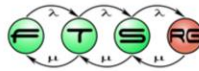


Felügyeletre tervezés

Kocsis Imre, Micskei Zoltán



Utolsó módosítás: 2014.04.13.

Felügyeletre tervezés

- Szoftverkomponensek egy IT rendszerben:
 - Komponensek felügyelhetősége általában korlátos
 - Felügyelhetőségre tervezés: erős ipari nyomás
 - MS MOF, IBM Autonomic Computing
 - DevOps

- Felügyeletre tervezéshez szükséges
 - Támogató API-k és platform mechanizmusok
 - ***Felügyeleti modell***

Miért foglalkozunk ezzel?

Why Manageability?

- Manageability is increasingly the differentiator between product offerings
 - Low end needs consistency and simplicity
 - Virtualization drives scale
 - Operations demand agility, quality and repeatability

Forrás: Jeffrey Snover , Refaat Issa: **Make your product manageable**, SAC-644T, BUILD 2011.
<http://channel9.msdn.com/Events/BUILD/BUILD2011/SAC-644T>



Miért lesz ez nekünk jó?

Szokásos szoftver kiadás:

Milyen jó lenne:



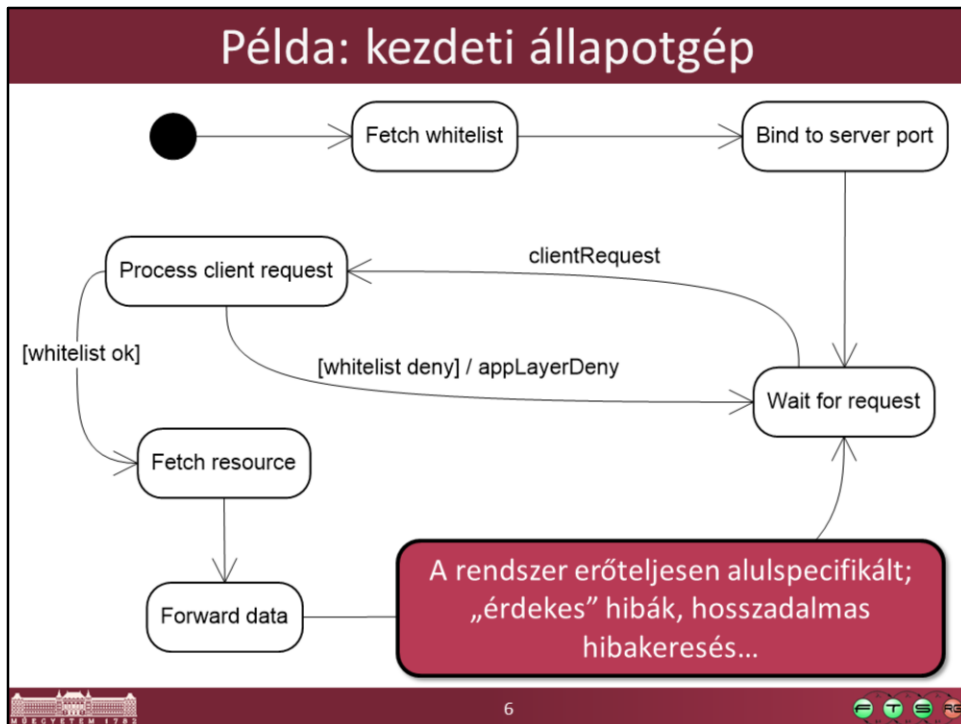
Forrás: [http://dev2ops.org/What is DevOps?](http://dev2ops.org/What%20is%20DevOps?)

- Nem csak fut/nem fut látszódná az alkalmazásból
- Hibát jelezné, és automatikusan lehetne beavatkozni
- Alkalmazás jelezné a növekvő terhelést, automatikusan új kiszolgálót rakhatnánk alá
- ...

Kép forrása: <http://dev2ops.org/blog/2010/2/22/what-is-devops.html>

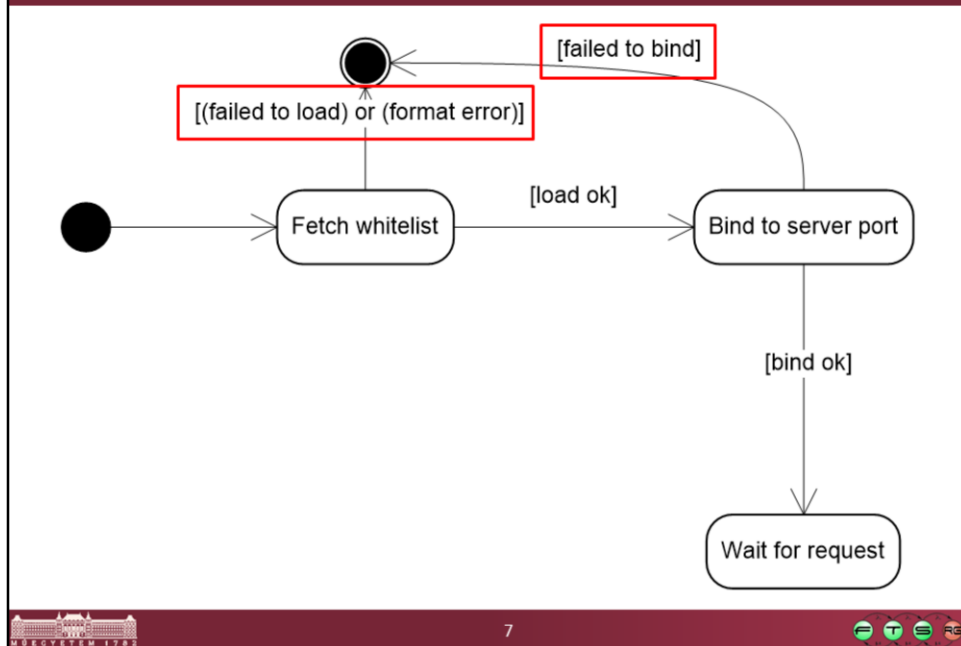
Felügyeletre tervezés - példa

- **Figyelem:** felkészülés a házi feladatra
- **Felügyelt SW komponens:** whitelisting HTTP proxy
 - Kliensek internet-hozzáféréseinek szűrése
 - Egyszerűsítés: egyszálú, nincs perzisztens HTTP, nincs SSL...
- **Analízis/modellezés:** felügyelethez fontos
 - állapotok,
 - események,
 - metrikák és tulajdonságok,
 - beavatkozási lehetőségek meghatározása.



