Hello Java Enterprise Edition

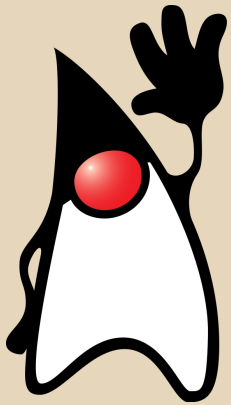
JavaSE vs JavaEE, JavaEE vs Spring

Óbudai Egyetem, Java Enterprise Edition
Műszaki Informatika szak
Labor 1

Bedők Dávid
2018-02-11
v1.0

Információ

Bedők Dávid (qwaevisz)
@UNI-OBUDA 2006
E-mail: bedok.david@nik.uni-obuda.hu
Hivatalos tárgyi weboldal:
http://users.nik.uni-obuda.hu/bedok.david/jee.html
Forráskódok: https://github.com/davidbedok/oejee



Előkövetelmény:

- **Java SE alapos** ismeret
- **ANSI SQL** és **adatbáziskezelés alapfokú** ismerete
- **Alapvető XML** és **XHTML** ismeret
- **Java Servlet** és **JSP alapfokú** ismeret
  - http://users.nik.uni-obuda.hu/bedok.david/jse.html
- Open-source szemlélet
- Java ecosystemre való nyitottság
- **GIT** verziókezelő használata (?)

- **Szerveroldali Java programozás** (NSTAJ1SVNB)
  - A tárgy neve és kódja egy korábbi kurzusból lett örökölve
  - A *Szoftvertechológia Intézet* (ST) **szabadon választható** kurzusa
  - **4** kredit, heti 3 labor (∼42 kontaktóra)
- **J2EE fejlesztés** (NIXJA1SBNE)
  - Az *Alkalmazott Informatikai Intézet* (AI) **Szoftvertervezés és -fejlesztés** szakirányának **kötelező** kurzusa
  - **4** kredit, heti 1 előadás + 2 labor (∼42 kontaktóra)
    - heti 3 labor + 2 konzultációs labor
    - számonkérés a konzultációs labor elején (kötelező)

Különbségek

A két tárgy követelménye között alapvető különbség nincsen, az értékelés minőségében azonban igen. Az egyik egy szabadon választható tantárgy, melynek elsődleges célja az érdeklődők számára egy technológia bemutatása, míg a másik egy már szakosodott hallgatói csoport felkészítése a technológia alkalmazására.

Követelmények (1..5)

- **Önálló projekt munka** elkészítése a tanult technológiák felhasználásával
  - 3. hét: Terv leadása PDF formátumban (max. 2 oldal)
  - 3. hét-13. hét: Projekt folyamatos fejlesztése (git history)
  - 13. hét: Projekt fejlesztésének befejezése
  - 13-14. hét: Projekt bemutatása (10-15 perces élő demo, max. 5 perc prezentáció)
  - 14. hét: Projekt fejlesztési dokumentáció leadása (15-20 oldal PDF formátumban)
  - Pótleadás: Ha sokminden hiányzik, akkor esélytelen, egyéb esetben egyedi elbírálás alapján.
    - Dokumentáció pótleadás vizsgaidőszakban (gyakorlati jegy póton)
- **Elméleti teszt zárthelyik**
  - átlag muszáj >= 2
  - Zárthelyik javítása lehetséges a 14. héten (átlagot lehet javítani)
  - Zárthelyik teljes pótlása lehetséges gyakorlati jegy póton

Követelmény (1..3)

- **Elméleti teszt zárthelyik**
  - átlag >= 3 $\rightarrow$ 2
  - átlag >= 4 $\rightarrow$ 3
  - Zárthelyik javítása lehetséges a 14. héten (átlagot lehet javítani)
  - Zárthelyik teljes pótlása lehetséges gyakorlati jegy póton

Tervek szerint legalább 10 elméleti zárthelyi lesz. TVSZ gyenge értelmezése szerint legalább 7 (inkább 8) megírása kötelező.

Az elméleti zárthelyik **felelet-választós tesztek** lesznek.

