HOUG
SZAKMAI NAP

JSF alkalmazások teljesítményhangolása JMeter és dynaTrace segítségével

Bakai Balázs
bakaibalazs@gmail.com
http://seamplex.blogspot.hu

2013. október 9.

Miről lesz szó?

- A JSF működése (röviden…)
- Terheléses tesztek készítése JSF alapú web-alkalmazásokhoz
- Problémák a terheléses tesztekkel
- Az alkalmazások gyenge pontjainak beazonosítása
- Nézzük meg a gyakorlatban…
- A performancia és stabilitási problémák megelőzése

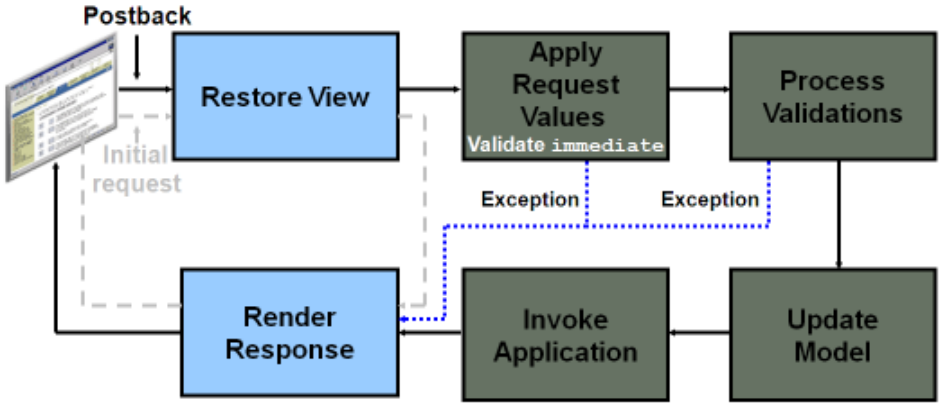
A prezentáció letölthető innen

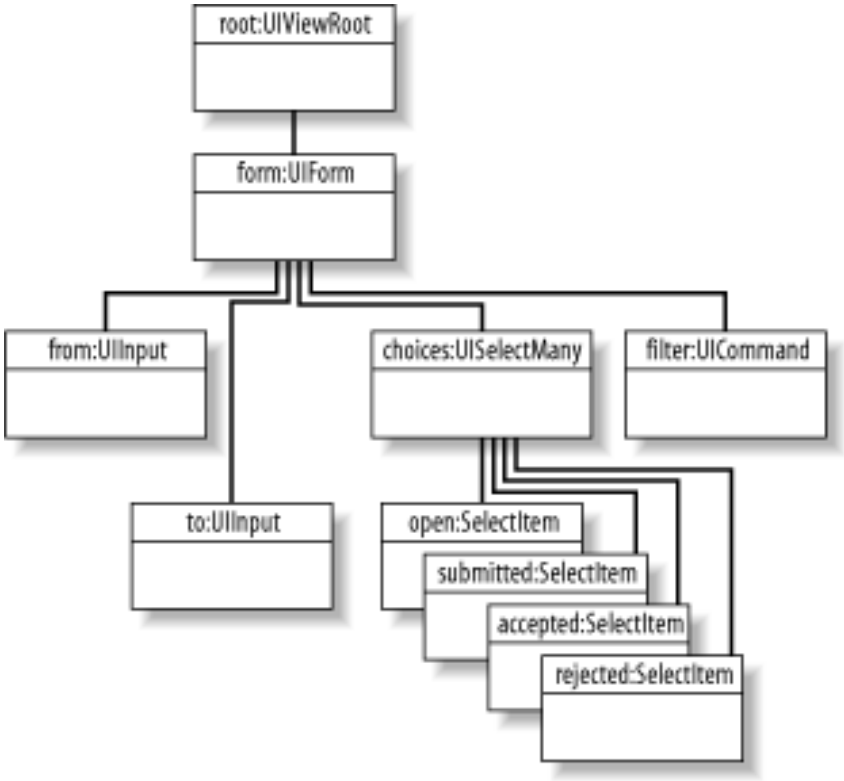
http://seamplex.blogspot.hu

A JSF működése (röviden…)

JavaServer™ Faces
JSF

A JSF működése (röviden…)





- Inital request (HTTP GET)
  - Az oldal kezdeti megjelenítése
- Postback request (HTTP POST)
  - Tipikusan a form-ok elküldése

A JSF működése (röviden…)

- Minden oldalhoz eltárolódik a komponensfa:
  - struktúrája (view tree)
  - állapota (view state)
- A komponensfát a kérések során valahol tárolni kell:
  - Kliens oldalon : Folyamatos szerializálás és deszerializálás a kérések során
  - Szerver oldalon : Http session
    - A postback kérésekkel továbbítani kell a view state id-t

<input id="javax.faces.ViewState" name="javax.faces.ViewState" type="hidden" value="abcdefghijklmn123456" />

