pokrenemo test,
6. U View Results in table komponenti a u koloni Label dobijamo ispisan datum koji smo naveli u HTTP Request komponenti.



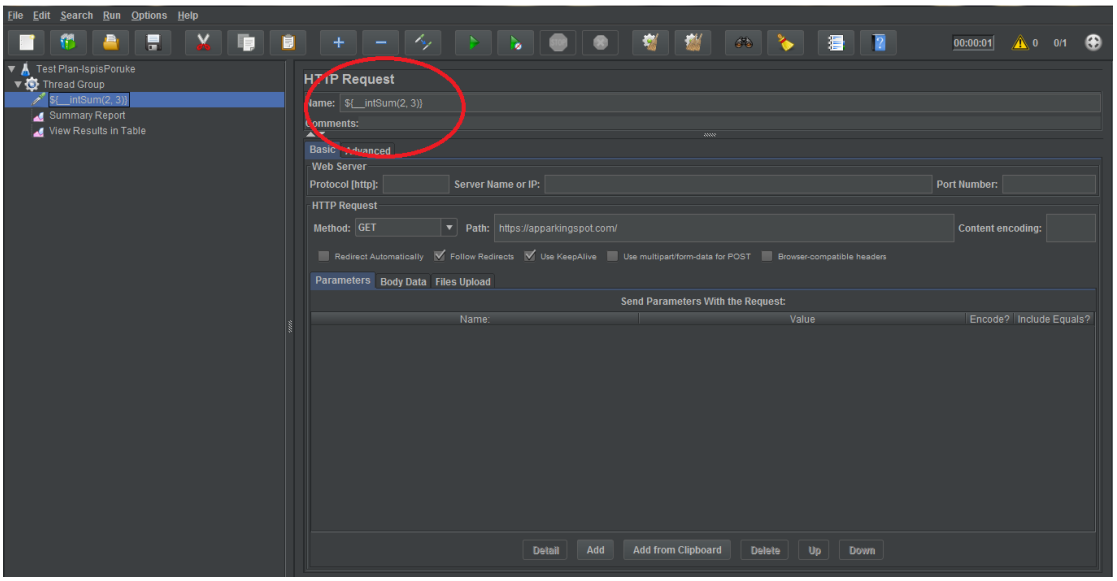
Slika 5. Rezultat testa za ispis datuma

Primer 3: Primena funkcije za izracunavanje i ispisivanje sume.

Resenje: Funkcija izgleda ovako: ${__intSum(2, 3)}

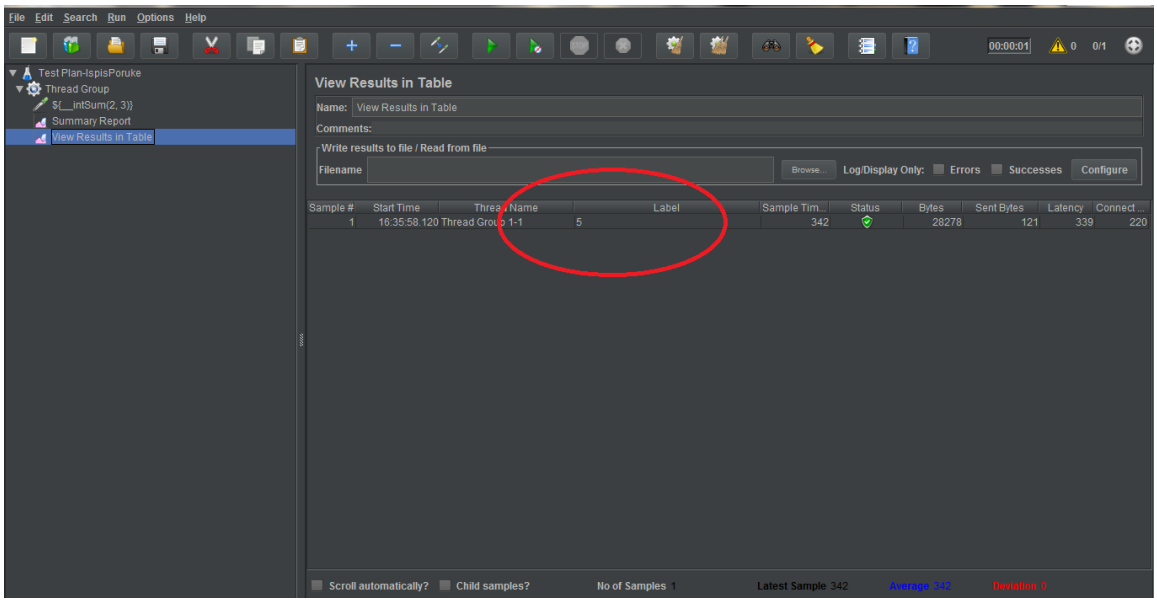
Kreiranje testa:

1. U test dodamo Thread Group-u, HTTP Request,
2. Dodamo Listenere: Summary Report i View Reults in Table,
3. U Path unesemo URL veb aplikacije koju testiramo,
4. Sada HTTP Request u polju *Name* preimenujem u ${__intSum(2, 3)},



Slika 6. Izgled testa za izracunavanje i ispis sume

5. Pokrenemo test,
6. U View Results in table komponenti a u koloni Label dobijamo ispisanu sumu koju smo naveli i izracunali u HTTP Request komponenti.



Slika 7. Rezultat testa za izracunavanje i ispis sume

3.7.Rad sa tajmerima

Tajmeri se najviše koriste da bi dočarali realnu upotrebu veb aplikacije prilikom testiranja. Oni omogućavaju:

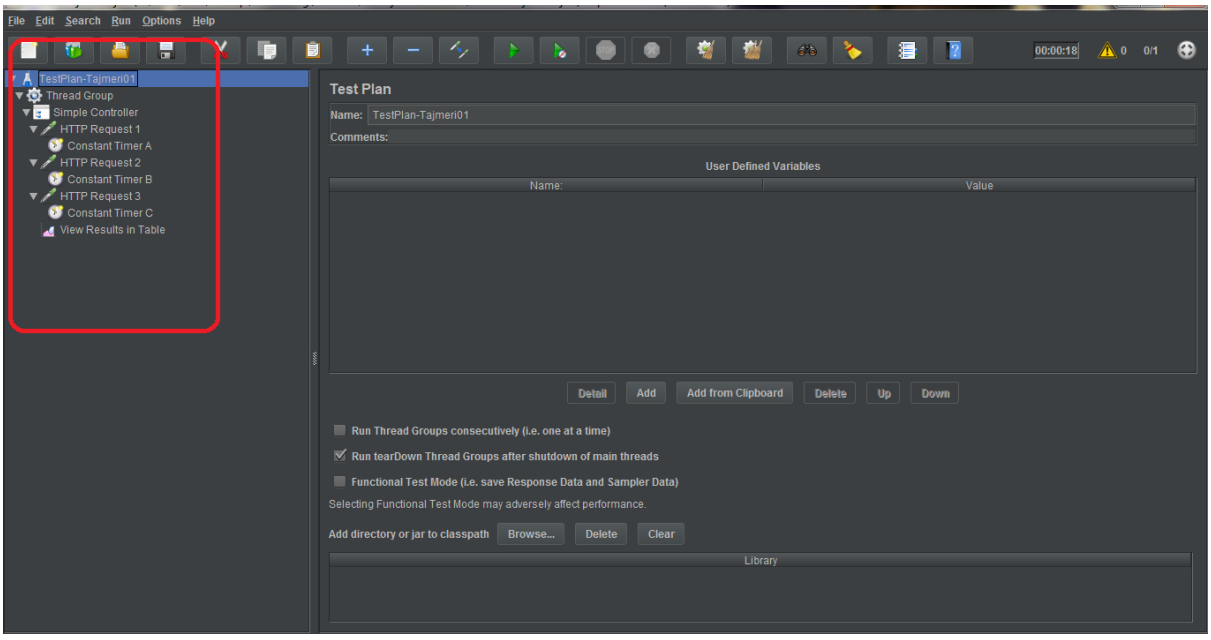
- Pauziranje korisnika (*Thread Group*) za određeni broj milisekundi.
- Dodavanje kašnjenja (*Delay*) između korisnika, tj. njihovog uključenja,
- Simuliranje realnog tempa u korišćenju i povećanju opterećenja aplikacije.

Najčešće korišćeni tajmeri su: Constant Timer i Uniform Random Timer. Oni pokrivaju 99% svih zahteva (Request-ova) koji se mogu desiti prilikom testiranja.