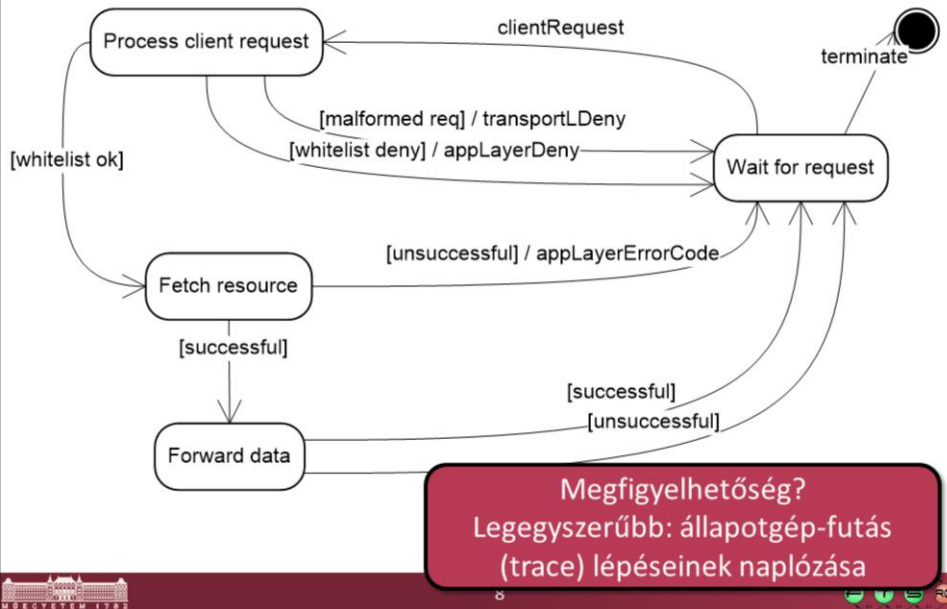
Első lépésként modellezzük valahogy a rendszer működését. A helyes működés leírása azonban még kevés, az ez alapján fejlesztett komponenst még nem nagyon tudnánk beilleszteni a meglévő felügyeleti rendszerünkbe, mert szinte csak működik/nem működik állapotokat látunk belőle kívülről.

Példa: hibaesetek specifikálása

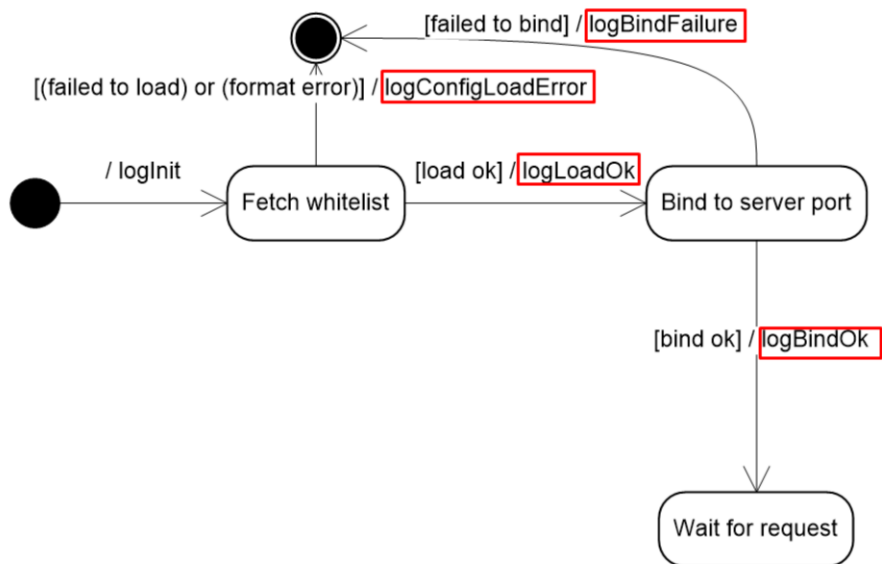


Elkezdjük a hibaeseteket számba venni, és definiáljuk, hogy milyen hibás események hatására milyen állapotba kerüljön a rendszerünk.

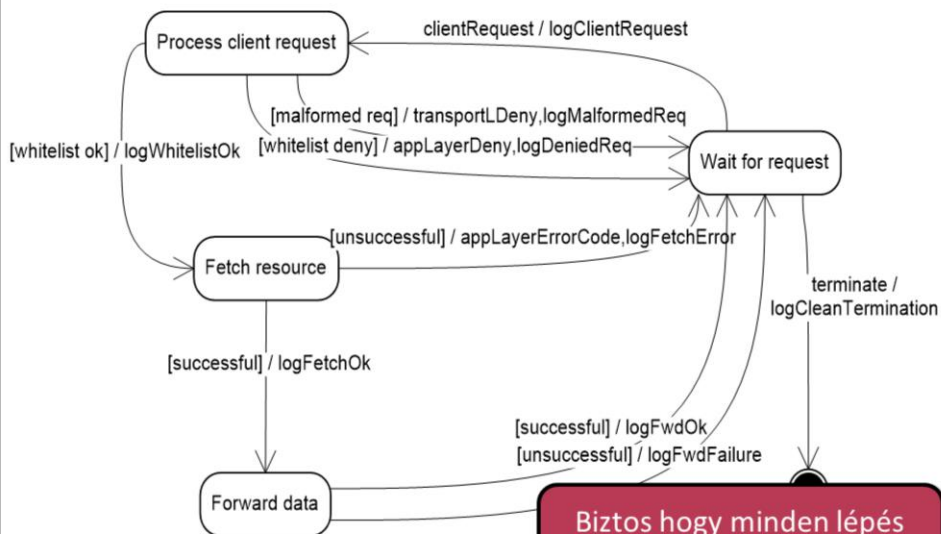
Példa: további hibaesetek



Példa: kiegészítés naplózással



Példa: kiegészítés naplózással



Biztos hogy minden lépés ugyanolyan fontos?

