### Teljesítménytesztelés

*A teljesítménytesztelés alapjai könnyen, érthetően megfogalmazva. Megtudhatjuk milyen fajtái vannak a performancia tesztelésnek, és melyiket mikor érdemes használni. A JMeter alapszintű bemutatásával gyakorlati alkalmazásban is láthatunk egy teljesítménytesztet. (Azért bizakodom, hogy nem egyszerre próbálja ki az összes Olvasó és állítjátok közösen fejre a weboldalunkat. – a szerk.)*

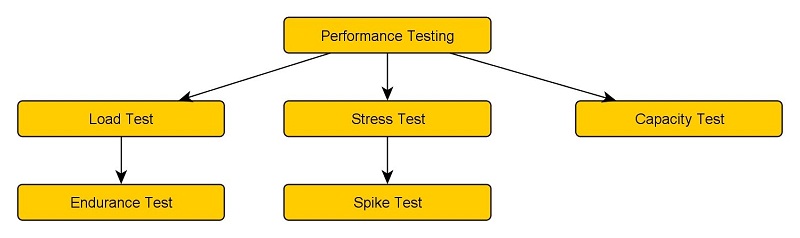
Nem funkcionális tesztelésről sajnos ritkábban esik szó. Örülünk, hogy működik az alkalmazásunk, kinek van arra ideje és pénze, hogy olyan kérdésekre válaszoljon, minthogy mennyire megbízható? Mennyire stabil? Karbantartható-e egyáltalán? Milyen könnyű a végfelhasználónak használni? Vagy, hogy mennyire terhelhető? Ebben a cikkben az utóbbi kérdéskört fogjuk körüljárni. Rövid elméleti áttekintés után használni fogunk egy alkalmazást, amivel terhelünk egy weboldalt, majd kimutatást készítünk, hogyan viselkedett a terhelés alatt.

#### Mi az a Performance Testing?

Az ISTQB fogalomtár a következőket írja: „Tesztelési folyamat, mellyel a szoftvertermék teljesítményét lehet meghatározni.” Ez kicsit túl általános, nekem jobban tetszik, amit a Wikipedián találtam: „Olyan tesztelési folyamat, ami meghatározza, hogy milyen gyors a rendszer egy bizonyos szempontból, adott terhelés alatt. Ezen kívül validálhat és verifikálhat más minőségi tényezőket is, mint skálázhatóságot, megbízhatóságot vagy forrás kihasználtságot.” A lényeg, hogy egy nem funkcionális tesztelési folyamat, ami szoftvertermék teljesítményével foglalkozik. Meg kell még említeni, hogy a fejlesztés korai szakaszában is van értelme vele foglalkozni. Segít felfedni a teljesítménybeli problémákat akkor, amikor még nem késő.

#### Milyen fajtái vannak?

A Performance Testing vagy PET egy gyűjtőfogalom. Sok fajtája létezik, én most 6-ot említek, amelyek szerintem a legfontosabbak webes alkalmazások szempontjából.  
  
Az elsőnek vegyük magát a *Performance Testing*et, ezt külön típusnak is lehet venni. Itt arra kell gondolni, hogy kifejezetten a sebességgel foglalkozunk. Olyan kérdésre ad választ, mint pl. az elég gyors lesz-e a weboldalam? Hogyan nő a válaszidő, ha egyre többen használják? Arra keresi a választ, hogy a termék megfelel-e a teljesítménybeli követelményeknek (ez az utóbbi természetesen annyira általános, hogy mindegyik fajtára igaz, attól függően, milyenek azok a követelmények).  
  
Az egyik leggyakoribb fajta a *Load Test*, vagy terheléses teszt. Itt azt verifikáljuk, hogyan viselkedik az alkalmazás normális vagy annál magasabb terhelés alatt. Változik-e a viselkedése terhelés alatt?  
  
A Load Test egy fajtája az *Endurance Test*. Leginkább „Terhelési próbaként” vagy „Tűrőpróbaként” lehetne lefordítani. Ez a tesztelési fajta kifejezetten hosszabb ideig tartó, folyamatos terhelés mellett vizsgálja a rendszer működését. Gondoljunk csak arra, hogy lehet, hogy egy 1 órás teszt alkalmával hibátlanul működik a rendszer, de egy több naposnál már kialakulhatnak olyan problémák, amelyek csak hosszabb tesztek alkalmával lehet megtalálni.  
  