A JSF FWK generálja, így az értéke megváltozhat

Terheléses tesztek készítése JSF alapú web-alkalmazásokhoz

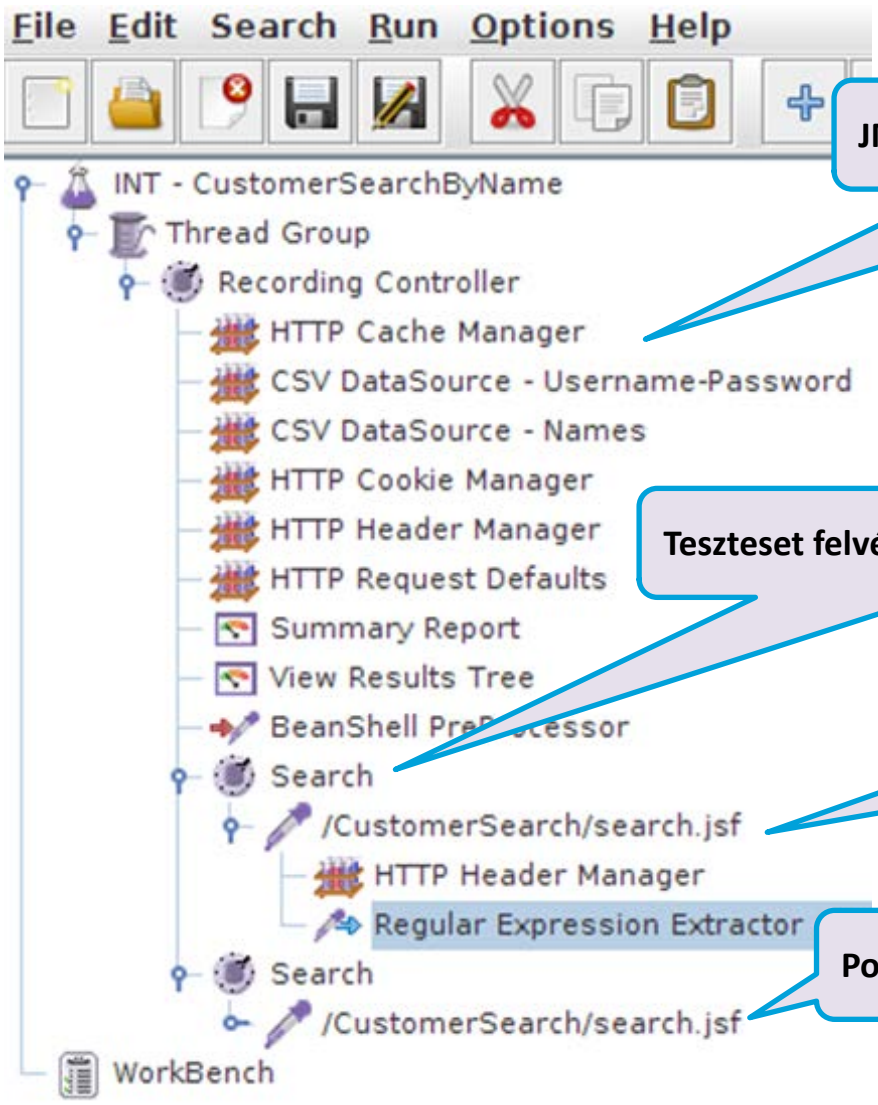
Apache JMeter

JMeter

- Kliens-szerver alapú alkalmazások teljesítményének mérésére használjuk
- Web-alkalmazások terheléses tesztelése
  - HTTP kéréseket küld és fogad
  - Nem hajtja végre a kliens oldali renderelést
  - Nem futtatja le a javascript-eket
- Tesztelhetünk még: Web-szolgáltatásokat, FTP, LDAP és DB szervert
- Chartok: Vizuálisan is követhetjük a válaszidők alakulását
- A beépített HTTP proxy támogatása révén
  - Automatikusan rögzíthetjük a böngészőn keresztül indított kéréseinket
  - Így később visszajátszhatjuk
  - Működése: Browser -> JMeter HTTP Proxy -> WebApp

http://seamplex.blogspot.hu/2012/02/jmeter-webalkalmazasok-terhelesi.html

A keresési teszteset felvétele



JMeter konfigurációs elemek felvétele

Teszteset felvétele a HTTP Proxyval

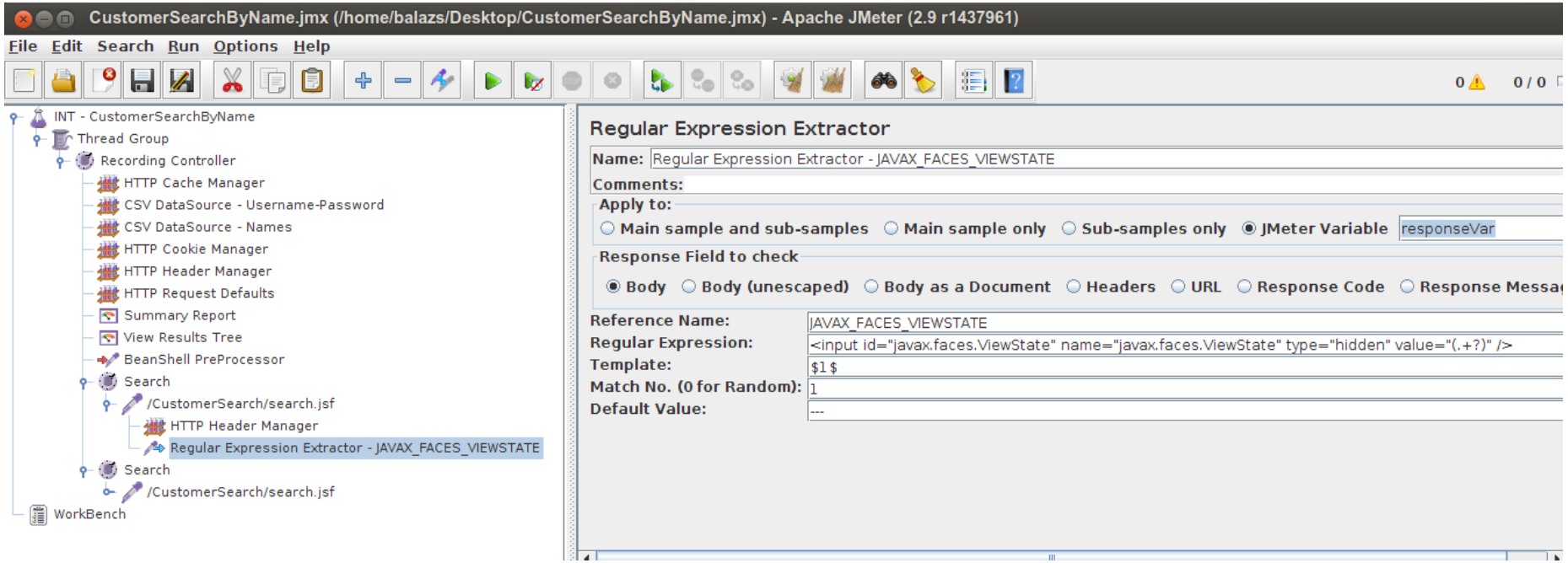
Initial request

Postback request

A feladat, hogy a JSF ViewState értékét szkriptelve aktualizáljuk

1. megoldás: Regular Expression Extractor

- JMeter Post Processor, a Regular Expression Extractor
  - Az initial kéréshez tartozó válasz forrásából kinyerjük a viewstate id-t
  - Eltároljuk a ${JAVAX_FACES_VIEWSTATE} változóba