Log események kategorizálása - példa

Kategória	Leírás
Critical	Fatal error or application crash.
Error	Recoverable error.
Warning	Noncritical problem.
Information	Informational message.
Verbose	Debugging trace.
...	

(.NET Framework: System.Diagnostics.TraceEventType)

Log események kategorizálása - példa

- logInit **Information**
- logLoadOk **Information**
- logConfigLoadError **Critical**
- logBindOk **Information**
- logBindFailure **Critical**
- logClientRequest **Verbose** (inkább, mint Information)
- logMalformedReq **Error**
- logDeniedReq **Error**
- ...

Példa: metrikák

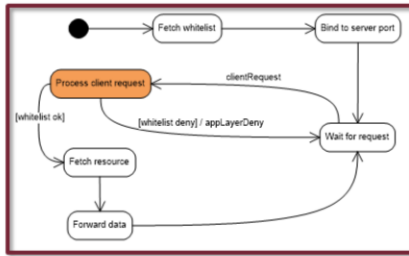
- **Funkció: kiszolgáló folyamat**
 - Uptime
- **Funkció: kérés-válasz kiszolgáló**
 - Beérkezett kérések száma
 - Sikeresen kiszolgált kérések száma
- **Funkció: HTTP proxy**
 - Rosszul formáltság miatt eldobott kérések száma
 - HTTP hibakód miatt nem kiszolgált kérések száma
 - Nem whitelist-be eső kérések aránya

1. Uptime kivételével: valamilyen csúszóablakra nézzük
2. Az utolsó metrika: inkább adatbiztonsági aspektus
3. A többi egy része is átnyúlik a szolgáltatásbiztonságba

További példa metrikák:

- Elfogadott kérések száma
- Visszautasított kérések száma
- Elfogadott, de nem kiszolgált kérések száma
- → kiszolg. TCP szintű hiba miatt nem kiszolgált kérések száma
- → kliens TCP szintű hiba miatt nem kiszolgált kérések száma

Példa: Futás közbeni állapot



Monitorozás / Lekérdezés

State: ProcessRequest
Uptime: 00:03:14
NumRequests: 15
FailedRequest: 0
...

Naplózás

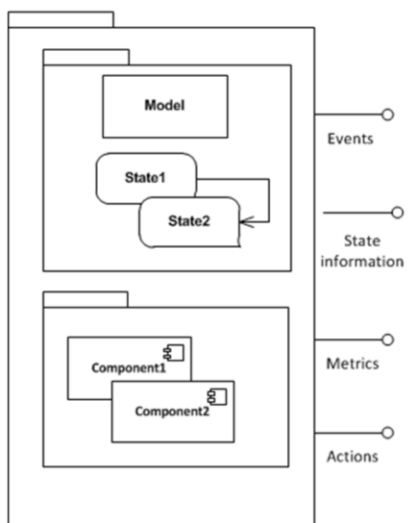
```
2010.03.20.;10:34;100;Information;Binding to port 80
2010.03.20.;10:34;101;Information;Waiting for request
...
2010.03.20.;10:35;102;Verbose;Request arrived
2010.03.20.;10:35;105;Error;Malformed request
```

Példa: Felügyeleti akciók

- **Terminálás**
 - A modell már tartalmazza
- **Whitelist-állomány újratöltése**
 - Ehhez a modellt is módosítani kellene!
- **A főbb nem megengedett, de kért oldalak listájának lekérése (+ gyakoriság, IP címek, ...)**

Ugye látszik, hogy bonyolultabb esetben mindezt (pl.) UML-ben ragadtuk volna meg?

Példa: hogy állunk most?



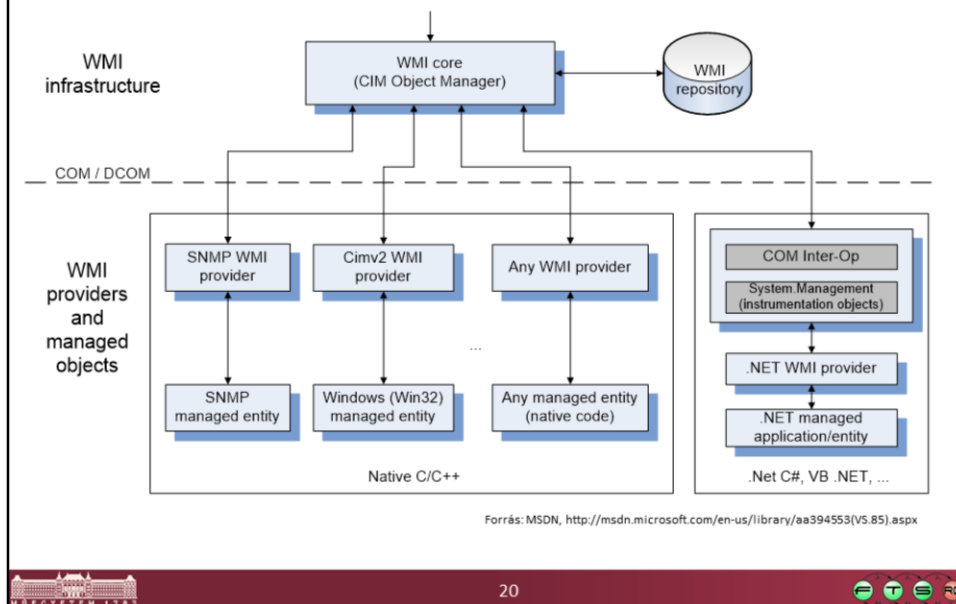
Integráció a rendszerfelügyelethez?