Primer:

Kreiramo Test Plan, dodajući sledeće komponente:

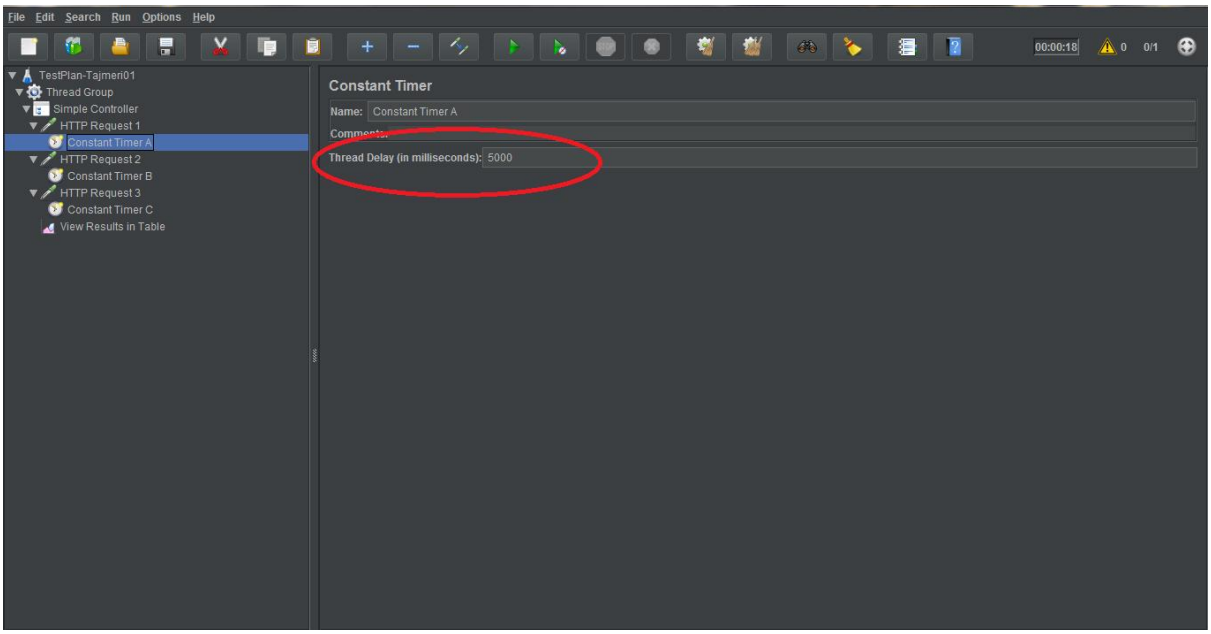
- Thread Group (setujem 1 korisnika),
- Simple Controller,
- Tri HTTP Request elementa 1, 2 i 3,
- Listener (View Results in Table),
- Tri Timer-a A, B i C, za svaki HTTP Request ide po jedan Constant Timer (Constant Timer).



Slika 8. Izgled Test plana sa Timer komponentama

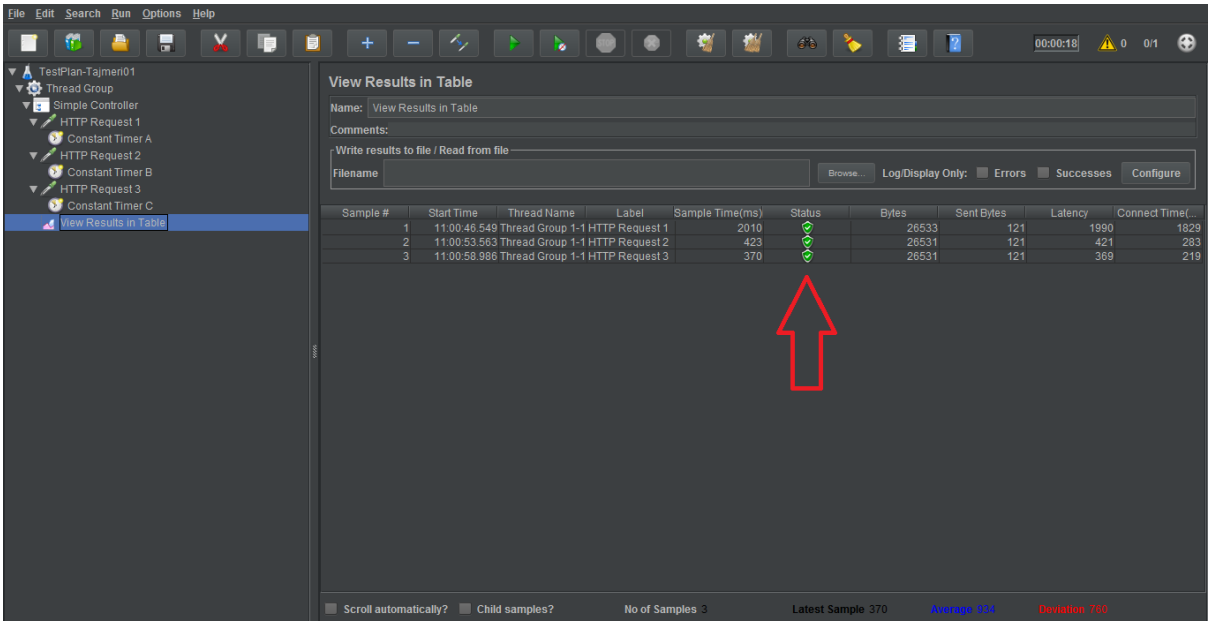
Timer komponente se dodaju desnim klikom na Simple Controller → Add → Timer → Constant Timer.

Zatim u Thread Delay polju u Contstant Timer-u setujem 5000 milisekundi (5 sekundi) da bude delay izmedju zahteva (Request-ova).



Slika 9. Setovan delay izmedju zahteva

Kada se test pokrene, na Listener komponenti (View Results in Table) mozemo videti da se zahtevi upucuju server sa vremenskim razmakom od 5 sekundi, koliko smo i setovali u Constant Timer komponenti.



Slika 10. Izvlazni rezultati testa sa setovanim razmakom od 5 sekundi izmedju Request-ova

Pored Constant Timer-a, u čestoj upotrebi je i Uniform Random Timer, koji se doneklae razlikuje od prethodnog. U Uniform Random Timer-u, postoje dva polja:

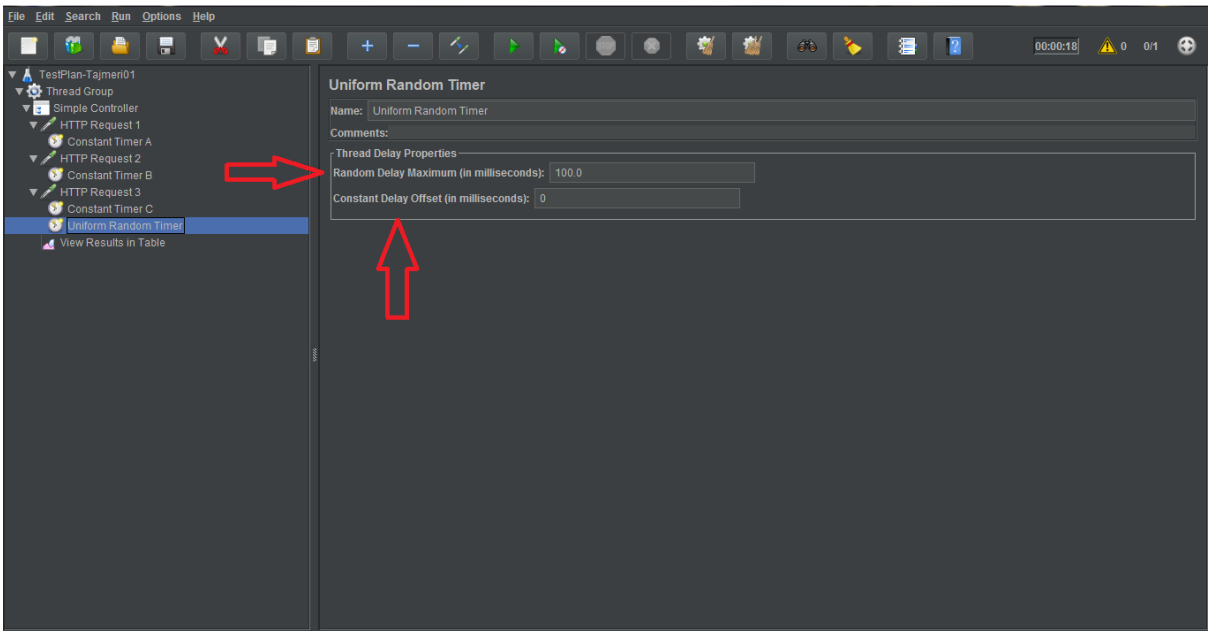
- Random Delay Maximum – znači da će delay imati neku slučajnu (random) vrednost do zadatog maksimuma,

- Constant Delay Offset – vrednost mogućeg odstupanja.