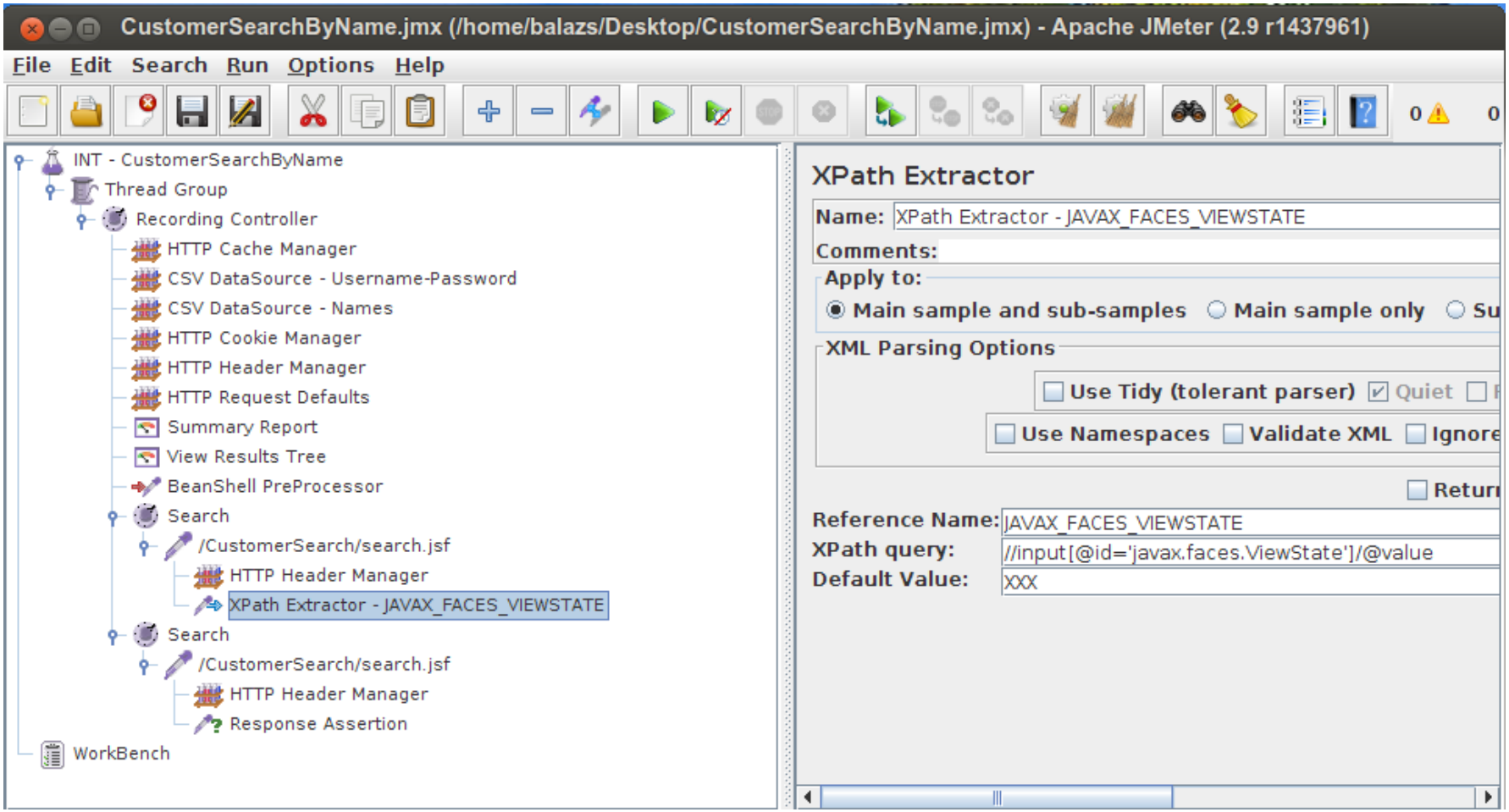
1. megoldás: Regular Expression Extractor

- Mi lehet ezzel a probléma?
- JSF-re épülő keretrendszerek néha több sorba is megtörik a viewStateId átadására használt rejtett input mezőt.

```
<input id="javax.faces.ViewState"
name="javax.faces.ViewState" type="hidden"
value="aCe2XnokY5W3qRKTuTgT12YSQ0gxrqOMV9PcIrDsUykah" />
```

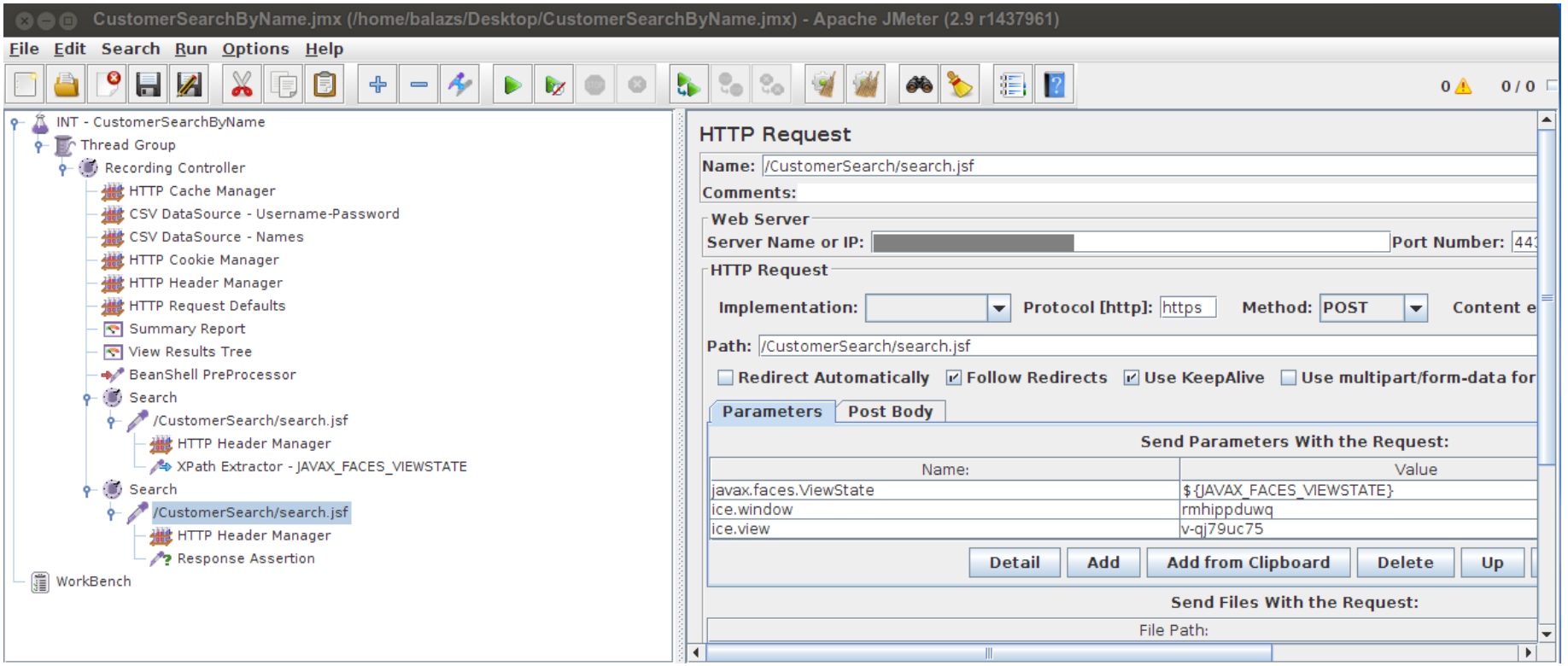
- Azonban a JMeter-es reguláris kifejezéseket nem tudunk több sorban elhelyezkedő inputra illeszteni!
- Mi lesz a megoldás?
- Használjuk az XPath Extractor Post Processor-t!