A következő a *Stressz teszt*. A Stressz teszt azt vizsgálja, hogy maximális, még inkább túlterhelés alatt, korlátozott erőforrásokkal, hogyan viselkedik, még inkább hogyan „hal” meg a rendszer, ill. „támad fel”.  
  
Ennek egy alfajtája a *Spike test*. Itt tipikusan arra kell gondolni, hogy rövid idő alatt sok felhasználót engedünk rá a célrendszerre, majd szintén gyorsan le is csökken a számuk, így vizsgálva meg, hogyan reagál a rendszer a gyors, drasztikus változásokra.  
  
Legvégül meg kell említenünk a *Capacity Test*et. A Kapacitás teszt azt vizsgálja, vajon mennyi felhasználót/tranzakciót bír el a rendszer? Segít kideríteni, milyen és mennyi erőforrást kell bevonni a későbbi növekvő felhasználószám esetén.

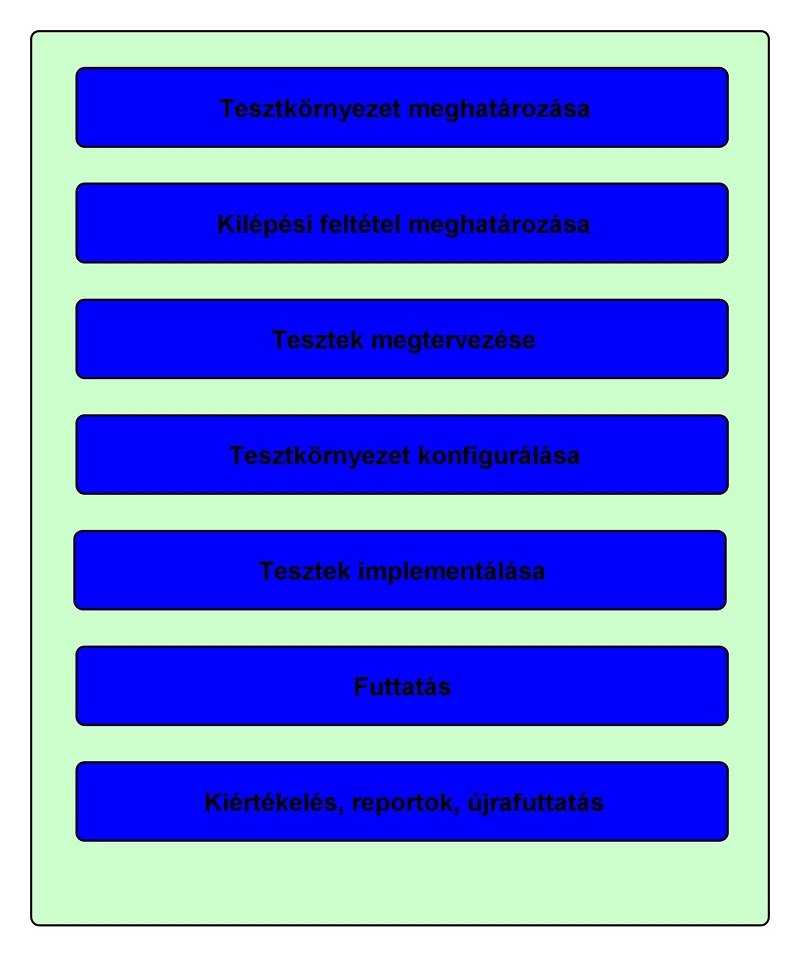


#### Milyen előnyei vannak?

A PET segít kideríteni, hogy a termék megfelel-e a teljesítménybeli követelményeknek. Elősegíti a teljesítménybeli gyenge pontok felfedését, rendszerhibák okainak felderítését. Összehasonlíthatunk vele kettő vagy több rendszert, minimalizálhatjuk vele a hardware, ill. software költségeket a megfelelő teljesítményhez. Elősegíti az applikáció skálázhatóságát, ezen kívül kapacitás és teljesítménnyel kapcsolatos statisztikai adatokat szolgáltat.