Féléves feladat

Téma követelmények

Egy valóságban is létező problématérre épülő alkalmazás készítése, melyben definiálható a téma körül értelmezett adat- és a rajtuk végzett művelethalmaz.
Az alkalmazásnak több "interface"-e kell hogy legyen:

1. Webes interface (3-5 JSP/JS dinamikus weboldal)
2. REST API (CRUD műveletek, akciók végrehajtása)
3. Queue/Topic interface pl. batch adatfeltöltésre, aszinkron művletvégzésre
4. Management felület (JMX)
5. SOAP WebService
6. Remote EJB (RMI)

- ▷ <3 db interface: sikertelen projekt
- ▷ 3 db interface: maximum 3-es érdemjegy
- ▷ 4 db interface: maximum 4-es érdemjegy
- ▷ >4 db interface: akár 5-ös érdemjegy

Az interface-ekhez ha szükséges, kliens programokat is kell készíteni (pl. Java vagy C# nyelven), vagy konfigurációt kell adni (pl. SOAP UI).

Féléves feladat

Adatbázis követelmények

A választott téma kapcsán egy olyan ANSI SQL db schema kialakítása, mely az alábbi komplexitásokat tartalmazza:

- Legalább 6 adatbázis tábla, kb. 25 mező
- Min. 1 db 1-N kapcsolat egy tábla és egy törzstábla között (a törzstáblából enum lesz entitás helyett)
- Min. 1 db 1-N kapcsolat két tábla között
- Min. 1 db N-M kapcsolat kapcsolótáblával
- Egyéni komplexitás (pl. partnerlekérdezéses tábla, topológia, stb.)
- Legalább 2 db unique index és legalább 2 normal index
- Léterhozó és takarító postgresql scriptek (minta alapján)
- Saját schema, user, role (minta alapján)

Az adatbázis schemára számos kész és működő minta script halmaz található a tárgy git repository-jában, így a fenti követelmény kizárólag a kreativitást írja elő, a "szakmai munka" része gyakorlatilag készen van (interpretálni kell csak).

Féléves feladat

Project GIT repository

A féléves feladatokkal folyamatos fejlesztés mellett az alábbi *git repository*-ban kell dolgozni:
https://github.com/davidbedok/oejee2018spring

- Ha még nincs **GitHub** account, létre kell egyet hozni
- A *GitHub account* és a diák egyértelmű összekötése a `project.json` állományban lesz (repository gyökerében). Ennek karbantartása a hallgató feladata!
- A *GitHub user* nevet e-mailben kell elküldeni (subject: "[OE][JEE][neptun] Lorem Ipsum git: loremipsum").
- Válasszon mindenki egy üzleti igényt/témát, illetve projekt nevet!
- `project.json` állományt kell kitölteni

Project állomány

```
{
  "period": "2016-2017/1",
  "projects": [
    {
      "name": "sample",
      "description": "Sample project",
      "platform": "weblogic",
      "members": [
        {
          "name": "David Bedok",
          "neptun": "Q59R7A",
          "github": "davidbedok"
        }
      ],
      "interfaces": [
        {
          "tpye": "restful",
          "goal": "handle crud operati
        },
        {
          "type": "jms",
          "goal": "bulk upload data"
        }
      ]
    }
  ]
}
```

A git repository-ban törekedjünk mindenhol az **angol nyelv** használatára.