- **Naplózás:** naplózó (osztály)könyvtárak
 - Kulcsrakész integráció platformszintű mechanizmusokhoz (pl. Event Log, WMI)
 - Hova naplózzon: konfiguráció, nem kód!
 - Példák: **MS Enterprise Library**, log4j, ...
- **Metrikák lekérdezhetősége, műveletek**
 - Jellemzően platform támogatás kihasználása
 - Példák: CIM szolgáltató készítése (pl.: .NET WMI provider), **Java Management Extensions (JMX)**

Tartalom

- Felügyeletre tervezés
- Mintapélda: felügyeleti modell elkészítése
- Felhasználható technológiák
 - **WMI szolgáltató készítése**
 - MS Logging Application Block
 - Java Management Extensions (JMX)

WMI szolgáltatók (privoder)



Forrás: MSDN. WMI Architecture, [http://msdn.microsoft.com/en-us/library/aa394553\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa394553(VS.85).aspx)

Megjegyzés: ez még a Windows 8 előtti állapot, Windows 8-ban bevezettek egy új natív API-t (MI API), amivel C/C++ nyelven lehet szolgáltatót írni COM használata nélkül.

WMI szolgáltató készítése .NET-ben

- **Adatok megadása: attribútumokkal**
 - WmiConfiguration: névtér megadása
 - ManagementEntity: osztály dekorálása
 - ManagementProbe: WMI-ből olvasható tulajdonság
 - ManagementTask: WMI-ből elérhető metódus
- **Példányosítás:**
 - **Singleton**: csak egy példánya van
 - **Multi-instance**: ManagementKey adja meg, hogy melyik példány kell nekünk
- **Futtatási mód:**
 - **Coupled**: WMI szolgáltatás folyamatán belül
 - **Decoupled**: külön folyamatban, az alkalmazással együtt
- Lásd a „Felügyeletre tervezés” segédletben



MSDN. WMI Provider Extensions, <http://msdn.microsoft.com/en-us/library/bb404670.aspx>

Egyszerű WMI szolgáltató C#-ban

```
// provider will be in the root/MortgageCalc namespace.  
[assembly: WmiConfiguration("root/MortgageCalc", HostingModel = ManagementHostingModel.Decoupled)]  
  
// This is the installer class that installs an instrumented assembly.  
[System.ComponentModel.RunInstaller(true)]  
public class TheInstaller : DefaultManagementInstaller  
{ }  
  
namespace DecoupledWMIProvider  
{  
    // Use the ManagementEntity attribute on the class to specify that this is a provider  
    [ManagementEntity(Name = "MortgageCalc")]  
    [ManagementQualifier("Description", Value = "Allows you to read and set configuration settings.")]  
    public class MortgageCalcWMIProvider  
    {  
        // Specify that a property is a read/write property in the provider.  
        [ManagementConfiguration]  
        public double highestLoanAmount  
        {  
            get  
            {  
                return this.highestLoanAmount;  
            }  
        }  
    }  
}
```

Hosting
modell

Szolgáltató
neve

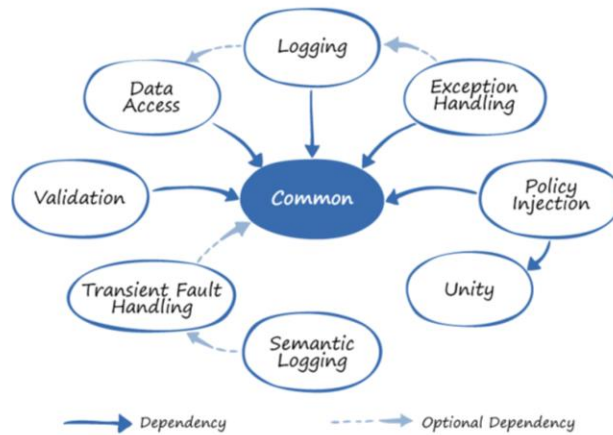
WMI-ből
lekérdezhető
tulajdonság

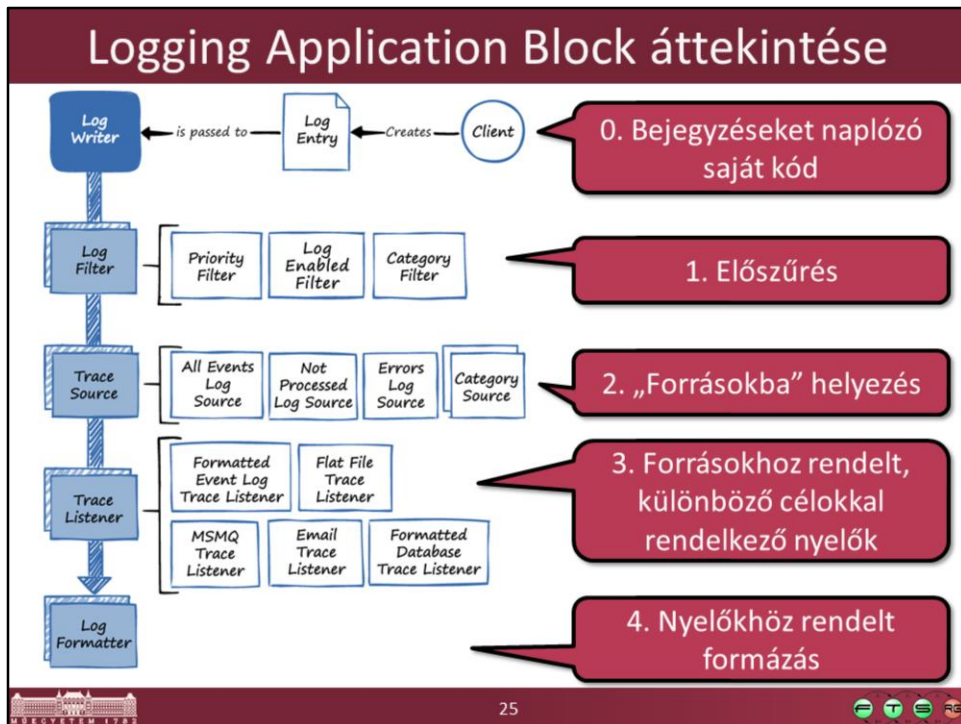
Tartalom

- Felügyeletre tervezés
- Mintapélda: felügyeleti modell elkészítése
- Felhasználható technológiák
 - WMI szolgáltató készítése
 - **MS Logging Application Block**
 - Java Management Extensions (JMX)