2. megoldás: XPath Extractor Post Processor



Terheléses tesztek készítése JSF alapú web-alkalmazásokhoz

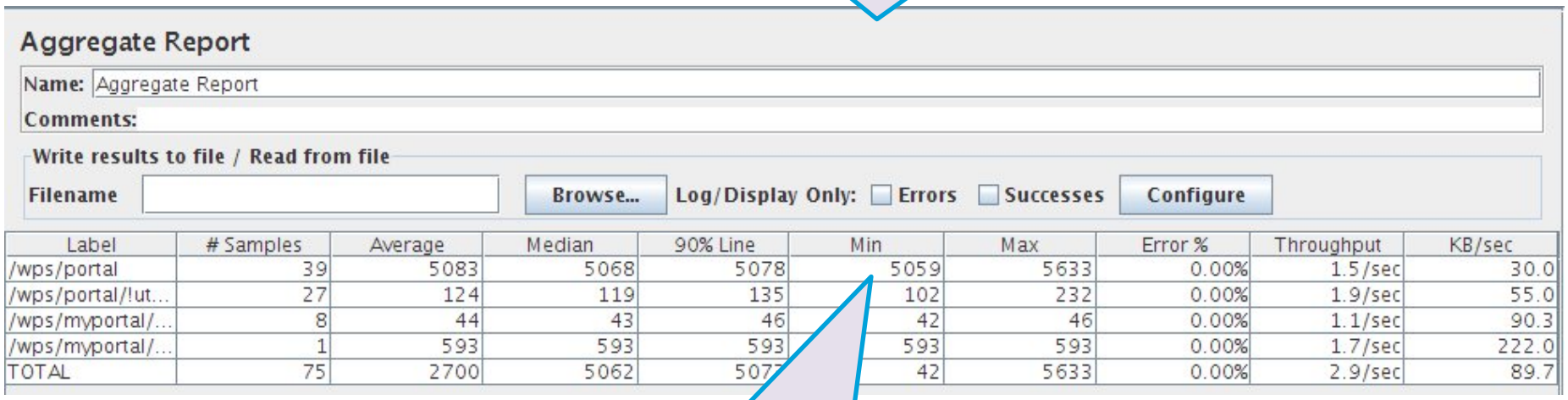
- A postback kérésnél cseréljük le a beégetett viewState értékét



Problémák a terheléses tesztekkel

Problémák a terheléses tesztekkel

Mit hiányolunk leginkább?



Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

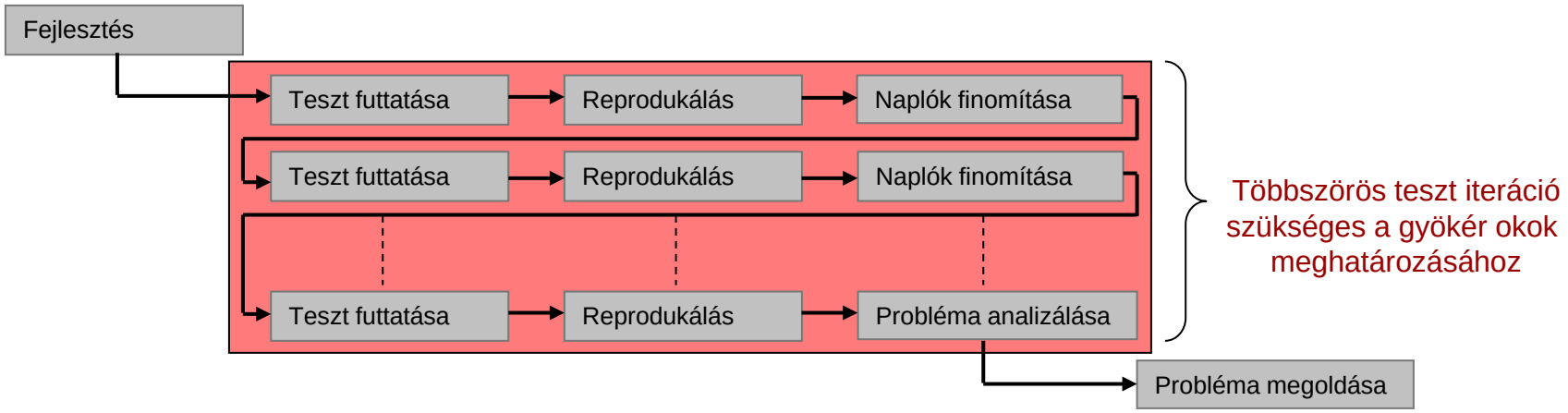
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/wps/portal	39	5083	5068	5078	5059	5633	0.00%	1.5/sec	30.0
/wps/portal/!ut...	27	124	119	135	102	232	0.00%	1.9/sec	55.0
/wps/myportal/...	8	44	43	46	42	46	0.00%	1.1/sec	90.3
/wps/myportal/...	1	593	593	593	593	593	0.00%	1.7/sec	222.0
TOTAL	75	2700	5062	507	42	5633	0.00%	2.9/sec	89.7

A kezdő oldal lassan töltődik be

De mi lehet ennek az oka?

Problémák a terheléses tesztekkel

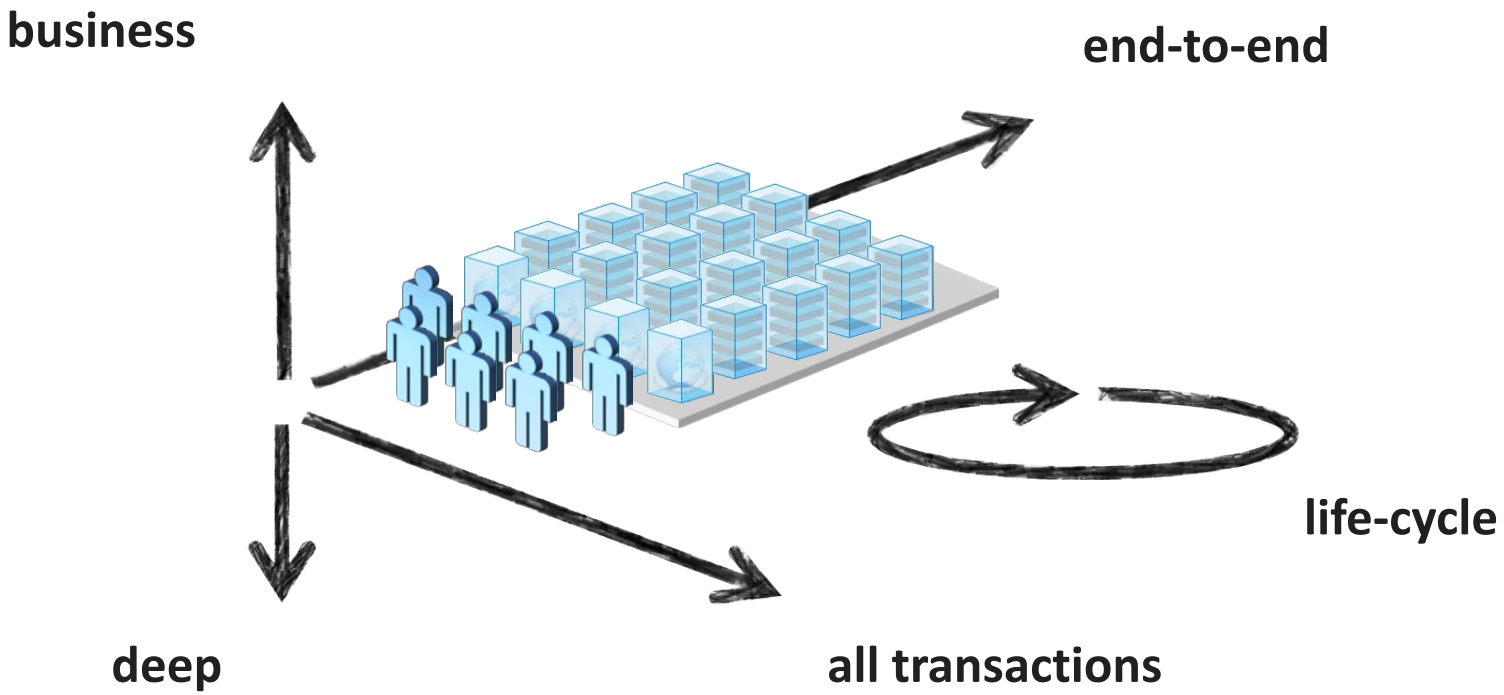
- A terheléses teszt-eszközök csak rámutatnak a ”lassú” oldalakra!
- Nem látjuk a magas válaszidők valósi okát!
- A tesztelt alkalmazásba nem látunk bele! (Black Box)
- A hiba beazonosítás folyamata erőforrás igényes:



Az alkalmazások gyenge pontjainak beazonosítása

COMPUWARE APM
DYNATRACE | GOMEZ

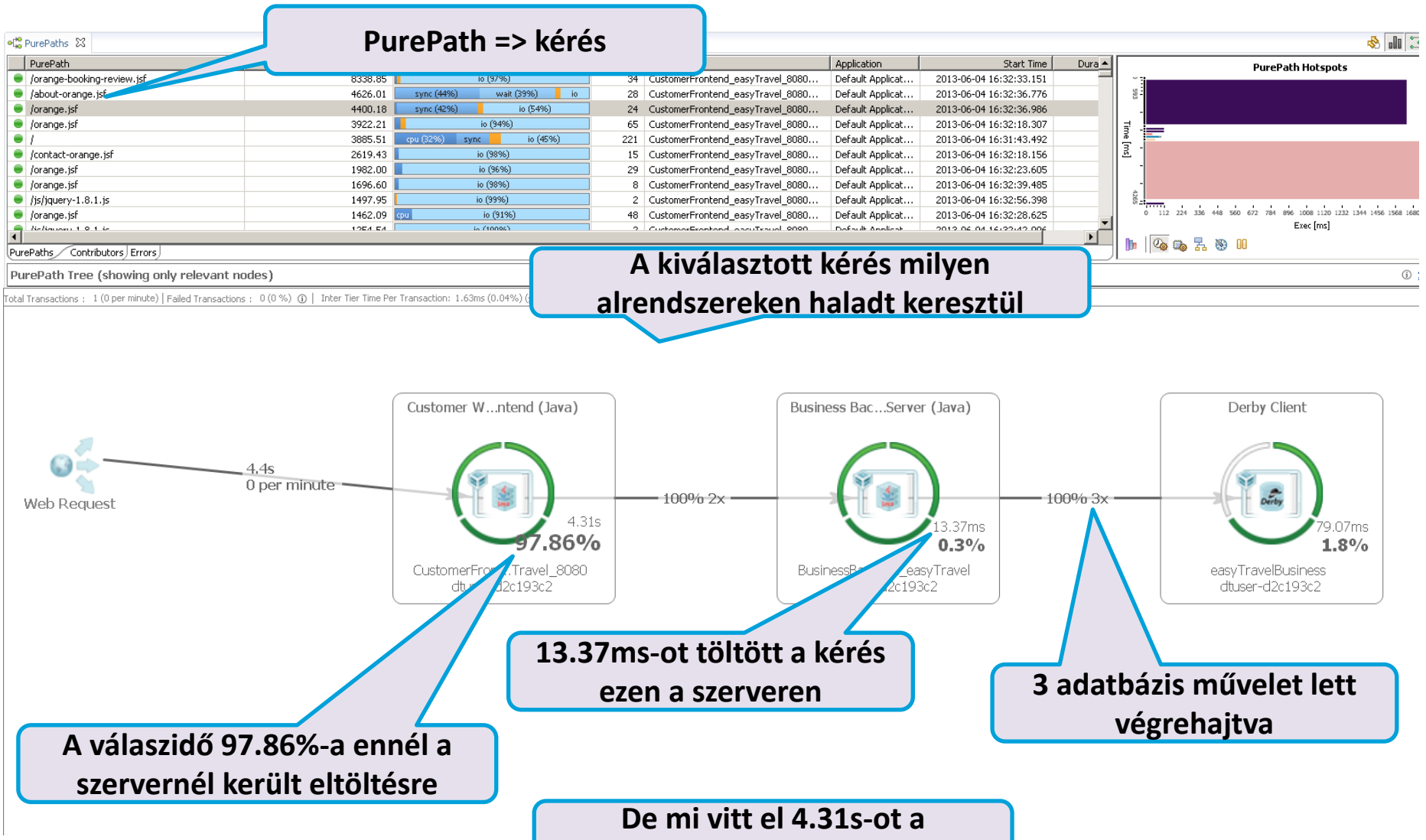
Az alkalmazások gyenge pontjainak beazonosítása