#### Hogyan történik mindez?

Ideális esetben, mielőtt bármit is tesztelnénk, nagyon sokat kell foglalkoznunk a tesztkörnyezet meghatározásával. Itt nagyon sok kérdést tehetünk fel és fel is kell tennünk. Ilyenek pl.: Mit tudunk a felhasználókról? Hogyan és honnan fogják elérni a rendszert? Milyen fizikai kapcsolatuk van (kapcsolat sebessége stb.)? Tűzfalon belül és kívül is használják? Milyen kliens oldali alkalmazással teszik ezt? Cache-elés szóba jöhet? Aztán ne felejtsük el a hardware tulajdonságokat sem. Milyen hardware-rel rendelkezik a szerver, ill. a kliens? Milyen processzorral, memóriával dolgoznak? Egyszóval, minden olyan kérdést fel kell tennünk, ami hatással van a teljesítményre. Ezért olyan fontos, hogy ezzel kezdjük a munkát. A második lépés a kilépési feltététel meghatározása. Ez határozza meg, mikor tekintjük befejezettnek az egész folyamatot. Ilyen konkrétumok lehetnek, pl. 1000 felhasználót „bírjon el” a weboldal, azaz elfogadható válaszidőn belül még elérhető legyen, vagy, hogy 500 felhasználói után is 40 ms alatt legyen a válaszidő.  
  
Ezután tervezzük meg a teszteseteket. Itt meg kell határozni a teljes tesztforgatókönyvet. Milyen felhasználókat fogunk ráereszteni a rendszerre? Milyen felhasználói esetet szimulálunk? Milyen tesztadatokkal fogunk dolgozni? Mivel fogjuk ezeket generálni? Itt tervezünk meg minden részletet a későbbi megvalósításhoz. Ezután hozzáláthatunk a tesztkörnyezet konfigurálásához. Ahhoz, hogy rendszerünket terhelésnek vessük alá, elő kell készítenünk a tesztkörnyezetet. Ki kell választani a megfelelő eszközöket/toolokat. Mivel és hogyan fogjuk elérni a kívánt terhelést? Hogyan fogunk monitorozni? Milyen plusz források kellenek a megtervezett tesztjeink későbbi futtatásához? Ha már mindent tudunk tesztjeinkről és azt is, hogy hogyan kell őket megvalósítani, nincs más hátra, mint a tesztek implementálása. A megvalósítási fázis után a futtatás következik. A keletkezett monitoring fájlokból tudjuk kiértékelni futtatásunkat.  
  
Valamilyen logolási folyamat mindig van, ez biztosítja a későbbi analizálás alapját, innen tudjuk kitalálni, hogy a kilépési feltétel teljesült-e. Jellemzően különböző kimutatásokat, diagramokat is készítünk, hogy az akár napokig tartó futtatásokból nyomon lehessen követni a rendszer változásait. Végül egy riportot készítünk a futtatásról, ezzel jelezve a folyamat végét. Természetesen ezekből az eredményekből merítve javítjuk a tesztjeinket az újrafuttatáshoz.