MS Enterprise Library

- Patterns & Practices csapat
- Tipikus feladatokra bevált megoldások gyűjteménye





Az ábra forrása: [http://msdn.microsoft.com/en-us/library/dn440731\(v=pandp.60\).aspx](http://msdn.microsoft.com/en-us/library/dn440731(v=pandp.60).aspx)

Log bejegyzések létrehozása

```
static void Main(string[] args)
{
    LogWriter defaultWriter = (new LogWriterFactory()).Create();

    LogEntry entry = new LogEntry();
    entry.EventId = 10;
    entry.Severity = TraceEventType.Error;
    entry.Message = "LoggingTest program initialized";

    defaultWriter.Write(entry);

    Console.WriteLine("Hello logging world");
}
```

Bejegyzés: LogEntry példány vagy közvetlenül a Write() változatai

- Elkérjük a konfiguráció által megadott aktuális LogWriter-t
- Beállítjuk a bejegyzés tulajdonságait
 - ID-t mindig adjunk meg!
- Kiíratjuk a log bejegyzést

FIGYELEM: a kódban sehol sem adtuk meg, hogy hova naplózzunk, azt a konfigurációs fájl határozza meg futási időben!

Enterprise Library Configuration Tool

The screenshot displays the Enterprise Library Configuration Tool interface. The title bar reads "Enterprise Library Configuration - C:\Users\meres\Documents\Visual Studio 2010\Projects\LoggingTest\LoggingTest\App.config". The menu bar includes "File", "Blocks", "Wizards", and "Environments". The left sidebar shows "Blocks Configuration" with expandable sections for "Application Settings", "Database Settings", and "Logging Settings".

The main workspace is divided into three panes:

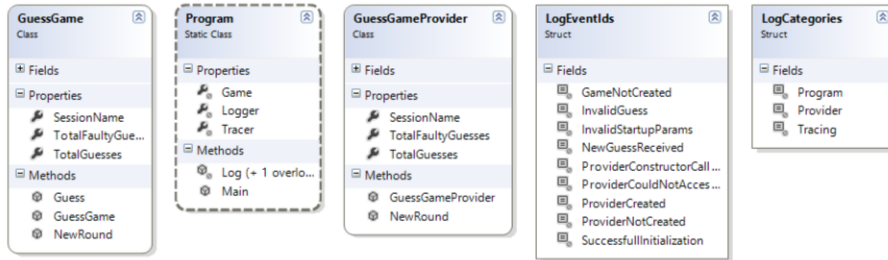
- Categories:** Shows a tree view with "General" selected. Below it, a table lists properties: "Name" (General), "Auto Flush" (True), "Listeners" (Name, Event Log Listener, Flat File Trace Li), and "Minimum Severity" (All). Below the table are "Special Categories" (All Events, Unprocessed Category, Logging Errors & Warnings) and "Logging Filters".
- Logging Target Listeners:** Shows "Event Log Listener" selected. Properties include: "Name" (Event Log Listener), "Formatter" (Text Formatter), "Log Name" (Application), "Machine" (.), "Severity Filter" (All), "Source Name" (LoggingTest), "Trace Output" (LogicalOperationStack, DateTime, Timestamp, ProcessId, ThreadId, Callstack), and "Type Name" (FormattedEventLogTraceListener). A "Flat File Trace Listener" is also visible below.
- Log Message Formatters:** Shows "Text Formatter" selected.

A red callout box in the bottom right of the interface contains the text: "Grafikus szerkesztő az XML-hez".

At the bottom of the window, there is a status bar with the number "27" and several icons.

DEMO Logging Application Block

- Log esemény kiírása Eseménynaplóba
- Átkonfigurálás:
 - naplózás egy szöveges állományba is



Tanácsok naplózáshoz

- Tervezzük meg előre az eseményeket!
- Mindig legyen ID-ja egy bejegyzésnek!
- Ne kódból, hanem konfigurációból állítsuk, hogy pontosan hova naplózzon!
- Súlyosság / prioritás / kategória egymástól független beállítási lehetőségek

Linkek

- Microsoft Enterprise Library 6.0
<http://msdn.microsoft.com/en-us/library/dn169621.aspx>
- Letölthető, teljes dokumentáció
<http://entlib.codeplex.com/>
- Lásd még a *Felügyeletre tervezés* segédletet

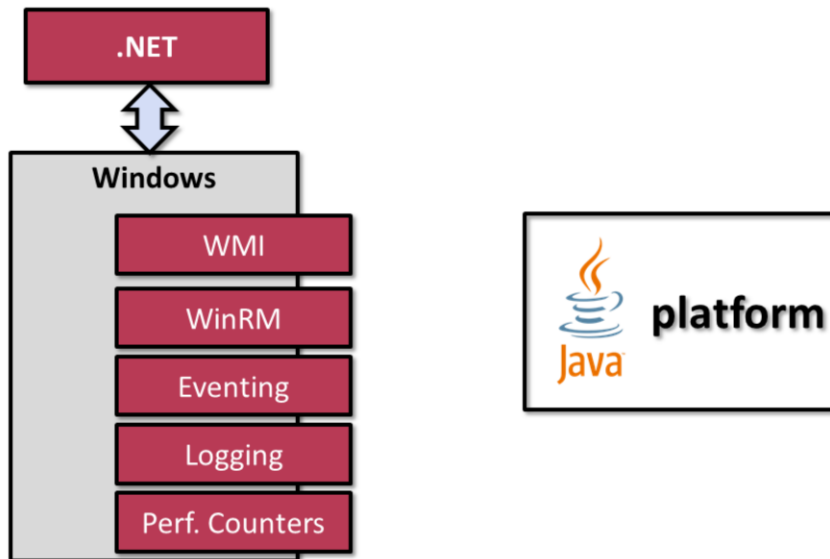


<http://msdn.microsoft.com/en-us/library/dn169621.aspx>
<http://entlib.codeplex.com/>