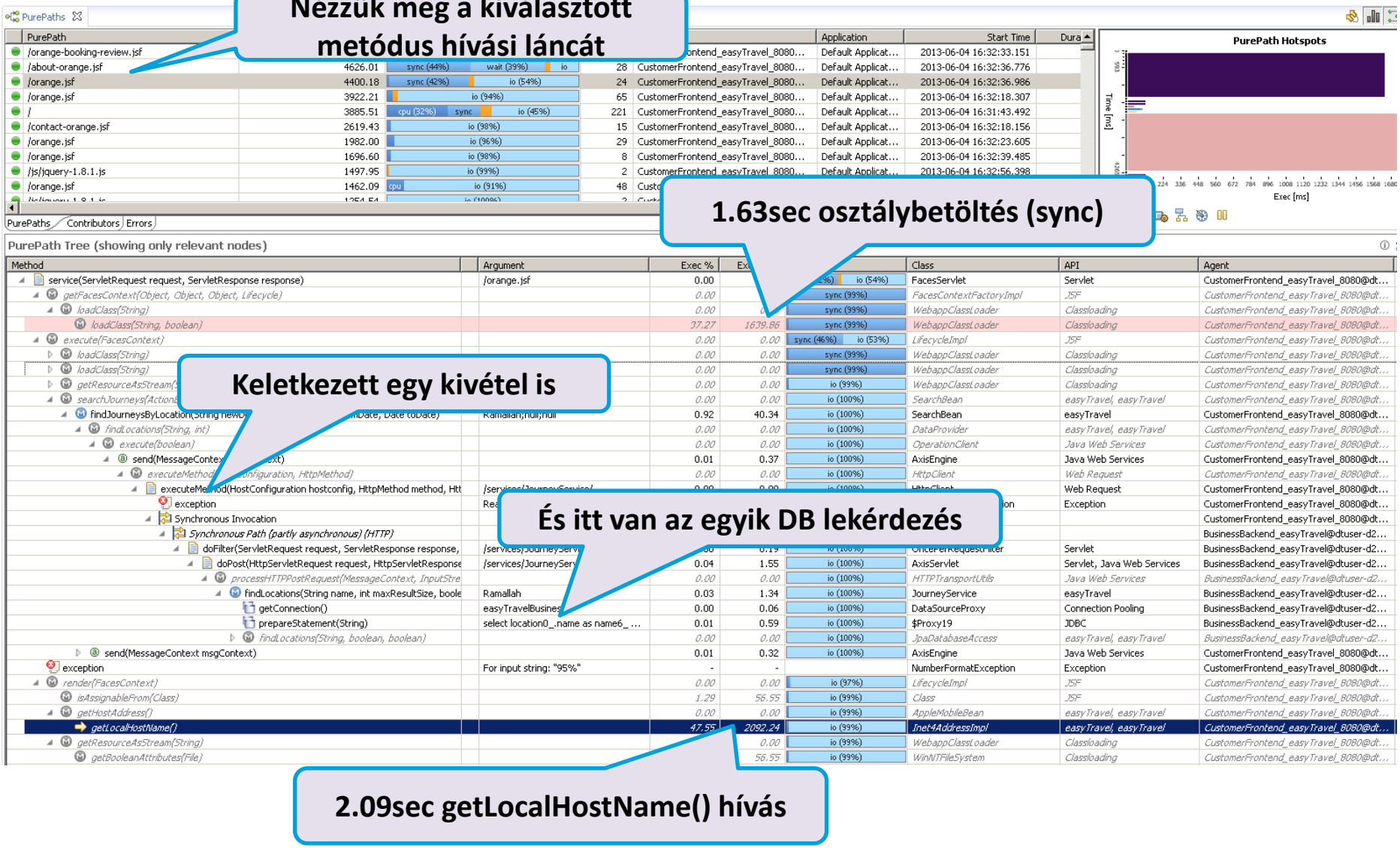
- Java, .NET, PHP, C/C++ támogatás
- Agent alapú, a használatához nem kell a forráskódot módosítani
- 4% alatti overhead miatt, éles környezetben is használható
  - A hibákat nem kell reprodukálni a tesztkörnyezetben!