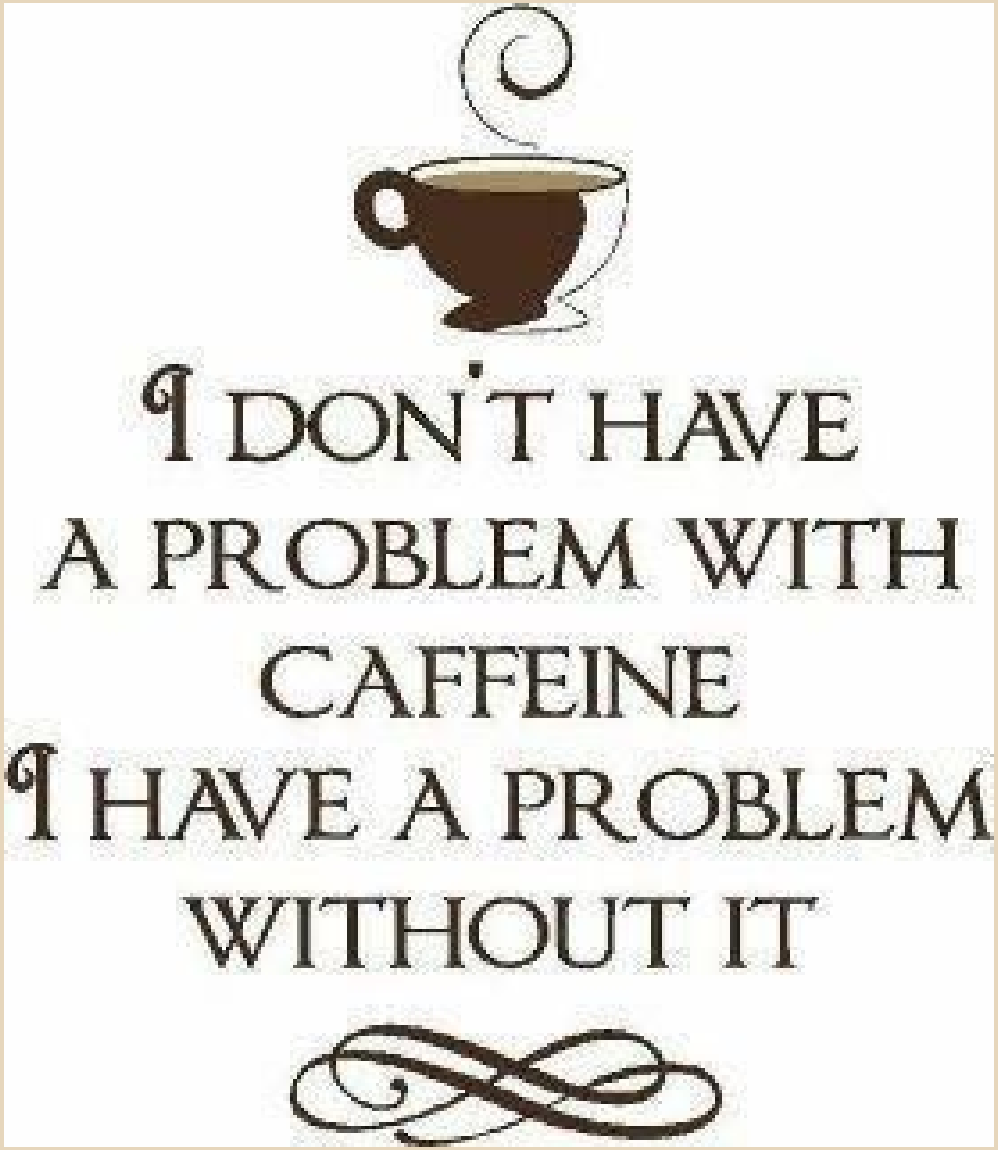
A project neve (`name` element) kisbetűs, angol, white space mentes egyértelmű és egyedi kifejezés legyen a repo-ra nézve. A project nevének megfelelő könyvtár létezzen a `/projects` könyvtárban alkönyvtárként. Ez legyen a saját projekt build root könyvtára.

A `members` tömb elemei hozzák létre a kapcsolatot a hallgató nevével, neptun kódjával és github felhasználója nevével.

`platform`: [weblogic|jms]
interface `type`: [web|rest|jms|jmx|soap|rmi]

1. kód `project.json`

Java szó eredete



Változatok

- **Java Card**
  - smartcard-ok számára
- **Java Platform Micro Edition** (Java ME, korábban* J2ME)
  - csökkentett erőforrások, mobil eszközök számára
- **Java Platform Standard Edition** (Java SE, korábban* J2SE)
  - workstation-ök számára
  - általános felhasználás, kliens gépek
  - **JavaFX** (rich desktop alkalmazások számára)
    - earlier it was a separate variation
- **Java Platform Enterprise Edition** (Java EE, korábban* J2EE)
  - elosztott vállalati környezetben, avagy széles(ebb) spektrum igény esetén