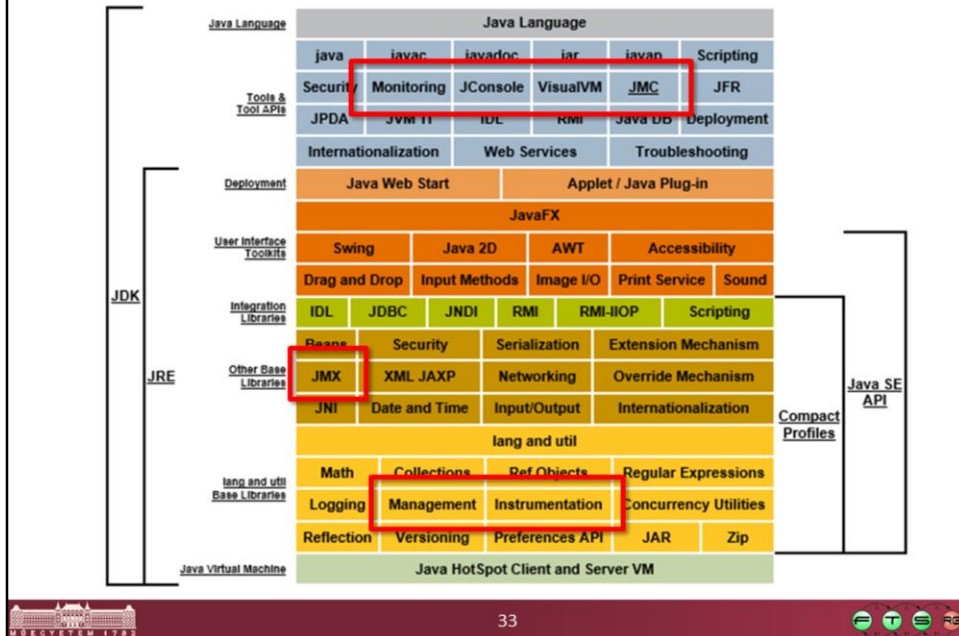
Tartalom

- Felügyeletre tervezés
- Mintapélda: felügyeleti modell elkészítése
- Felhasználható technológiák
 - WMI szolgáltató készítése
 - MS Logging Application Block
 - **Java Management Extensions (JMX)**

Modern folyamat-virtuálisgépek felügyelete



A Java platform és menedzsmentje



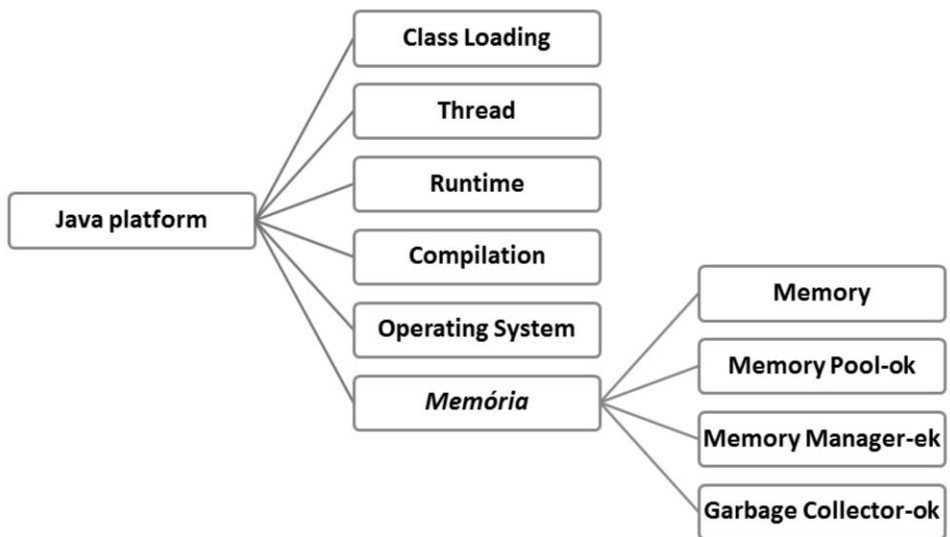
Lásd: <http://www.oracle.com/technetwork/java/javase/tech/index.html>

JRE felügyeleti támogatás

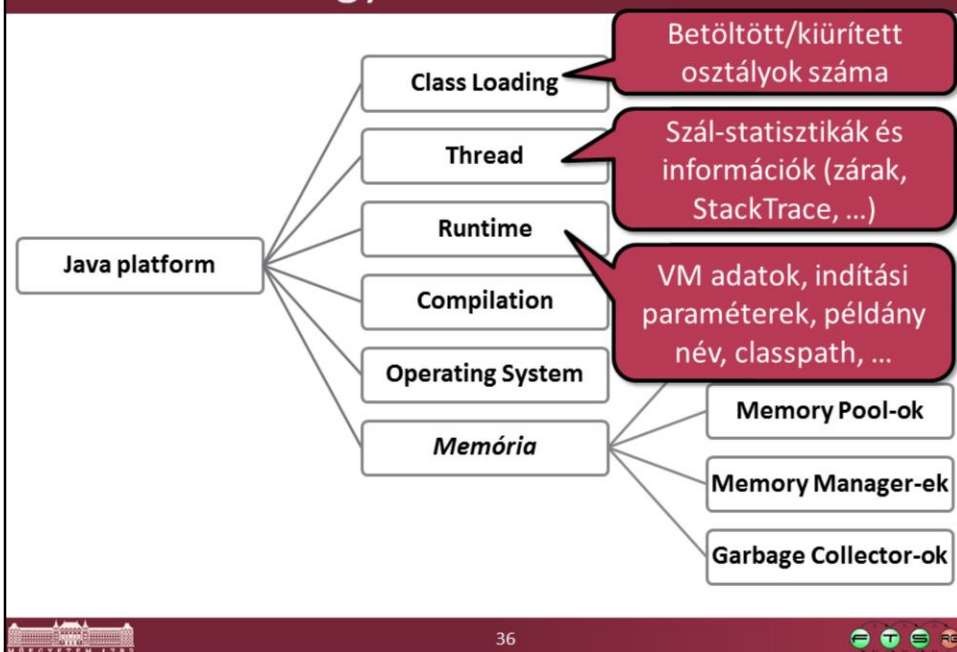
Java API: platformszintű információk lokális elérésére
`java.lang.management`