http://seamplex.blogspot.hu/2013/08/dynatrace-te-meg-nem-hasznalod.html

Az alkalmazások gyenge pontjainak beazonosítása



Az alkalmazások gyenge pontjainak beazonosítása

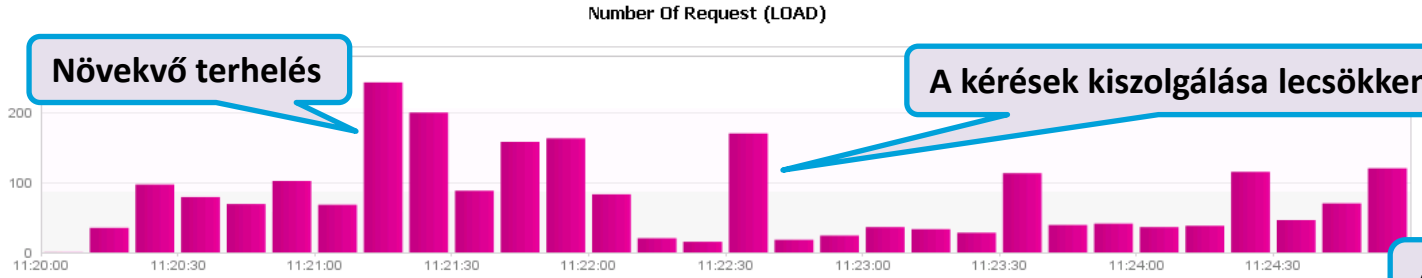


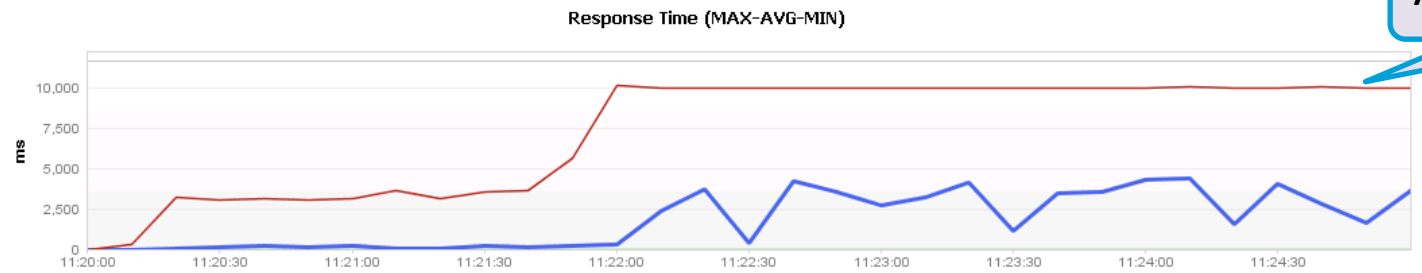
Nézzük meg a gyakorlatban

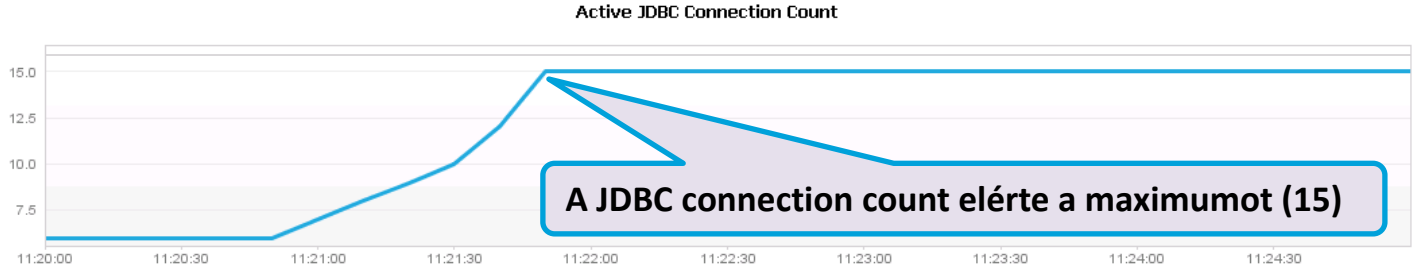
Nézzük meg a gyakorlatban

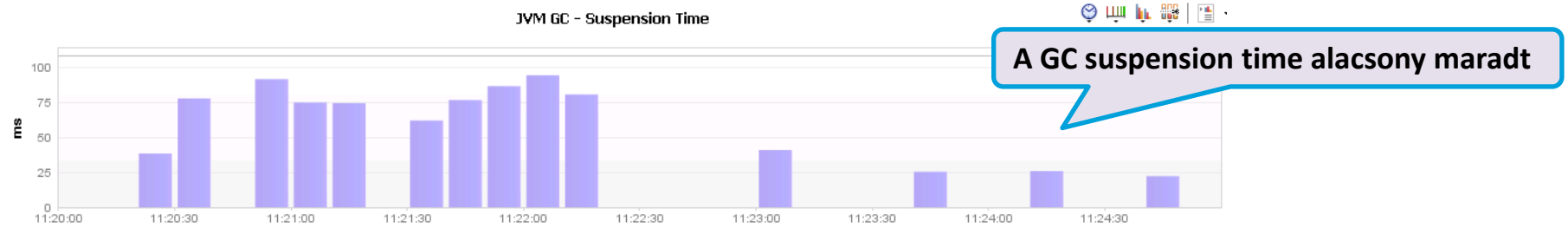
- Egy JSF alkalmazáshoz, JMeter terheléses teszt készült!
- Feladat: a rendszer szűk keresztmetszeteinek a beazonosítása!