*: 2006 előtt más néven voltak hivatkozva, de ez zavaró volt

1991 SUN (Stanford University Network): **Oak**→ **Green**

- **Dr. James A. Gosling**
- Mike Sheridan, Patrick Naughton

1996.01.23 Java 1.0 [AWT]
1997.02.19 Java 1.1 [Inner class, JDBC, RMI, Reflection API]
1998.12.08 Java 1.2 **Playground**
2000.05.08 Java 1.3 **Kestrel** [Java Sound, JNDI API]
2002.02.06 Java 1.4 **Merlin** [regexp, exception chain, Image IO, Pref. API]
2004.09.30 Java 5 **Tiger** [autoboxing, generic types]
2006.12.11 Java 6 **Mustang**
2007 GPL, open-source and free software license
2009 Oracle acquisition
2011.07.28 Java 7 **Dolphin**
2014.03.18 Java 8 **Spider** [lambda expression] - Current **8u144**
2017.09.21 Java 9 (money, currency API, better nativ code integration, ..)
2018 Java 10 (removal of primitive types)

1998	Java Professional Edition
1999	J2EE 1.2 (Java 2 Platform, Java SE 1.2+)
2001	J2EE 1.3 (Java SE 1.3+)
2003	J2EE 1.4 (Java SE 1.4+)
2006	JEE 5 / JavaEE 5 (Java SE 5+)
2009	JEE 6 / JavaEE 6 (Java SE 6+)
2013	JEE 7 / JavaEE 7 (Java SE 7+)
2017Q4	JEE 8 / JavaEE 8 (Java SE 8+)

JRE verziószáma 1.4 után: 5, 6, 7, 8, ..
JDK verziószáma még sok helyen: 1.5.0, 1.6.0, 1.7.0, ..
JRE 8.x és JDK 1.8.x már official, JEE 8 még készülőben

JavaEE - Elemei

Java Community Process's JSR (Java Specification Request)

JDBC	Java Database Connectivity [JSR54, JSR114, JSR221]
RMI-IIOP	Java Remote Method Invocation over Internet Inter-Orb Protocol
JNDI	Java Naming and Directory Interface Specification
Java Servlet	[JSR154, JSR315, JSR340]
JSP	JavaServer Pages [JSR152, JSR245]
JSTL	JavaServer Pages Standard Tag Library [JSR52]
EJB	Enterprise JavaBeans [JSR153, JSR220, JSR318, JSR345]
JMS	Java Message Service [JSR914, JSR343]
JTA	Java Transaction API [JSR907]
JCA	J2EE Connector Architecture [JSR112, JSR322]
JAAS	Java Authentication and Authorization Service
JSF	JavaServer Faces [JSR127, JSR252, JSR314, JSR344]
JMX	Java Management Extensions [JSR3, JSR160, JSR255, JSR262]
JAX-WS	Java API for XML-Based Web Services [JSR224]
JAX-RS	Java API for RESTful Web Services [JSR311, JSR339]
JAXP	Java API for XML Processing [JSR206]
JAXB	Java Architecture for XML Binding [JSR222]
JPA	Java Persistence API [JSR220, JSR317, JSR338]
SAAJ	SOAP with Attachments API for Java [JSR67]
EL	Expression Language [JSR245, JSR341]
CDI	Contexts and Dependency Injection [JSR299, JSR346]
Interceptors	[JSR318]
...	

JEE verziók és elemeinek verziói

Content	J2EE 1.2	J2EE 1.3	J2EE 1.4	JEE 5	JEE 6	JEE 7
JDBC	2.0		3.0		4.0	4.1
JNDI	1.2					
RMI-IIOP	1.1					
Java Servlet	2.2	2.3	2.4	2.5	3.0	3.1
JSP	1.1	1.2	2.0	2.1	2.2	2.3
EJB	1.1	2.0	2.1	3.0	3.1	3.2
JMS	1.0		1.1			2.0
JTA	1.0			1.1		1.2
JAXP	-	1.1	1.2	1.3		
JSTL	-	1.0	1.1	1.2		
JCA	-	1.0	1.5		1.6	1.7
JAAS	-	1.0	1.3			
JSF	-	-	1.1	1.2	2.0	2.2
JMX	-	-	1.2			2.0
JAX-WS	-	-	-	2.0	2.2	
JAXB	-	-	-	2.0	2.2	
SAAJ	-	-	-	1.3		
JPA	-	-	-	1.0	2.0	2.1
JAX-RS	-	-	-	-	1.1	2.0
EL	-	-	-	-	2.2	3.0
CDI	-	-	-	-	1.0	1.1
Interceptors	-	-	-	-	1.1	1.2