Formula za kašnjenje (*Delay*) u slučaju korišćenja Uniform Random Timer-a, je:

0,X * Random Delay Max + Constant Delay Offset

X – je nasumična vrednost od 0 do 9



Slika 11. Izgled Unifom Random Timer komponente

3.8.Upotreba Templejta

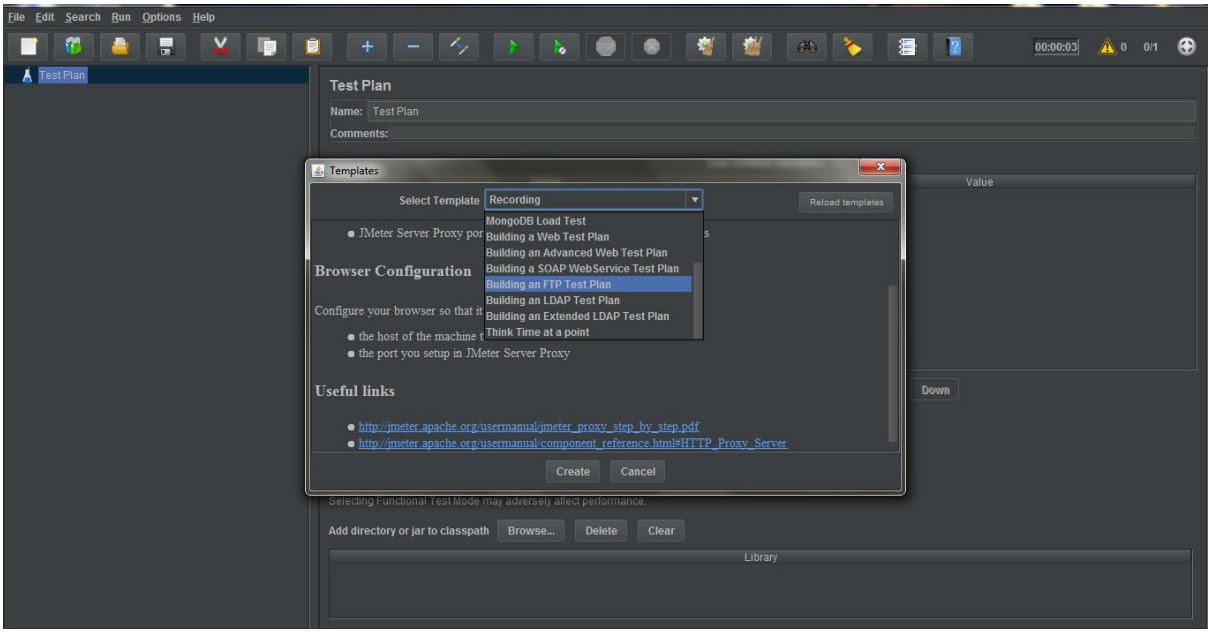
Templejti su predefinisani Test Planovi koji su već unapred kreirani i sačinjeni od odgovarajućih komponenata. Oni su veoma korisni ukoliko postoji potreba za testiranjem nekih slučajeva i scenarija koji se često ponavljaju. Zato se koriste Templejti, jer su to test skripte koje se mogu višestruko upotrebljavati.

Test Planovi pomoću Templejta se kreiraju klikom na dugme *Templates* na liniji sa alatima JMeter-a, ili se ide na File → Templates → nakon cega se otvori prozor "Templates" u kome mozemo da odaberemo zeljeni Templejt iz polja *Select Template.*

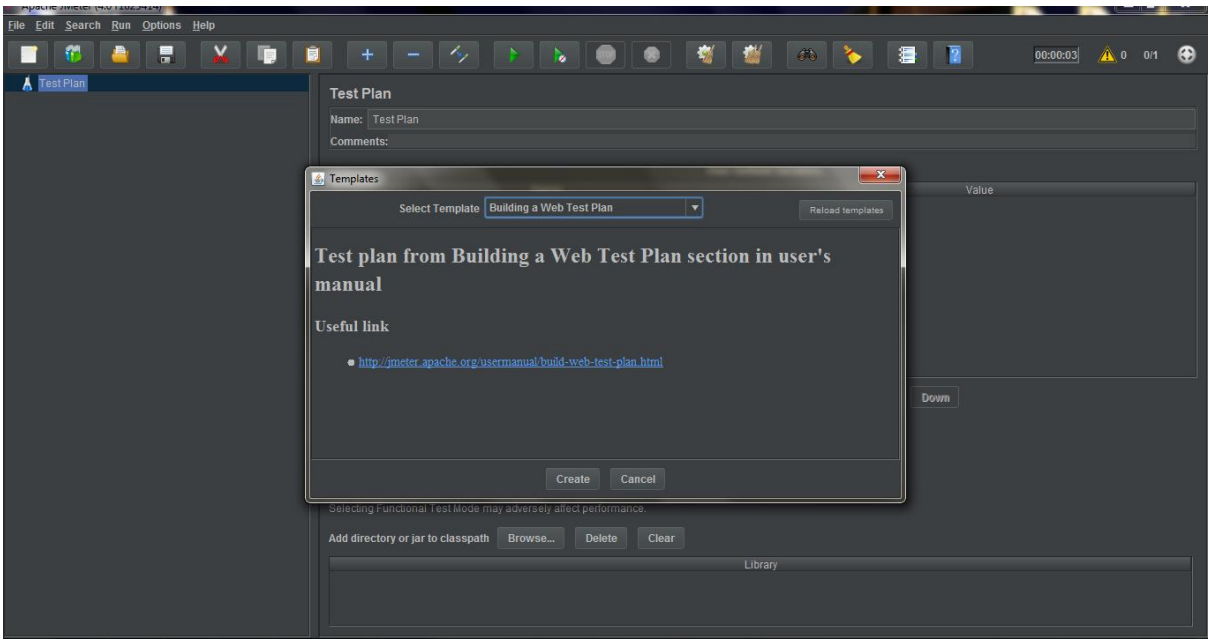
Kada se odabere željeni templejt u prozoru Templates se otvori opis tog templejta u kome se navodi za šta se on koristi i obično se daje i link ka nekom od sajtova na kome se nalazi preporučena literatura.

Klikom na dugme Create, kreira se novi Test plan sa potrebnim komponentama, generisanim na osnovu odabranog templejta.

Ukoliko se na već postojeći Test Plan, želi dodati Templejt, onda se na već postojećem Test Planu, selektuje komponenta na koju se Templjet želi dodati i odabere se novi templejt i klikne se Merge (ovo dugme se u ovom slučaju pojavi umesto dugmeta Save).



Slika 12. Izgled prozora Templates pri izboru Templejta iz padajuće liste



Slika 13. Izgled opisa odabranog Templejta u otvroenom prozoru Templates

4. Kreiranje Testova

4.1. Osnovni koraci u kreiranju JMeter testa

Osnovni koraci u kreiranju Jmeter testa su:

- Startovanje JMeter-a;
- Kreiranje Test Plana;
- Kreiranje korisnika (thread group);
- Dodavanje sampler-a;
- Dodavanje listener-a;
- Pokretanje testa (run test).

Setovanje korisnika u Thread Group

Može se dodati različit broj korisnika u Thread Group. Ovaj podatak se setuje tako što u *Number of Threads* unesemo neki broj, npr. 10.

Takođe, možemo da setujemo vreme za koje će se jedan po jedan korisnik dodati u test, odnosno na web stranicu. Polje za setovanje vremena je Ramp-up Period. Ako se stavi npr. 20s (na 10 korisnika), to znači da će se 10 korisnika postepeno uključivati na stranicu u periodu od 20s, odnosno na svake 2s će se uključiti po 1 korisnik dok se svih 20 korisnika ne uključi.

Polje *Loop Count* definiše koliko puta ćemo ponavljati iteraciju sa uključivanjem korisnika. Ako se klikne na Forever, onda će u predašnje navedenom testu 10 korisnika biti uključeno stalno dok se ne prekine petlja testa tj. izvršavanje testa.

Brisanje rezultata se vrši klikom na *Clear* i *Clear All*.

Zaustavljanje testa se vrši klikom na jedno od sledećih opcija:

- STOP – prisilno zaustavlja izvršenje testa i svih thread-ova odjednom.
- Shutdown – postepeno zaustavlja thread-ove jednog po jednog i na kraju zaustavi ceo test.

Napomena: kada se testira web aplikacija, uvek se kao sampler dodaje HTTP request.

4.2.Snimanje UI test plana

Postoje dva načina za snimanje UI Test Plana a da ne ide kroz sam JMeter. Preporučuje se korišćenje BadBoy softvera koji se skida sa www.badboy.com ili korišćenje Blazemeter plugin-a za Chrome browser.

Princip snimanja sa oba alata je sledeći:

1. Otvori se alat;
2. Snimi se test kliktanjem i unošenjem informacija,
3. Export-uje se snimljeni .jmx fajl koji se pri tome izgenerisao,
4. Potom se importuje taj isti snimljeni .jmx fajl u Jmeter,
5. U Test Plan se dodaju listener-i i mogu da se brišu višak fajlovi (ako postoje), setuje se broj korisnika i time se kompletira test plan,
6. Pokrene se skripta i prate se rezultati.

4.3.Izvršavanje test plana iz CMD-a

Izvršavanje testova bez GUI-a se vrši zato što sam JMeter-ov GUI koristi previše resursa lokalne mašine na kojoj se izvršavaju testovi, pa se zato na računarima koji imaju manje RAM memorije ili slabiji procesor testovi pokreću i izvršavaju direktno kroz cmd (Windows) ili terminalu (Linux). Takođe, ako je potrebno integrisati JMeter s nekim alatom za upravljanje procesima testiranja (npr. Jenkins) isto se koristi pokretanje i izvršavanje testova kroz cmd.

Maksimalan broj korisnika koji se može simulirati u testiranju pomoću JMetera zavisi od ograničenja mašina (računara) i mreže na kojoj se vrši testiranje, kompleksnosti testova i test skripti, i slično. Oko 500 korisnika je približno dovoljan broj za veb aplikaciju, koja ima manji broj targetiranih korisnika. Maksimalan broj broj korisnika koji se mogu simulirati sa 4GB RAM-a je 8000 – 9000 kada se sve optimizuje.

Osnovna pravila za optimizaciju JMeter testova su:

1. Koristiti JMeter Listenere samo za debug i za setovanje testova u početku,
2. Pokretati testove kroz cmd a ne kroz GUI,
3. Povećati HEAP u jmeter.bat fajlu na 2GB (2058 MB),
4. Koristiti LAN mrežu umesto Wi-Fi konekcije za load testiranje.