JMX	JNI	X	JAXP
Serialization	Extension Mechanism	X	JAXP
JAR	Logging	Management	
Regular expressions	Versioning	Zip	Instrumentation

Platform felügyeleti adatok: MXBean-ek



Platform felügyeleti adatok: MXBean-ek



DEMO Platform felügyeleti adatok

- Java Mission Control (JDK része!)
- Csatlakozás JVM-hez
- MBean Server megnyitása
- Platform adatok lekérdezése



37



Java Mission Control leírás:

<http://docs.oracle.com/javase/8/docs/technotes/guides/jmc/>

Saját alkalmazások felügyelete

- Jó lenne hasonlóan: objektum írja le az aspektust
- DE: Valami registry / broker is kell (pl. metaadatok)
- Távoli hozzáférés?



JRE felügyeleti támogatás

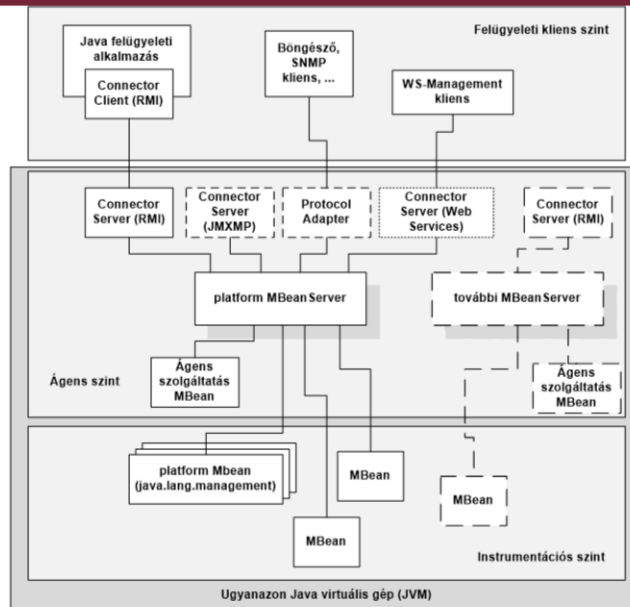
JMX	JNI	Math	
Serialization	Extension Mechanism	XML JAXP	
JAR	Logging	Management	
Regular expressions	Versioning	Zip	Instrumentation

- Instrumentációs és (távoli) menedzsment Java API
- Deklarációs mechanizmusok
- Szükséges platform-támogatás
`javax.management(.*)`

JMX (Java Management Extensions)

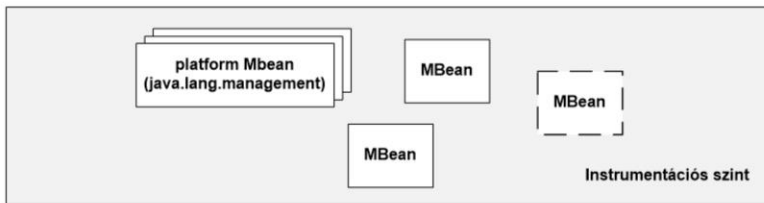
„Kibocsájtó”:	Java Community Process (JCP)
Megalkotók:	Sun, IBM, Apache, BEA, ...
Verzió:	JMX 2.0, JSR255 (2008)
Cél:	A Java platform és alkalmazások menedzmentjének szabványosítása, szerver és kliens oldali API-k és elvárt szolgáltatások megadásával
Impl.:	J2SE 5.0-tól és J2EE 1.4-től kötelező

JMX - architektúra



MBean-ek

- **Managed Bean:** alapvetően egy Java objektum
 - Bean osztály: elnevezési, létrehozási és viselkedési konvenciók
 - Szinte akármit reprezentálhat
 - Platform MBean-ek (MXBean): JVM erőforrások
 - EE környezetek szolgáltatásai (JDBC, tranzakciókezelés, ...)
 - Saját modell



MBean-ek nevei

- Az MBean-ekre objektum-referenciánk nincs!
- **Objektumnév (ObjectName):**
 - Forma: **domain:key=value,key=value...**
 - Domain név: egyszerű, nem hierarchikus névtérkezelés
 - + kulcs tulajdonságok rendezetlen halmaza
 - Az „eredeti” típus (Java osztály) nem jelenik meg!
 - Best practice: „type” és „name” nevű kulcsok
- MXBean-ek: szabványos nevek

MBean név példák

```
Catalina:type=Cache,host=localhost,path=/tomcat-docs
```

```
Catalina:type=Cache,host=localhost,path=/servlets-  
examples
```

```
Catalina:type=ThreadPool,name=jk-8009
```

```
java.lang:type=Runtime
```

```
java.util.logging:type=Logging
```

```
com.sun.management:type=HotSpotDiagnostic
```



44



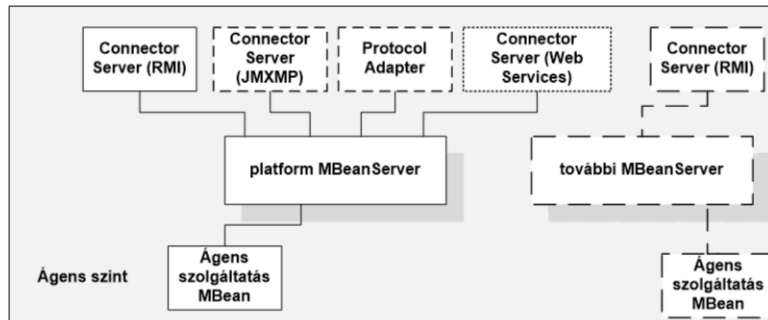
- Az első példában a Catalina az objektumnév első része, utána a kettőspont után következnek a kulcsok felsorolása. Ebben az esetben a path az, ami megkülönbözteti az egyes példányokat.
- A java.util.logging példa mutatja, hogy bár az objektumnév csak egy sztring, és nincs benne hierarchia, megfelelő konvenciót alkalmazva azért bele lehet csempészni a hierarchiát.