Spring Framework

Java EE versenytársa

- **Kliens és Szerver oldali** alkalmazás keretrendszer
- Nyílt forráskód
- **Inversion of Control** (IoC) konténer a Java Platform számára
  - dependency injection
  - reflection használata direkt példányosítás helyett

Tartalma:

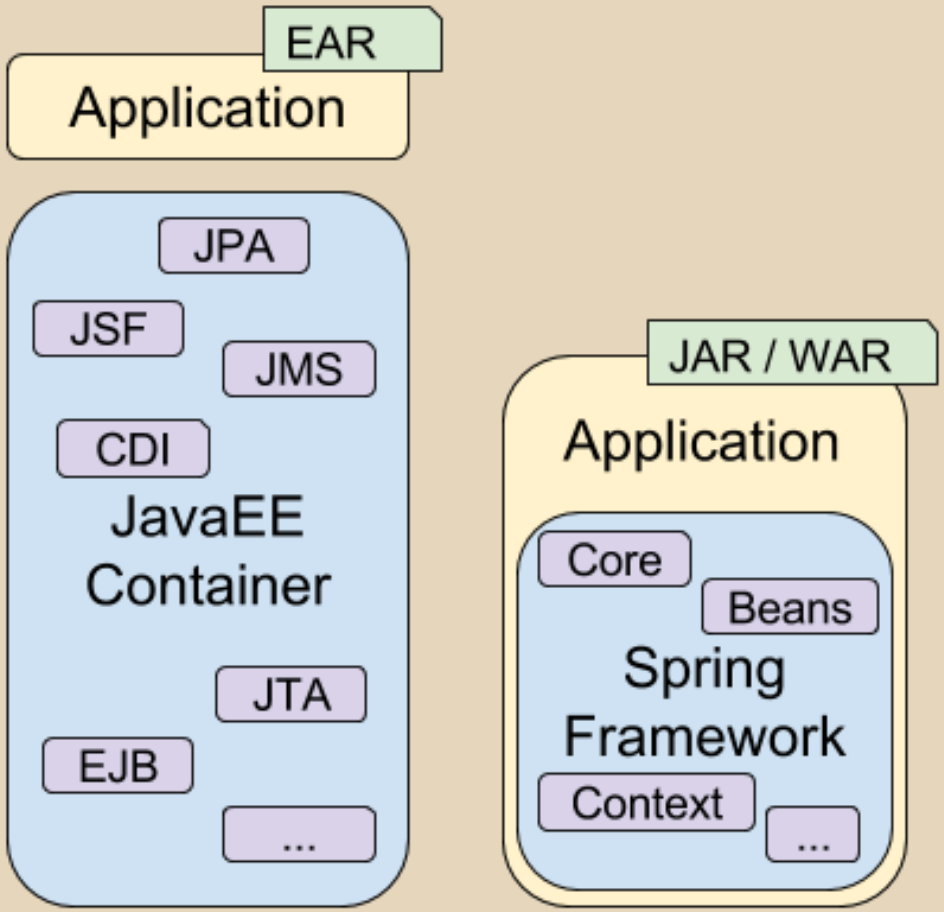
- **Aspect-Oriented Programming** (AOP) framework
- Data Access framework
- Transaction management framework
- **Model–View–Controller** framework
- Remote access framework
- ...

2002 October	**Rod Johnson**: Expert One-on-One J2EE Design and Development
2004 March	Spring Framework 1.0
2006 October	Spring Framework 2.0
2007 November	Spring Framework 2.5
2009 December	Spring Framework 3.0
2013 December	Spring Framework 4.0
2015 July	Spring Framework 4.2
2016 June	Spring Framework 4.3

Spring Framework vs Java Enterprise Edition

Számos teljes implementáció létezik. A modulok **szabványok** által vezéreltek (JSRs).

Kicsi alkalmazások, de JavaEE compliant **heavy-weight konténer** szükséges.



Léteznek library-k melyek összekötik a Spring-et a szabvány könyvtárakkal (pl. `spring-jms` vagy `spring-data-jpa`).