Koraci su sledeći:

1. Pomoću GUI-a kreiramo testove na klasičan način – dodavanjem Thread Group-a, HTTP Request-a.
2. Ukloni se listener iz Test Plana, ako postoji.
3. Ugasi se JMeter.
4. Otvori se cmd i kroz njega se pozicionira u bin folder u JMeter-u. Path za to je npr. C:\Users\besic\Desktop\Jmeter\bin

5. U cmd se unese komanda za izvršenje testa:

jmeter -n -t [adresa jmeter test skripte] -l [lokacija fajla u koji se upisuje rezultat]

Oznake u komandi su:
-n – oznaka za izvršaanje testa u modu bez GUI-a (non GUI mode)
-t – oznaka koja upućuje korisnika da iza nje dolazi lokacija test skripte
-l – oznaka koja upućuje korisnika na lokaciju na kojoj se nalazi fajl sa rezultatima, lokacija na koju će JMeter upisati rezultate.

Napomena: Potrebno je voditi računa o formatiranju naziva, kako test skripte tako i fajla sa rezultatima. Ne sme postojati razmak u nazivu!

Npr:

Test Plan – primer 08.jmx → nije dozvoljeno

TestPlan-primer08.jmx → dozvoljeno.

Komanda kojom se izvršava test u Jmeter-u kroz cmd izgleda ovako:

jmeter -n -t C:\Users\manojlovic\Desktop\Jmeter\Jmeter-test-planovi\TestPlan-primer08.jmx -l C:\Users\manojlovic\Desktop\Jmeter\Jmeter-test-rezultati\Test08.csv

6. Na kraju radi pokretanja testa se klikne **Enter**.

Pomoću komande **jmeter -h** u cmd-u izlistavamo listu komandi i naredbi koje su nam na raspolaganju.

Pomoću komande **jmeter -?** u cmd-u dobijamo listu prostih komandi i naredbi koje su nam na raspolaganju u Jmeter-u.

4.4.Kreiranje HTML izveštaja upotrebom CMD-a

Prvi korak je kreiranje test plana korišćenjem GUI-a. U test plan je potrebno dodati sledeće komponente:

1. Thread Group.
2. Tri HTTP Request komponente (za Home Page, About Page i Archives Page)
3. Dodajemo i Response Assertion, gde setujemo:
   - Odaberemo Main Sample Only;
   - Odaberemo Response Code;
   - Odaberemo Equals;
   - U Patterns to Test upišemo 200 kao Response Code koji želimo da testiramo;
   - Radi prvobitnog pregleda rezultata ubacimo View Results Tree listener.

Drugi korak je da otvorimo cmd i kroz njega se pozicioniramo u bin folder JMeter alata. Path za bin: C:\Users\besic\Desktop\Jmeter\bin

Treći korak je da u cmd unesemo komandu kojom pokrećemo Test Plan. Komanda je:

jmeter -n -t [lokacija_test_skripte] -l [lokacija_fajla_sa_rezultatima] -e -o [lokacija_foldera_u_kom_se_nalaze_rezultati]

Pri tome u ovoj liniji imamo dve nove komande:
-e – komanda kojom se naređuje generisanje HTML izveštaja;
-o – komanda kojom se definiše lokacija foldera sa rezultatima u koji se ima smestiti generisana HTML forma.

Komanda za pokretanje testa izgleda ovako:

jmeter -n -t C:\Users\manojlovic\Desktop\Jmeter\Jmeter-test-planovi\TestPlan-primer12.jmx -l C:\Users\manojlovic\Desktop\Jmeter\Jmeter-test-rezultati\Test12.csv -e -o C:\Users\manojlovic\Desktop\Jmeter\Jmeter-test-rezultati2

Napomena: Prilikom kreiranja ovih fajlova potrebno je znati da se fajl Test12.csv zapravo kreira sam izvršenjem komande iz cmd-a tako da zapravo u fajl u koji se generišu rezultati treba da bude prazan. Takođe, folder Jmeter-test-razultati2 koji se tiče HTML izveštaja mora biti prazan jer u njega Jmeter smešta izveštaje koje napravi.

Napomena: HTTP izveštaji koji se generišu imaju kompletnu metriku prikazanu grafički. Dvostrukim klikom na index.html se otvaraju potrebni izveštaji.

Pored generisanja HTML izveštaja nakon izvršenog testa, moguće je kreirati HTML izveštaj iz već ranije kreiranog .csv fajla nekog od prethodnih testova. Ovo je pogodno ukoliko se vrši retesting pojedinih delova web aplikacije i pri tome je dobro izvršiti poređenje dobijenih rezultata u prvobitnom testiranju i novom (retesting) testiranju.

Za generisanje HTML izveštaja iz staih .csv fajlova potrebno je uneti sledeću komandu u cmd:

jmeter -g [lokacija_csv_fajla] -o [lokacija_foldera_za_rezultate]

Komanda izgleda ovako:

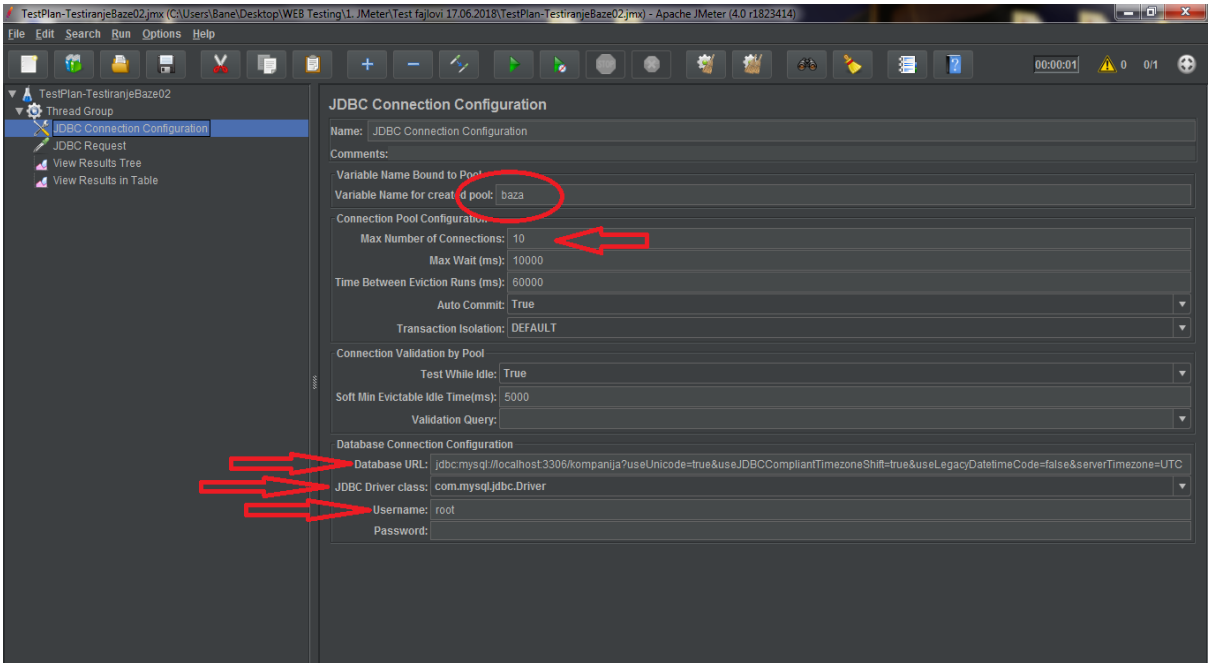
jmeter -g C:\Users\manojlovic\Desktop\Jmeter\Jmeter-test-rezultati\Test12.csv -o C:\Users\manojlovic\Desktop\Jmeter\Jmeter-test-rezultati3

5. Primeri testiranja

5.1.Testiranje baze podataka

Za kreiranje ovakvog test plana radi se sledeće:

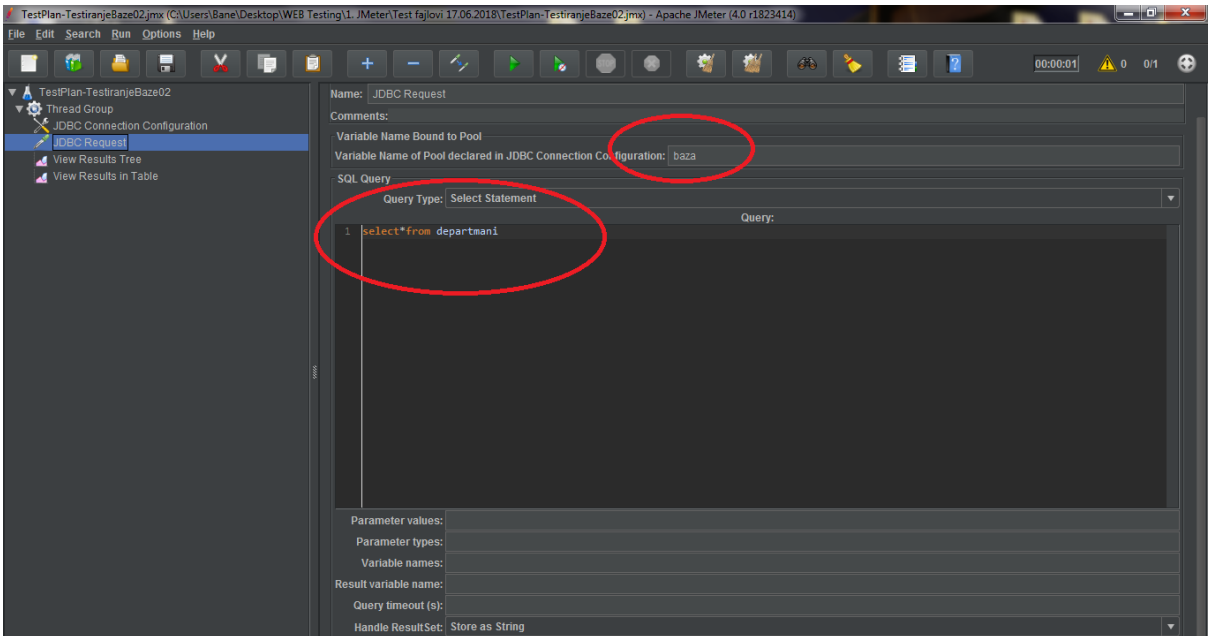
1. Kreira se novi test plan,
2. Dodaju se korisnici,
3. Na Thread Group-u se doda komponenta JDBC Connection Configuration koju konfigurišemo da gađa bazu podataka. Konfigurisanje se vrši na sledeći način:
   - U polje Max Number of Connections upisuje se neki broj, npr. 10;
   - U polje Database URL upisuje se URL baze podataka npr: **jdbc:mysql://localhost:3306/kompanija?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC**
   - U polje JDBC Driver Class: unese se oznaka drajvera za povezivanje s bazom, npr. com.mysql.jdbc.Driver (potrebno je dodati jar fajl tj. biblioteku u lib folder i restartovati JMeter);
   - U polje Username se unese korisničko ime za pristup bazi (*root*);
   - U polje Password se unese šifra za pristup bazi.



Slika 14. TestPlan za testiranje baze podataka

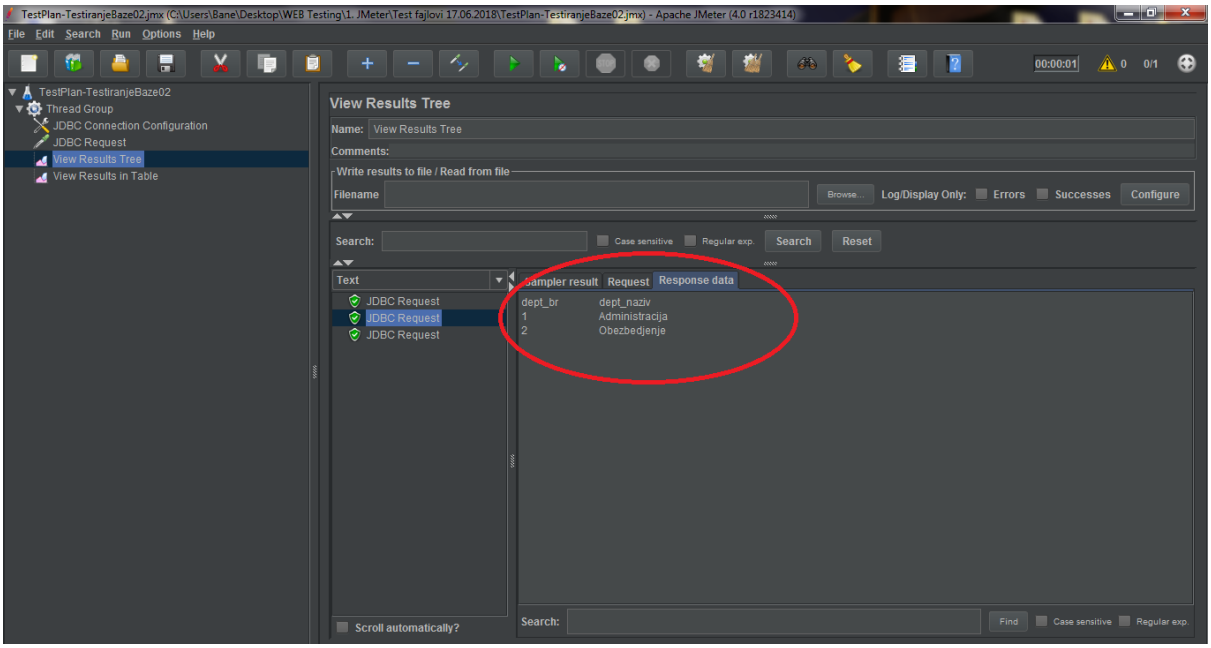
Ukoliko želimo da testiramo da li se neki podatak eksplicitno nalazi u bazi podataka onda je potrebno prilikom kreiranja test plana navesti tačno koji podatak tražimo kao i preciznu poziciju u tabeli baze na kojoj se i taj podatak nalazi.

Zatim je potrebno napisti upit koji se upućuje bazi podataka. Ovaj upit se upisuje u polju *Query* u komponenti JDBC Request.



Slika 15. Upit u bazu podataka

Po izvršenju testa dobija se ispisan sadržaj baze u polju *Response* u View Results in Tree.



Slika 16. Rezultat test ispisan sadržaj iz baze

Da bi se ovakav test plan kreirao, potrebno je prvo napraviti test plan za testiranje baze koji poseduje sledeće komponente:

1. Thread Group – korisnici;
2. JDBC Connection Configuration – za uspostavljanje veze s bazom;
3. JDBC Request – za upis upita koji se želi izvršiti;
4. Listener-i – za prikaz rezultata (View Results Tree i View Results in Table)-

Kada smo kreirali test plan za testiranje konekcije sa bazom podataka, potrebno je dodati sledeće komponente u test plan:

5. Komponenta Response Assertion koju dodamo u okviru JDBC Request komponente. U Response Assertion potrebno je setovati sledeće:
   - Izabrati opciju *Jmeter Variable Name to Use* i u polje pored ove opcije upisati oznaku tačnog polja u tabeli u kome se nalazi podatak koji se proverava i to u formatu:

     (oznaka_kolone)_(broj_reda) = C_1

     Prethodno ovu oznaku setujemo u komponenti JDBC Request gde u polju za Variable names unesemo oznaku za svaku kolonu u tabeli baze, npr. ako imamo 6 kolona oznake su A, B, C, D, E, F.

   - Zatim, u Response Assertion je potrebno uneti podatak koji se proverava tj. traži u tabeli. Ovo unesemo u polju *Patterns to Test*.
   - Poslednje, u Response Assertion u okviru sekcije Response field to Test odabiramo opciju Text Response jer je ona potrebna budući da proveravamo tekstualni sadžaj tj. očekujemo tekstualni response.

6. Za proveru rezultata testa dodajemo Assertion Results listener komponentu u okviru test plana. U polju Assertion ove komponente se proverava da li je otišao ili ne.
7. Pokrenemo test.

5.2.Testiranje Debagovanjem

Testiranje debagovanjem podrazumeva da se pri kreiranju Test Plana koristi View results in Tree, zato što on u kartici sa rezultatima omogućava izbor željenog ofrmata za prikaz rezultata kako bi se vizuelnom proverom utvrdilo stanje koda i pronašle vidljive greške ukoliko postoje.

Pri testiranju debagovanjem koristimo:
- Debug Sampler,
- Debug PostProcessor,
- Stap-By-Step Debugger.

Debug Sampler, se koristi za analizu promenljivih koje se koriste u testovima prilikom testiranja performansi veb aplikacija.

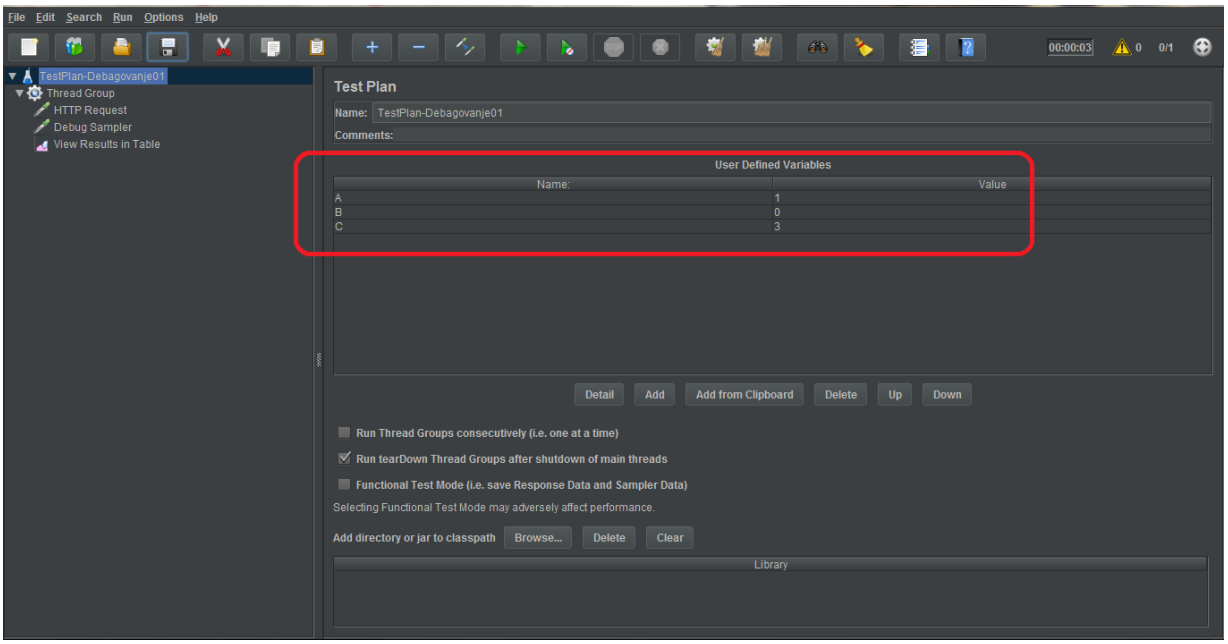
Debug Sampler, dodajemo u Test Plan, desnim klikom na Thread Group → Add → Sampler → Debug Sampler. Kada dodamo Debug Sampler u test pla, vidimo da na raspolaganju imamo opcije: JMeter properties, JMeter variables, System properties. One mogu imati vrednosti TRUE ili FALSE, čime se odabira šta će se terstirati u datom test planu.