Terhelés: 10 felhasználó/perc

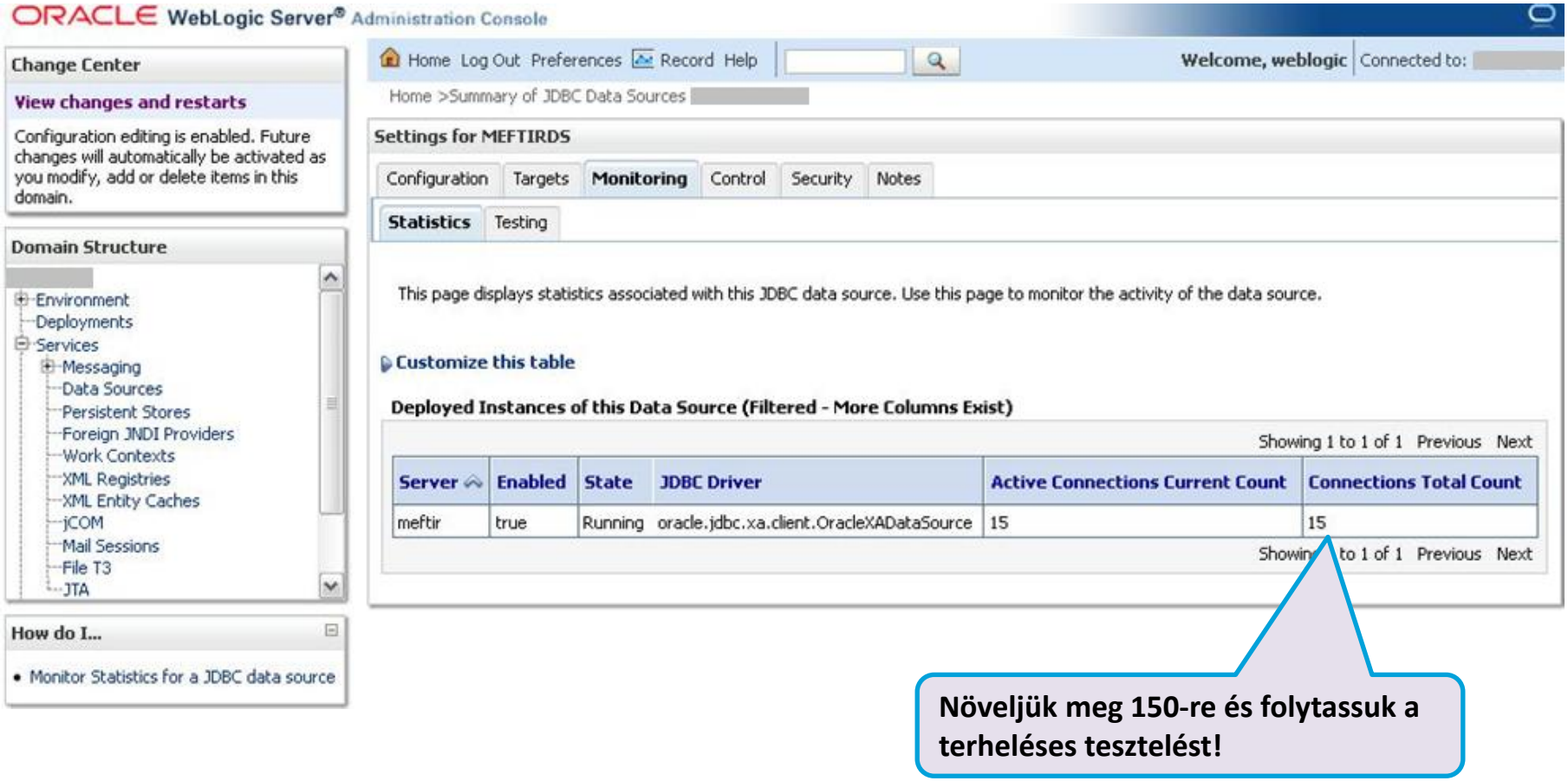




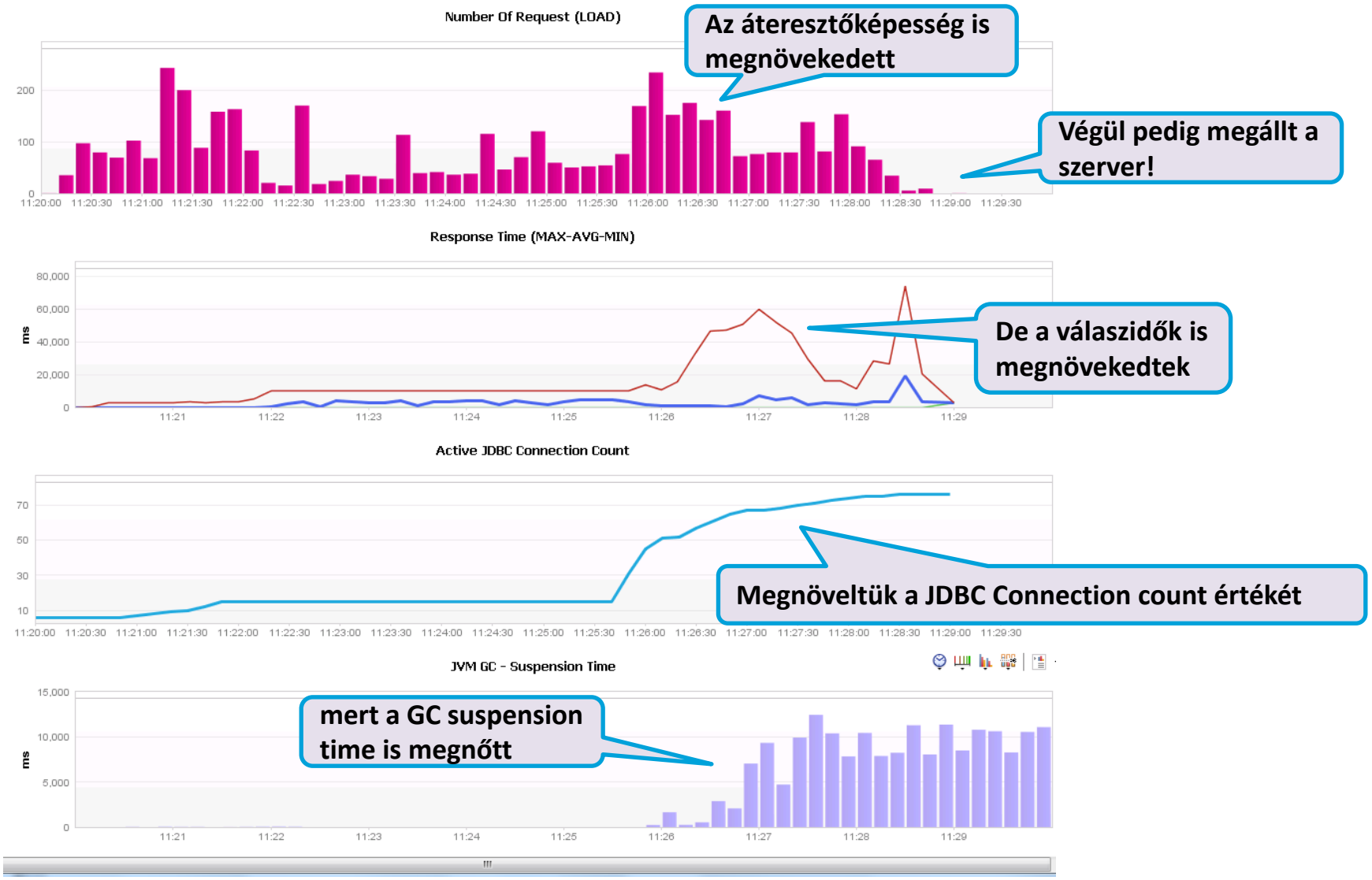




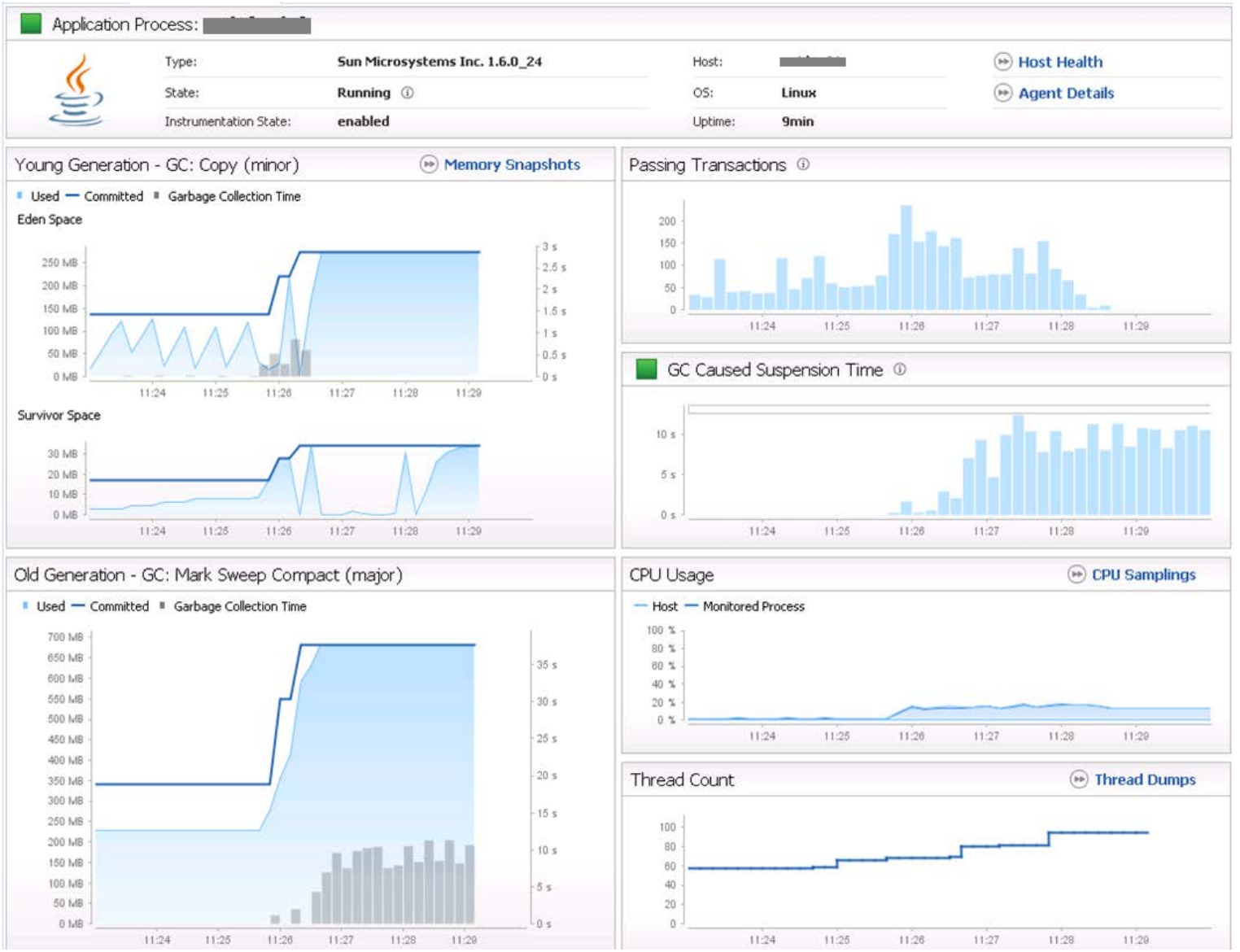
A probléma oka: A JDBC max Connection count túl alacsony volt



Terhelés: 10 felhasználó/perc + JDBC connection count = 150



JVM információk



A performancia és stabilitási problémák megelőzése

Jenkins

A performancia és stabilitási problémák megelőzése

- Használjunk folyamatos integrációs eszközt (Jenkins)
  - A tesztek automatizált futtatása + integrálva a dynaTrace-szel: dinamikus jellemzők
  - Automatizált kód-minőség ellenőrzés (SonarQube) : statikus jellemzők

- Napi szinten ill. release-enként lekövethetők a trendek (dinamikus és statikus jellemzők)
- A problémákat így hamarabb kiderülnek
- Ezért kevesebb erőforrás ráfordítással javíthatók!

Q&A

Köszönöm a figyelmet!

balazs.bakai@telvice.hu