Nagy alkalmazások a Spring könyvtárat becsomagolása végett, cserébe egy **light-weight web-konténer** elegendő a futtatáshoz.

Software architektúrák

- Monolitikus (monolithic)
- Többrétegű / N-rétegű (multitier/n-tier)
- Szolgáltatás orientált (service oriented)
  - Üzenet vezérelt (message oriented)
  - Microservice
- "Szervernélküli" (Serverless)

JavaEE architektúrái

Nincsen minden célra megfelelő architektúra a software fejlesztésben. Mindegyiknek megvan/meglehet a maga előnye és hátránya. Az egyik jobban illeszkedik a JavaEE világába, a másik kevésbé.

Monolitikus architektúra

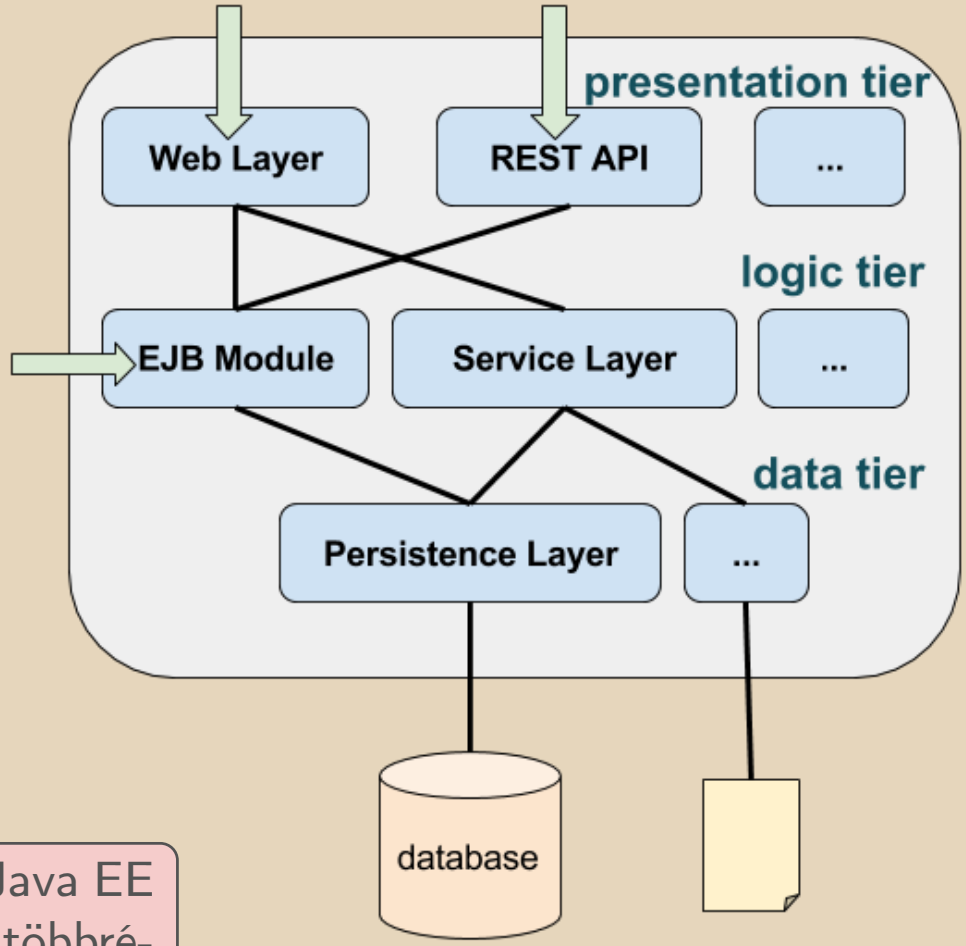
modularitás nélkül tervezve

Egy **monolitikus alkalmazás** önálló, független más alkalmazásoktól. A tervezés koncepciója hogy az alkalmazás nem csak egy feladatért, hanem egy meghatározott üzleti folyamat minden eleméért felel.

A JavaEE alkalmazások monolitikusak?

Egy komplex JavaEE alapú EAR artifactot tekinthetünk monolitikus alkalmazásnak, de a Java EE sokkal több mint egy monolitikus alkalmazásokat összeépítő keretrendszer.