Primer:

Kreiramo test u kome dodajemo:

- Thread Group,
- HTTP Request,
- Debug Sampler,
- View Results Tree.

Zatim u Test Plan komponenti u polju User Defined Vriables dodamo neke probne varijable, npr: A = 1, B = 0, C = 3

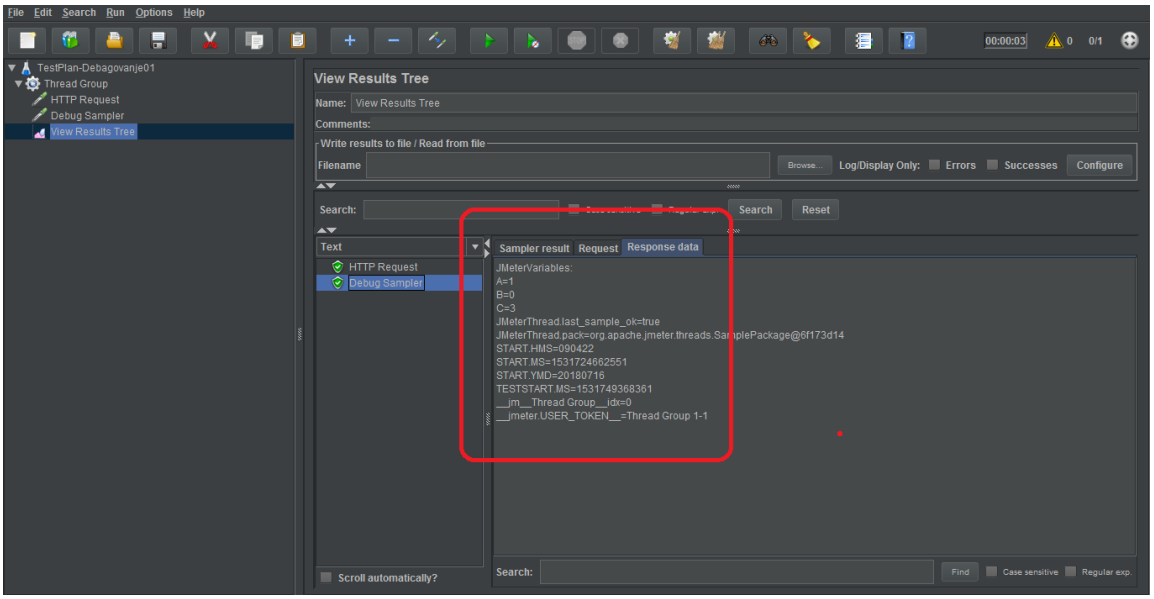


Slika 17. Dodajemo varijable u Test Plan

Nakon što pokrenemo test, kao odgovor u Response Data dobićemo ispisane JMeter varijable kao:

JMeterVariables:
A = 1,
B =
C = 3

Ovde možemo pregledati i proveriti dobijene rezultate.



Slika 18. Ispis varijabli na Response Data u View Results in Tree

Debug PosProcessor, ima sve što i Debug Sampler, samo što Debug Sampler ne može da se doda unutar HTTP Request-a. Zato, ukoliko želomo da istestiramo i debagujemo svaki pojedinačni zahtev (*Request*), koristićemo Debug PostProcessor.

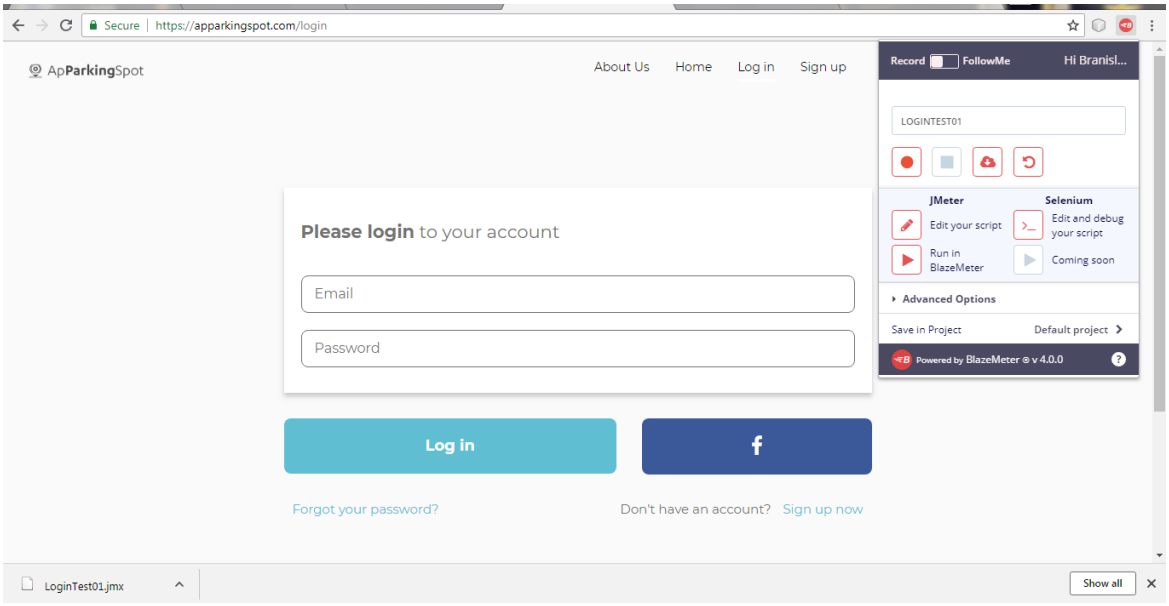
Step-By-Step Debugger, omogućava da se izlazni rezultat pregleda korak po korak sa breakpoint-ima, kao kada se kod debaguje u radnom okruženju. Step-By-Step Debugger dodajemo u pomoću plug-in menadžera u JMeter, tako što se klikne na dugme za *Plugin Manager* i potom se klikne na karticu *Available Plugins* i tamo se izabere BlazeMeter Step-By-Step Debbuger i potom se klikne na dugme *Apply Changes and Restart JMeter*.

5.3.Testiranje Login forme

Testiranje Login forme, vrši se snimanjem testa, pomoću BlazeMeter plugin-a za Chrome browser. Ovo je ujedno i primer kako se snima test skripta, koja se posle importuje u JMeter.

Osnovni koraci su:

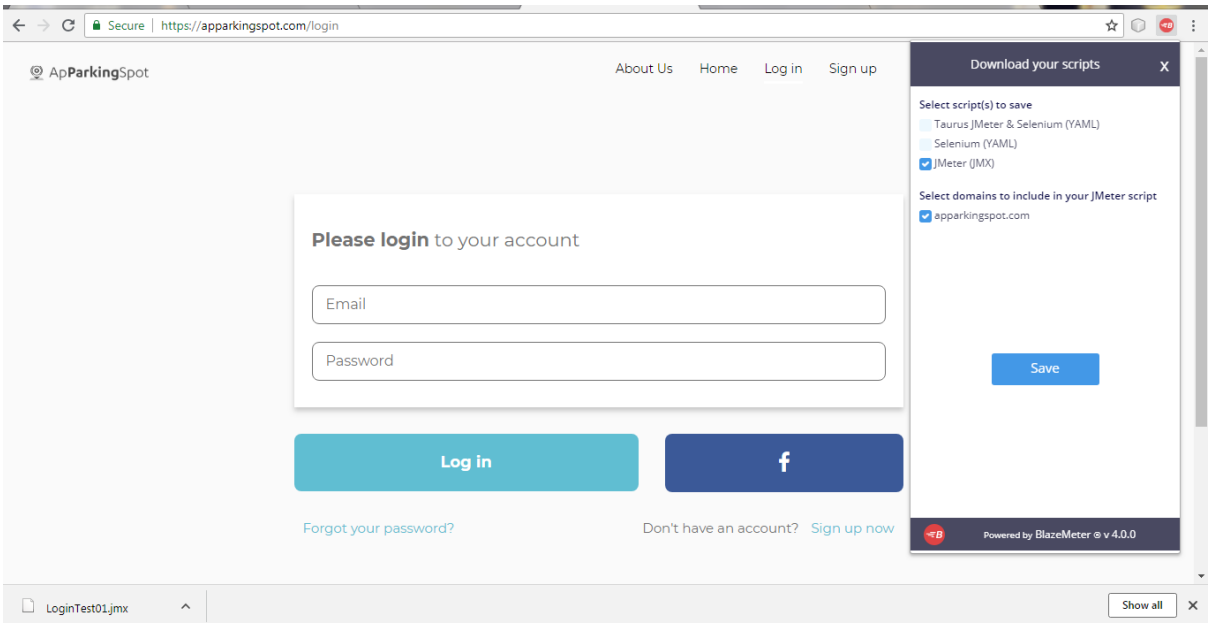
- Instaliramo BlazeMeter Recorder ekstenziju na Chrome browser sa google stor-a,
- Startujemo BlazeMeter Recorder i logujemo se u BlazeMeter profil na ekstenziji,
- Unesemo URL adresu veb aplikacije koju testiramo u browser i odemo na tu adresu,
- Otvorimo stranicu na kojoj se nalazi login forma koju testiramo,
- Unesemo naziv testa pod kojim će biti zapamćena i test skripta koja se izgeneriše,
- Starujemo snimanje testa,
- Napravimo planirane test korake za dati test slučaj (unesemo *Username*, unesemo *Password*, klik na *Login*),
- Kada smo završili planirane test korake, kliknemo na *Stop* dugme za završetak snimanja testa.