MBeanServer

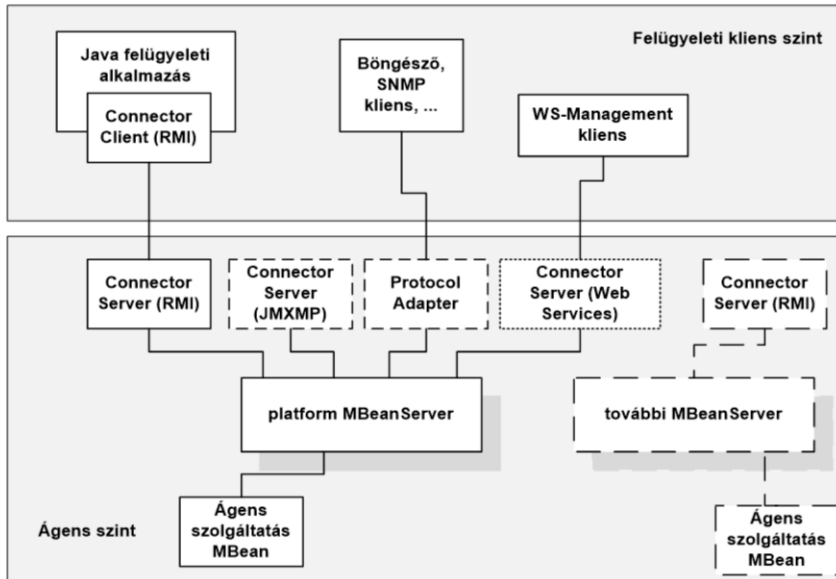
- MBean-eket nyilvántartó „broker” objektum
- Távoli és lokális interfészei különböznek!
- Műveletek:
 - Létrehozás/törlés
 - Keresés (név és név-minta szerinti)
 - Lekérdezés (attribútum- és metódushalmaz)
 - Attribútumok olvasása és írása
 - Metódusok hívása
 - Jelzésekre feliratkozás

Konnektorok

- MBeanServer(-ek) lokális elérése: Factory minta
`static MBeanServer ManagementFactory.getPlatformMBeanServer()`
- Másik JVM-ből?



Konnektorok



MBean-ek fejlesztése

- Egy MBean egy konkrét Java osztály, ami
 - Implementálja a saját MBean interfészét vagy
 - a `DynamicMBean` interfészt, illetve
 - Opcionálisan a `NotificationBroadcaster` interfészt
- Az első opció: „standard” MBean
 - Menedzsment interfész: egyszerű szabályok a struktúrán

Standard MBean példa – interfész

Az osztálynév: MyClass

```
public interface MyClassMBean {  
  
    public int getState();  
    public void setState(int s);  
    public void reset()  
  
}
```

További művelet

Van megfelelő getter/setter: a State attribútum látszani fog



49



Az Mbean interfész nevének MBean-re kell végződnie, és az eleje a menedzselt osztály neve.

Standard MBean példa – megvalósítás

```
public class MyClass implements MyClassMBean{  
    private int state = 0;  
    private String hidden = null;  
  
    public void warble() {}  
  
    public String getHidden(){return hidden;}  
    public void setHidden(String h){hidden = h;}  
  
    public void setState(int s) {state = s;}  
    public int getState()  
  
    public void reset()
```

Rejtett marad

+ a publikus konstruktorok látszanak (az MBeanServer is példányosíthat)

Standard MBean példa – regisztráció

```
public static void main(String[] args){  
  
    MBeanServer mbs =  
        ManagementFactory.getPlatformMBeanServer();  
  
    MyClass m = new MyClass();  
  
    try {  
        mbs.registerMBean(m,  
            new ObjectName("inf.mit.bme.hu:" +  
                "type=MyClass,name=probe"));  
    }  
    [...]  
}
```

Platform MBeanServer: távoli hozzáférés

```
java -Dcom.sun.management.jmxremote.port=9004 \  
-Dcom.sun.management.jmxremote.ssl=false \  
-Dcom.sun.management.jmxremote.authenticate=false \  
ManagedApp
```

FIGYELEM: NEM BIZTONSÁGOS!
(Lokális hozzáférés máshogy)

Standard MBean példa

The screenshot displays the MBean Browser interface. On the left, the MBean Tree shows a hierarchy of packages, with 'control' selected under 'hu.bme.mit.ftsrg.beeper'. The main area shows the MBean Information for the selected MBean, with the following details:

MBean Information Item	Value
MBean Name	hu.bme.mit.ftsrg.beeper.type=control
Properties in creation order	type=control
MBean domain	hu.bme.mit.ftsrg.beeper
type	control
MBean Java Class	hu.bme.mit.ftsrg.jmxbeep.BeeperControl
MBean Description	Information on the management interface of the MBean

JDK-támogatás

Tools &
Tool APIs

java	javac	javadoc	jar	javap	Scripting
Security	Monitoring	JConsole	VisualVM	<u>JMC</u>	JFR
JPDA	JVM TI	IDL	RMI	Java DB	Deployment
Internationalization	Web Services		Troubleshooting		

DEMO Saját JMX alkalmazás megfigyelése

- Java Mission Control
- Csatlakozás az alkalmazást futtató JVM-hez
- Saját MBean:
 - metaadatok
 - tulajdonságok lekérdezése
 - beavatkozás

Linkek

- JMX technology page (Oracle):
 - <http://docs.oracle.com/javase/7/docs/technotes/guides/jmx/index.html>
- JMX tutorial:
 - <http://docs.oracle.com/javase/tutorial/jmx/index.html>
- JMX adatbiztonság-menedzsment (hitelesítés / titkosítás):
 - <http://java.sun.com/javase/6/docs/technotes/guides/management/agent.html>