Többrétegű architektúra



A prezentációs, feldolgozó és adat kezelő funkciók fizikailag szét vannak választva.

A legtöbb esetben a Java EE alapú alkalmazások többrétegűek.

Szolgáltatás orientált architektúra (SOA)

"Do one thing and do it well"[1]

Szereljünk szét egy JavaEE monolitikus többrétegű alkalmazást kisebb darabokra.

Előnyök:

- moduláris fejlesztés magasabb szinten kezelhető
- az alkalmazást könnyebb lesz értelmezni, fejleszteni és tesztelni

Variációk a darabok közötti kommunikáció függvényében:

- **Üzenet alapú** (pl. JMS, stb.)
- **RESTful** (pl. Microservices, stb.)

Microservice-ek

A Microservice alapú fejlesztés azonban több mint egy példa a SOA-ra:

- lazán kapcsolt szolgáltatások halmaza
- kis szolgáltatások - adott funkció kifinomult kiszolgálására (FaaS)
- minden szolgáltatás rugalmas, alakítható, minimális és teljes

[1] Unix filozófia

Szervernélküli architektúra

Function as a Service (FaaS)

A "szervernélküli" (serverless) architektúra egy felhő alapú végrehajtási modellre utal, ahol a felhő kiszolgáló dinamikusan kezeli a felhőben lévő erőforrások allokációját/kiosztását.

Szerver nélkül?

Természetesen szerverekre itt is szükség van. A név onnan ered, hogy a szerverek és azok kapacitásának kezelése, az ezekkel kapcsolatos tervezési döntések teljes egészében rejtettek a fejlesztő és az operátor előtt.

Java fejlesztői környezet

- Java
  - **Oracle Java JDK**, Open JDK, ...
- Source Control
  - **Git**, Mercury, SVN, ...
- Integrated Development Environment (IDE) + plugins
  - Eclipse, IntelliJ IDEA, Netbeans
- (Enterprise) Application Server (EAS/AS)
  - Apache Tomcat, RedHat **JBoss**, Oracle Glassfish, Oracle **WebLogic**, ...
- Test tools and libraries
  - Selenium, jUnit, **TestNG**, SoapUI, ...
- Persistence layer / Storage
  - **PostgreSQL**, MySQL, Redis, Derby (JavaDB), ...
- Messaging Framework
  - Active MQ, HornetQ, ...
- Report frameworks
  - Jasper Reports, ...
- Continuous Integration (CI) support
  - PMD, Codestyle, static checks, ...

Java SE JDK

Install

Letöltés:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Verzió: **8u144**

Környezeti változók:

- **JAVA_HOME** →`j:\java\jdk1.8.0_102`
- **Path** módosítása → `%Path%;%JAVA_HOME%\bin`

```
1 >java -version
2 java version "1.8.0_102"
3 Java(TM) SE Runtime Environment (build 1.8.0_102-b14)
4 Java HotSpot(TM) 64-Bit Server VM (build 25.102-b14, mixed mode)
```

Git

Source Code Management (SCM)

- Nyílt forráskódú elosztott verziókezelő
- **Linus Torvalds**
- Verzió: **2.14.1**
- Letöltés: https://git-scm.com/
- Install (windows installer)
  - Use Git from the Windows Command Prompt
  - Use OpenSSH
  - Checkout Windows-style, commit Unix-style line endings
  - Use MinTTY
  - Disable file system caching

```
1 >git --version
2 git version 2.8.1.windows.1
```

GitHub

Open source development platform

- Public repository-k számára ingyenes
- Elsősorban open-source társaságok számára (üzleti érdekszférába tartozó projektek esetén a `public repository` később kellemetlenséget okozhat)
- https://github.com/
- Regisztráljunk!
- https://github.com/davidbedok

BitBucket:

- https://bitbucket.org/
- `Private repository`-k számára is ingyenes 5 fejlesztőig

OEJEE Repository

https://github.com/davidbedok/oejee.git

```
1 >git clone https://github.com/davidbedok/oejee.git
2 Cloning into 'oejee'...
3 remote: Counting objects: 4, done.
4 remote: Compressing objects: 100% (3/3), done.
5 remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
6 Unpacking objects: 100% (4/4), done.
7 Checking connectivity... done.
```