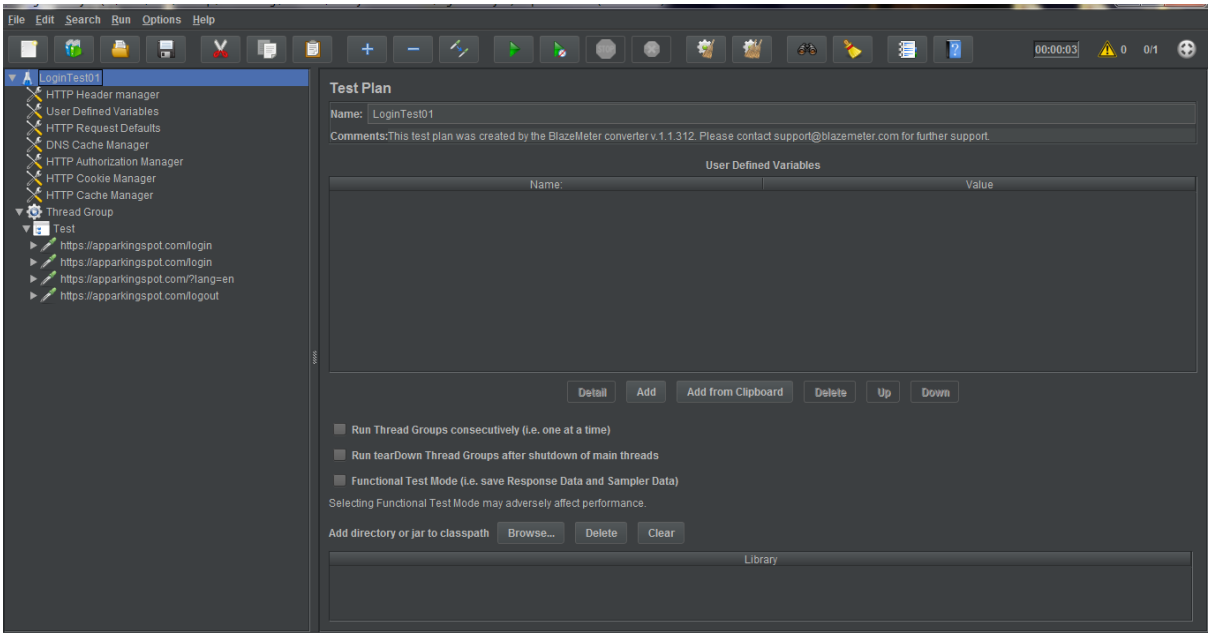
Slika 19. Otvorena Login forma i upaljena BlazeMeter ekstenzija

Kada smo snimili test, dobili smo generisanu test skriptu koju eksportujemo iz BlazeMeter recorder-a. Prilikom eksportovanja test skripte, mormo da selektujemo opciju JMeter (JMX) kako bi se dobila skripta sa ekstenzijom .jmx, jer je to skripta koju JMeter može da importuje i pročita, i potrebno je selektovati koji domen mora da se uključi u test skriptu (apparkingspot.com), kako bi testovi imali tačan URL koji gađaju.

Zatim, ovu test skriptu importujemo u JMeter i dobili smo kreirani test plan koji možemo kasnije da optimizujemo izbacivanjem nepotrebnih komponenata ili dodavanjem nekih drugih potrebnih komponenata (npr Listener-a), ili možemo setovati neke parametre po potrebi (broj korisnika i slično).



Slika 20. Eksportovanje snimljene test skripte



Slika 21. Importovana test skripta u JMeter

Nakon što smo izvršili potrebne izmene i memorisali test plan, možemo ga pokrenuti. Izlazne rezultate posmatramo u dodatim Listener komponentama.

5.4.Testiranje API-ja veb servisa

API – Application Programming Intefrace, se koristi kod veb servisa za komunikaciju između klijenta i server. API obrzbeđuje brzu i sigurnnu komunikaciju između 2 entiteta (tj. servera).

Razlikujemo SOAP i REST servise.

a) Testiranje REST API servisa

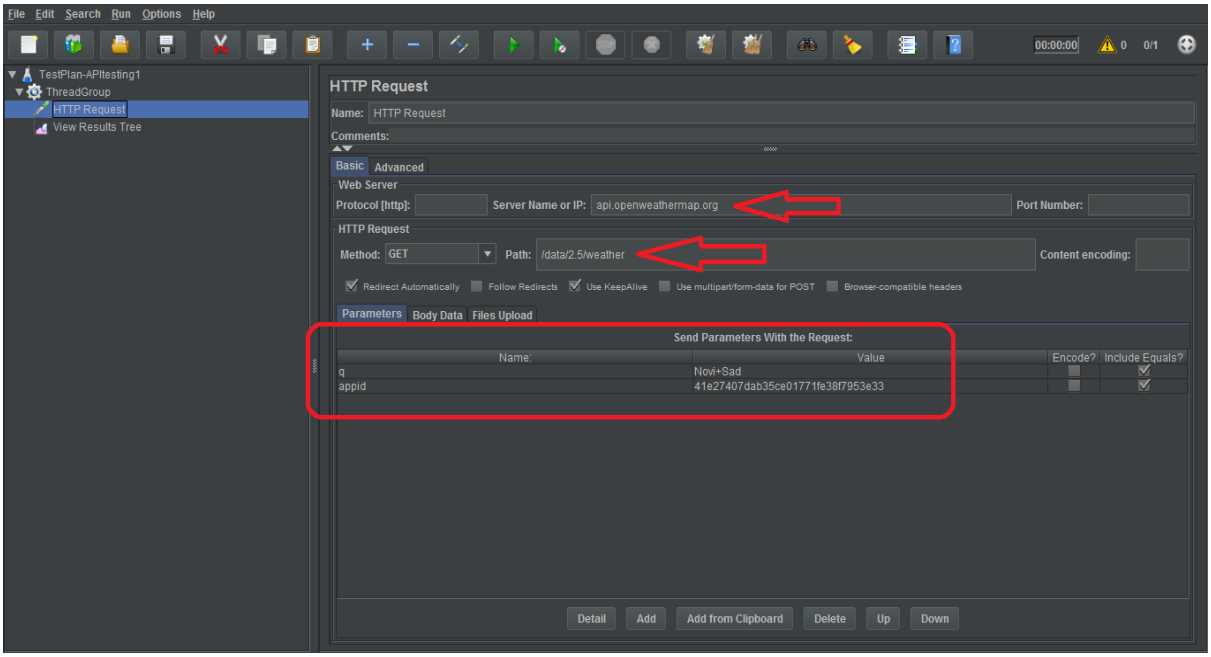
Za testiranje REST APIservisa kreiramo Test Plan u kome moramo dodavati HTTP Request Sampler element (ranije se koristio SOAP/XML-RPC Request).

U Test Plan dodajemo:
- Thread Group,
- HTTP Request Sampler,
- Listener *View Results in Table*.

Zatim moramo da pribavimo kredencijale za testiranje REST API servisa. U primeru koji je prikazan ove kredencijale smo dobili sa sajta *openweather.org*. Sa njega smo kreiranjem profila skinuli API key (41e27407dab35ce01771fe38f7953e33).