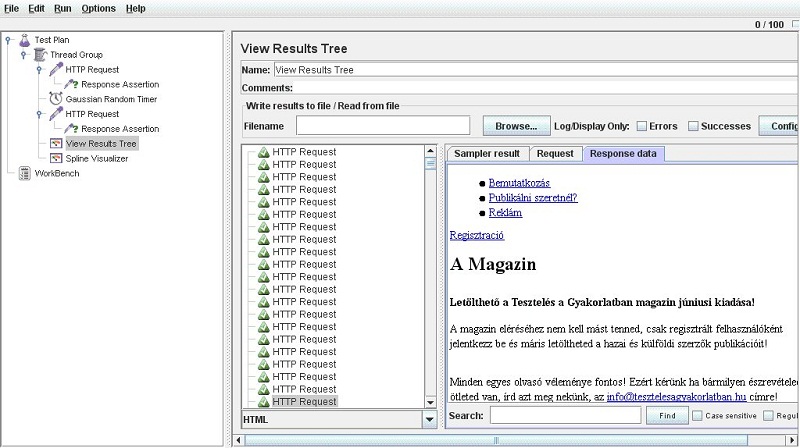


#### Lássuk gyakorlatban!

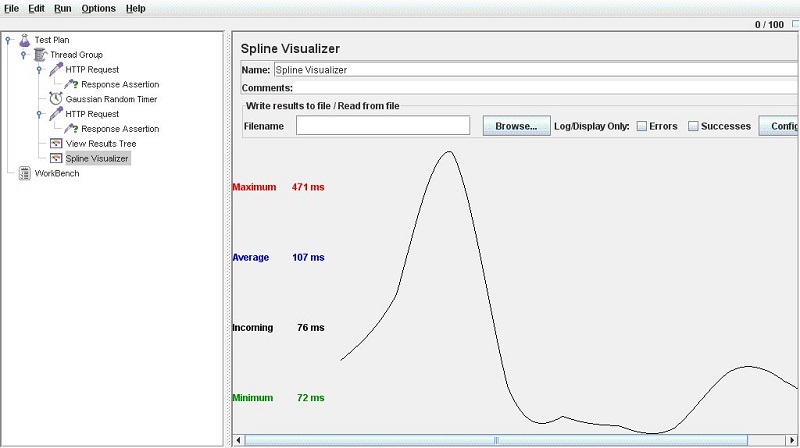
Az alábbiakban meg fogjuk ismerni a JMetert. Az Apache JMeter egy nyílt forráskódú performance tesztelő szoftver. A következőket tesszük: Meglátogatjuk a [www.tesztelesagyakorlatban.hu](http://www.tesztelesagyakorlatban.hu/) weboldalt. Verifikáljuk, hogy a kezdőlapon megjelenik-e a „A Magazin” felirat. Ha igen, várjunk nagyjából 5 másodpercet és látogassuk meg a „Reklám” menüpont alatt található oldalt és ott is győződjünk meg róla, hogy megjelenik-e a „Hirdetések” felirat. Mindezt tegyük 100 db virtuális felhasználóval úgy, hogy fokozatosan, másodpercenként 1 újabb virtuális felhasználó lépjen be a tesztbe. Készítsünk róla egy kimutatást, hogy hogyan változik a válaszidő a terhelés növelésével! Első lépésként töltsük le a JMetert a <http://jakarta.apache.org/jmeter/> oldal „Download Releases” menüpontja alól. Válasszuk a bináris változatot és töltsük le a tömörített fájlt. Kicsomagolás után indítsuk el a „jakarta-jmeter-<verziószám>\bin” könyvtárban lévőjmeter.bat fájlt! Grafikus kezelőfelület fogad minket.

#### Mi lesz veled, weboldal?

A fenti feladat megvalósításához első lépésként létre kell hoznunk egy *Thread Groupot.* Ezt úgy tehetjük meg, hogy jobb egérgombbal kattintunk a baloldali *Test Plan* címkére, majd *Add -> Threads (Users) -> Thread Group*ot választjuk. Minden virtuális felhasználó logikailag egy folyamatként, szálként jelenik meg a tesztünkben. Ezeket fogja össze csoportba a *Thread Group*.  
  
Itt három dolgot állíthatunk be.  
A virtuális *felhasználók darabszám*át, a *ramp-up periódust* (mennyi idő alatt induljon el az összes szál) és hogy *hányszor* fusson le a *Thread Group* alatti rész.   
A *Number of Threadshez* írjunk 100 felhasználót!   
A *ramp-uphoz* szintén 100-at, hiszen, ha azt szeretnénk, hogy másodpercenként egy-egy újabb felhasználó is becsatlakozzon a tesztben, akkor 100 másodperc alatt biztosan elindul 100 szál. Ha itt mondjuk 10 másodpercet írnánk, akkor 10 másodperc elteltével mind a 100 felhasználó biztosan elindulna, de ebben az esetben 10/100, azaz egy tized másodpercenként lép be egy újabb user. Ha egy felhasználó befejezte munkáját, a szál automatikusan megszűnik.   
Ha mindezeket beírtuk, akkor a *Loop Countnál* hagyjuk az alapméretezett 1-es értéket, ezzel jelezve, hogy mindez csak egyszer fusson le.  
  
