Kliens oldali virtualizáció 2 – Gyakorlat

1. Seamless window mode kipróbálása VMware Workstation alatt
   1.1. Indítsuk el az *eclipse-vm* Windows XP virtuális gépet!
   1.2. Indítsuk el benne a Calculatort, majd kapcsoljunk át Unity módba!
   1.3. Milyen folyamatokat látunk a gazdagépen? Látunk-e calc.exe-t, miért?

2. Virtual Appliance létrehozása
   2.1. Indítsuk el a *VMware_Studio* virtuális gépet!
   2.2. Az eclipse-vm virtuális gépben egy böngészőből navigáljunk el a VMware Studio címére, és ismerkedjünk a felülettel.
   2.3. Indítsuk el az eclipse-vm virtuális gépben az Eclipse-t (C:\eclipse)!
      2.3.1. Korábban fel lett az Eclipse-be telepítve a VMware Studio pluginje[1], ami elérhető a következő update site-ról: `http://<StudioIPaddr>/eclipse/update/`
   2.4. A *VMware Studio Exlorer*ben nézzük meg, hogy milyen elemek és erőforrások érhetőek le jelenleg a VMware Studioban!
   2.5. Csatlakozzunk a *VMware Studio Web Console*-hoz! Nézzük meg a *test-vm* virtuális gép profilt! Módosítsuk az üdvözlő üzenetét, állítsuk be, hogy 300 MB memóriát kapjon, hozzunk létre egy test nevű felhasználót, és adjuk hozzá az NTP csomagot.
   2.6. Most nézzük meg a *test-vApp* tulajdonságait. Mit lehet egy vApp esetén beállítani?
   2.7. (Opcionális) Gyártsuk le a test-vApp-ot!

3. VMware Integrated Virtual Debugger használata
   3.1. Indítsuk el a virttech könyvtárban lévő Eclipse-t (eclipse-linux) a gazdagépen. Nyissuk meg a *vmware-remote-debug* workspace-ben lévő *test-app* alkalmazásunkat. Nekünk ezt teljesen jól működik, de állítólag az egyik felhasználónál hibás volt egy windowsos gépen. Biztos a felhasználóban van a hiba, de azért ellenőrizzük ezt
   3.2. Hozzunk létre egy új *Run Configuration*t, amiben az *eclipse-vm* virtuális gépben futtatjuk le a *ThreadedApplication* Java alkalmazásunkat. Futtassuk is le! Látunk-e valami hibát (helyes működés esetén a számlálónak folyamatosan nőnie kell).
   3.3. Próbáljunk meg felvenni, majd visszajátszani egy hibás lefutást a Workstation *Record Execution* funkciójával. Mire jó a marker hozzáadása? Próbáljuk ki azt is.
   3.4. A 6.5-ös Workstationhöz való eclipse-es Integrated Virtual Debugger sajnos nem tudja a reverse debuggingot, meg különben is az Visual Studio alatt is csak natív C/C++ kóddal megy. (7-es Workstation esetén már van legalább Linux gdb támogatás. Azért érdemes ránézni a leírásra, csak, hogy érezzük, hogy a replay debuggingot mennyire nem triviális feladat lehetett megoldani: Replay Debuggin on Linux)
   3.5. Úgyhogy csak annyit fogunk kipróbálni, hogy hogyan megy az élő debug. Hozzunk létre egy új *Debug Configurationt*, ami elindítja az alkalmazásunkat (*VMware execute Java application*). Állítsunk be egy töréspontot, és próbáljuk ki, hogy működik-e! Ha nem (erre azért van esély:), akkor a *Windows / Show view / Error log* ablakban találunk utalást, hogy mi volt a gond.

[1] lásd: http://www.vmware.com/support/developer/studio/studio21/eclipse_plugin.html

3.6. Próbáljuk a hibakeresésben is megtalálni a hiba okát!

3.7. Végezetül nézzük meg, mi történik a háttérben! Hozzunk létre egy olyan *Debug Configuration*t, ami csak csatlakozik egy már futó folyamathoz. Ehhez a Java alkalmazásunkat úgy kell elindítani a távoli gépen, hogy engedélyezve legyen a távoli debug benne. Ehhez a java.exe-t a következőképpen kell paraméterezni (ezután még értelemszerűen meg kell adni a futtatandó class fájlt):

```
java.exe -Xdebug -Xnoagent -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000
```

Letöltések

- VMware Workstation 30 napos kipróbálható verzió: https://www.vmware.com/tryvmware/?p=workstation-w
- VirtualBox (ingyenes, és a seamless window módot itt is ki lehet próbálni): http://www.virtualbox.org/