Za uspostavljanje konekcije (pogađanje) željenog veb servisa koristimo sledeću URL adresu:

http://api.openweathermap.org/data/2.5/weather?q=Novi+Sad&appid=41e27407dab35ce01771fe38f7953e33



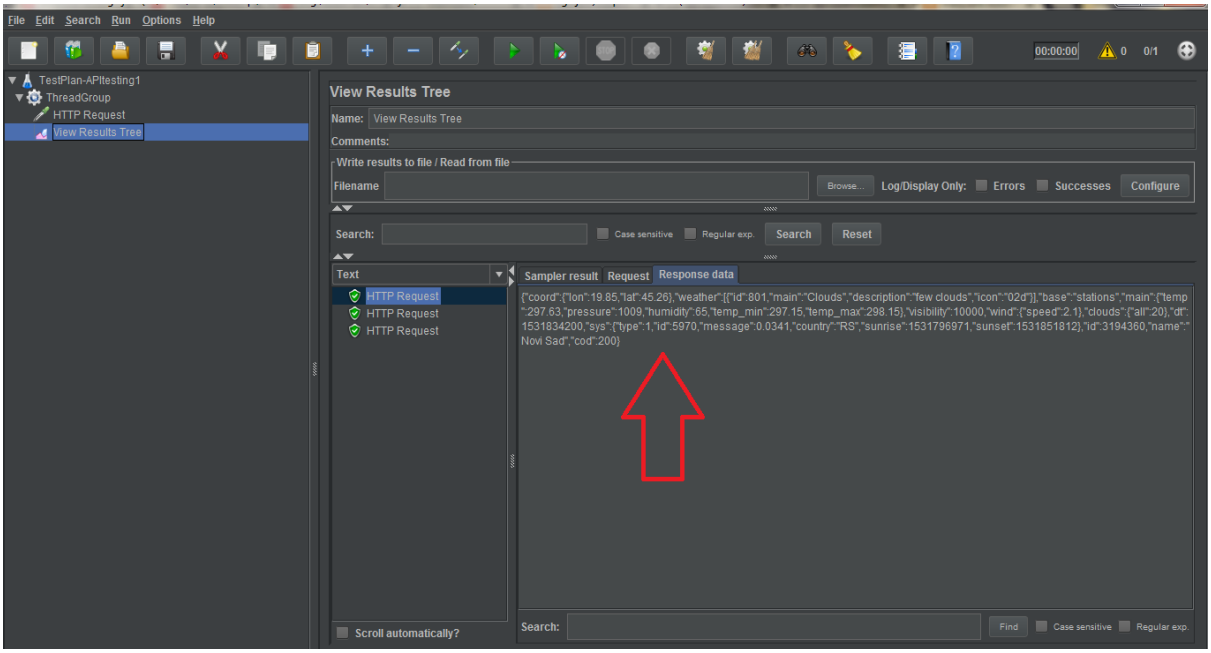
Slika 22. Test Plan za testiranje REST API servisa

Osnovni parametri koje unosimo u HTTP Request su:

- api.openweathermap.org – IP adresa veb servisa kog gađamo,
- /data/2.5/weather – je PATH ka kom idemo na željenom veb server,
- q=Novi+Sad – parametar koji se odnosi na grad,
- appid=41e27407dab35ce01771fe38f7953e33 – parametar koji se odnosi na API ključ.

Pri čemu se q i appid upisuju u polju Send Parameters with Request u HTTP Request komponenti.

Nakon što smo popunili sve parametre pokerćemo test i analiziramo dobijene rezultate.



Slika 23. Dobijeni rezultati nakon izvršenog testiranja REST API servisa

b) Testiranje SOAP API servisa

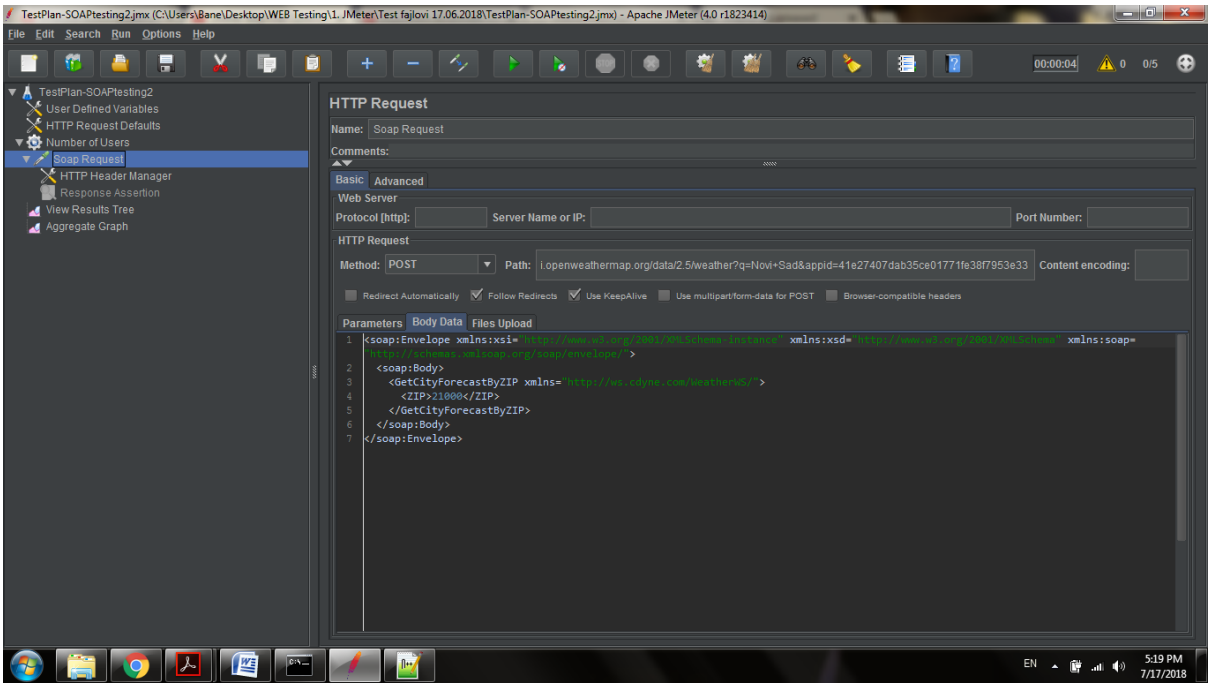
Za testiranje SOAP API-ja, potrebno je upotrebiti templejt, koga kreiramo pomoću *Templates* → u Select Templates odaberemo *Building a SOAP Web Service Test Plan* → *Create*.

Zatim na kreiranom Test Planu, potrebno je disejblovati *Response Assertion* komponentu koja je importovana sa templejtom. U *Soap Request* komponenti treba dodati za Path, sledeći URL:

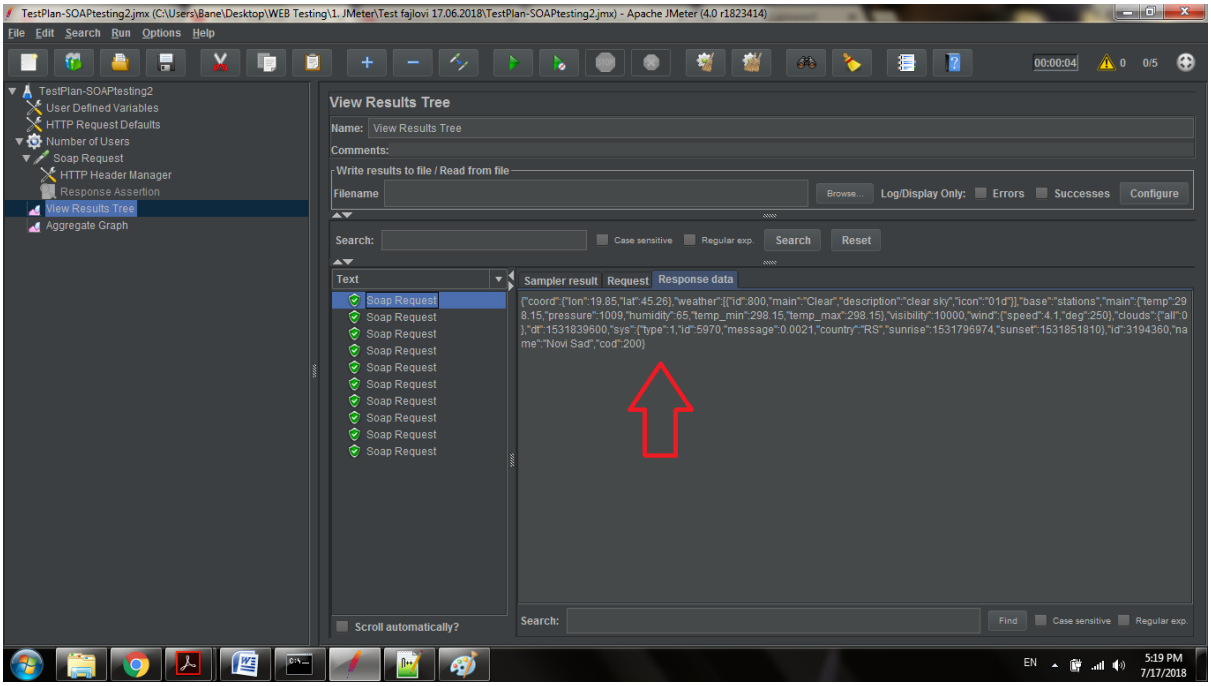
http://api.openweathermap.org/data/2.5/weather?q=Novi+Sad&appid=41e27407dab35ce01771fe38f7953e33

A u Body Data kartici unutar taga <ZIP>21000</ZIP> treba uneti potrebni postanski (ZIP) broj.

Nakon što smo uneli željene izmene, potrebno je pokrenuti test i analzirati dobijene rezultate.



Slika 24. Test Plan za testiranje SOAP API servisa



Slika 25. Dobijeni rezultati nakon izvršenog testiranja SOAP API servisa

6. Literatura

[1] Popvic J., "Testiranje Softvera u praksi", Računarski fakultet Beograd, 2012,

[2] Publikacija, "*Apache JMeter User Manual*" – sa sajta: www.jmeter.apache.org

[3] "*JMeter Begginer Tutorial*" – sa kanala: "Automation Step by Step – Raghav Pal"

[4] "*Performance Testing Tutorial*", sa sajta: www.guru99.com