Ezután meg kell mondanunk a programnak, hogy látogasson el a Tesztelés a gyakorlatban kezdőoldalára.  
Ezt úgy tehetjük meg, hogy jobb egérgombbal kattintunk a Add -> Sampler -> HTTP Requestet választjuk. Itt a *Server name or IP-hoz*írjuk be, hogy [www.tesztelesagyakorlatban.hu](http://www.tesztelesagyakorlatban.hu/), port numberhez beírhatjuk, hogy *80*, de ez az alapméretezett http port is, úgyhogy hagyhatjuk üresen is.  
A Path-hoz írjuk be, hogy „/index.php” jelezve, hogy a kezdőlapot akarjuk meglátogatni. A többi mezőt hagyjuk üresen.  
  
Most vizsgáljuk meg, hogy megjelenik-e a „Magazin” felirat a kezdőlapon. Megint csak jobb egérgombbal kattintsunk a *Http Requestre,* majd *Add -> Assertations -> Response Assertation.* A megjelenő képernyőn válasszuk a Contains lehetőségeket a rádió gombok közül, majd kattintsunk az *Add* gombra. Itt a *Patterns to test* részhez kattintva pötyögjük be: „Magazin”. Ezzel megvizsgáltuk, hogy a kezdőlapon megjelenő „A Magazin” feliratnak valóban rész-e a „Magazin” felirat.  
  
Most jön a gondolkodási idő. A feladat „nagyjából 5 másodpercet” említ. Ezt egy *Gaussian Random Timerrel* oldjuk meg. Ez garantálja, hogy nagy valószínűséggel 5 másodpercet vár a JMeter, ettől nagyobb eltérésnek már kisebb a valószínűsége. Minél nagyobb eltérés mutatkozik az 5 másodperctől, annál kisebb a valószínűsége, hogy előfordul, a kitérés maximális mértékét pedig mi adjuk meg. Ez garantálja, hogy véletlenszerűen 5 másodperc körüli ideig vár. Kattintsunk jobb egérgombbal a Add -> Timers -> Gaussian Random Timer. A Constant Delayhez pedig 5000 milisecondot. Azaz 5 másodpercet várunk és adunk egy kis (tizedmásodperc) kilengési időt. Adjunk hozzá egy újabb Http Requestet a *Thread Grouphoz* a fent leírtak szerint, annyi különbséggel, hogy a *Path-hoz* „/reklam.php”-t írjunk. Ugyanígy az előzőekhez hasonlóan, adjuk hozzá egy Assertationt is „Hirdetés”-t írva a *Pattern to test* részhez. Így a fenti feladat tesztlépéseivel megvagyunk.



Szeretnénk azért látni valamilyen féle visszajelzést is, főleg, ha kimutatást kell készítenünk a válaszidőről. Úgynevezett *Listenereket* kell hozzáadnunk a szálcsoportunkhoz. Ezt már a jól ismert jobb egérgombos módszerrel *Add -> Listener -> View Results Tree* és az ugyanitt található *Add -> Listener -> Spline Visualizert* is adjuk hozzá. Az előbbi azért felel, hogy lássuk, http kéréseink vajon „betalálnak-e”, az utóbbi pedig interpolált görbét rajzol nekünk, hogy lássuk, hogyan változik a válaszidő az idő elteltével. Ha hozzáadtuk őket a *Thread Group*hoz, válasszuk a Run menüből a Startot! Ha a két Listener valamelyikére kattintunk futás közben, láthatjuk az eredményeket. A *View Result Treenél* a http kérések sikerességét, valamint magát a válaszokat láthatjuk akár HTML formátumban is. A Spline Visulizer pedig a válaszidő változását mutatja. Ha nem történt semmi hiba, valami hasonlót láthatunk, mint az ábrán:



#### Végszó

Javaslom, játszadozzunk el a felhasználószámmal és a ramp-up time-mal, figyeljük meg, hogyan változik a válaszidő! Túlterhelni az oldalt úgysem tudjuk, szerencsére gátat szab saját gépünk kapcsolata. Remélem, egy jó kis ízelítőt kaptunk a Performance Testingről, és kedvet kaptunk a JMeterhez is. Aki teheti, nézzen utána a képességeinek (funkcionális tesztre is alkalmas) és hasznosítsa munkája során.

Forrás: https://tesztelesagyakorlatban.hu/teljesitmenyteszteles/