Hello Java Enterprise Edition

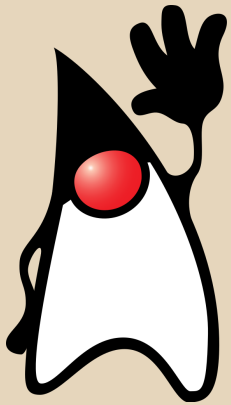
JavaSE vs JavaEE, JavaEE vs Spring

Óbuda University, Java Enterprise Edition
John von Neumann Faculty of Informatics
Lab 1

Dávid Bedők
2018-02-11
v1.0

Information

Dávid Bedők (qwaevisz)
@UNI-OBUDA 2006
E-mail: bedok.david@nik.uni-obuda.hu
Official webpage:
http://users.nik.uni-obuda.hu/bedok.david/jee.html
Source codes: https://github.com/davidbedok/oejee



Prerequisite:

- **Deep** knowledge of **Java SE**
- **Basic** knowledge of **ANSI SQL database management**
- **Basic** knowledge of **XML**, **XHTML**
- **Basic** usage of **Java Servlet** and **JSP** technology
  - http://users.nik.uni-obuda.hu/bedok.david/jse.html
- Open-source approach
- Openness to Java ecosystem
- **GIT** usage (?)

- **Szerveroldali Java programozás** (NSTAJ1SVNB)
  - The name and the code of the course is inherited.
  - It is an **optional** course of the *Szoftvertechológia Intézet* (ST)
  - **4** credits, 3 labs/week (∼42 contact hours)
- **J2EE fejlesztés** (NIXJA1SBNE)
  - It is an **obligatory** course of the **Szoftvertervezés és -fejlesztés** (*Alkalmazott Informatikai Intézet*, AI) specialization
  - **4** credits, 1 lecture and 2 labs / week (∼42 contact hours)
    - 3 lectures/week + 2 consultation labs/week
    - exams at the beginning of the labs (obligatory)

Differences

The requirements are the same, but the quality of the evaluation differs. One of the course is an optional item, here the main goal is presenting the technolgy for those who interested. On the other hand the main goal of the obligatory course is preparing the students the usage of the related technology.

Requirements (1..5)

- **Project work** with the learned technologies
  - 3rd week: Plan in PDF format (max. 2 pages)
  - 3rd-13th week: Continuous development (git history)
  - 13th week: Finish the implementation
  - 13th-14th week: Project presentation (live demo in 10-15 minutes, max. 5 min presentation)
  - 14th week: Submit project development documentation (15-20 pages in PDF format)
  - Replacement: If you have not got lots of missing things, replacement will be available based on an individual judgment.
    - Documentation replacement (in the exam period)
- **Theoretical test midterms**
  - average must >= 2
  - correction of the tests could be on the 14th week (average correction)
  - complete replacement in the exam period

Requirements (1..3)

- **Theoretical test midterms**
  - average >= 3 $\rightarrow$ 2
  - average >= 4 $\rightarrow$ 3
  - correction of the tests could be on the 14^{th} week (average correction)
  - complete replacement in the exam period

According to the plans it will be at least 10 exams. 7 or 8 exams are mandatory based on the learning rules

The theoretical exams will be **multiple-choices** tests.

Semi-annual assignment

Subject/Topic requirements

Application for a real world problem, where a database schema and some opearations over that schema can be interpreted.
There must be several interfaces of the application:

1. Webes interface (3-5 JSP/JS dynamic webpages)
2. REST API (CRUD operations, other actions)
3. Queue/Topic interface e.g. batch import, async operations
4. Management interface (JMX)
5. SOAP WebService
6. Remote EJB (RMI)

- ▷ <3 interfaces: unsuccessful project
- ▷ 3 interfaces: maximum 3 grade
- ▷ 4 interfaces: maximum 4 grade
- ▷ >4 interfaces: even 5 grade

You have to create client applications (e.g. in Java or C#) for the interfaces if needed. In same cases adequate configurations are sufficient (e.g. SOAP UI).

Semi-annual assignment

Database requirements

According to the choosen topic an ANSI SQL db schema creation is required with the following rules:

- At least 6 database table, about 25 columns
- Min. 1 piece of 1-N relation between a table and a trunk table (trunk table: it can be represented as an enum in ORM)
- Min. 1 piece of 1-N relation bwtween two tables
- Min. 1 piece of N-M relation with relation/meta table
- Custom complexity (e.g. partner query table, topology, etc.)
- At least 2 piece of unique indecies and at least 2 normal indecies
- Creation and cleaning postgresql scripts (based on sample files)
- Own schema, user, role (based on sample files)

You can find lots of sample database files ans scripts in the git repository of the course. The above mentioned requirements only need some creativity and not technical skills.

Semi-annual assignment

Project GIT repository

During the continuous development you have to push your work at least weekly into the following git repository:
https://github.com/davidbedok/oejee2018spring

- If you have not created a **GitHub** account yet, you have to create one.
- To bind the *GitHub account* and the student you have to edit the `project.json` file on the root of the repository.
- You have to send your *GitHub user* name via e-mail (subject: "[OE][JEE][neptun] Lorem Ipsum git: loremipsum").
- Choose a unique project name.
- Fill the `project.json` file.

Project file

```
{
  "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"
        }
      ]
    }
  ]
}
```

Use the **english language** wherever possible.

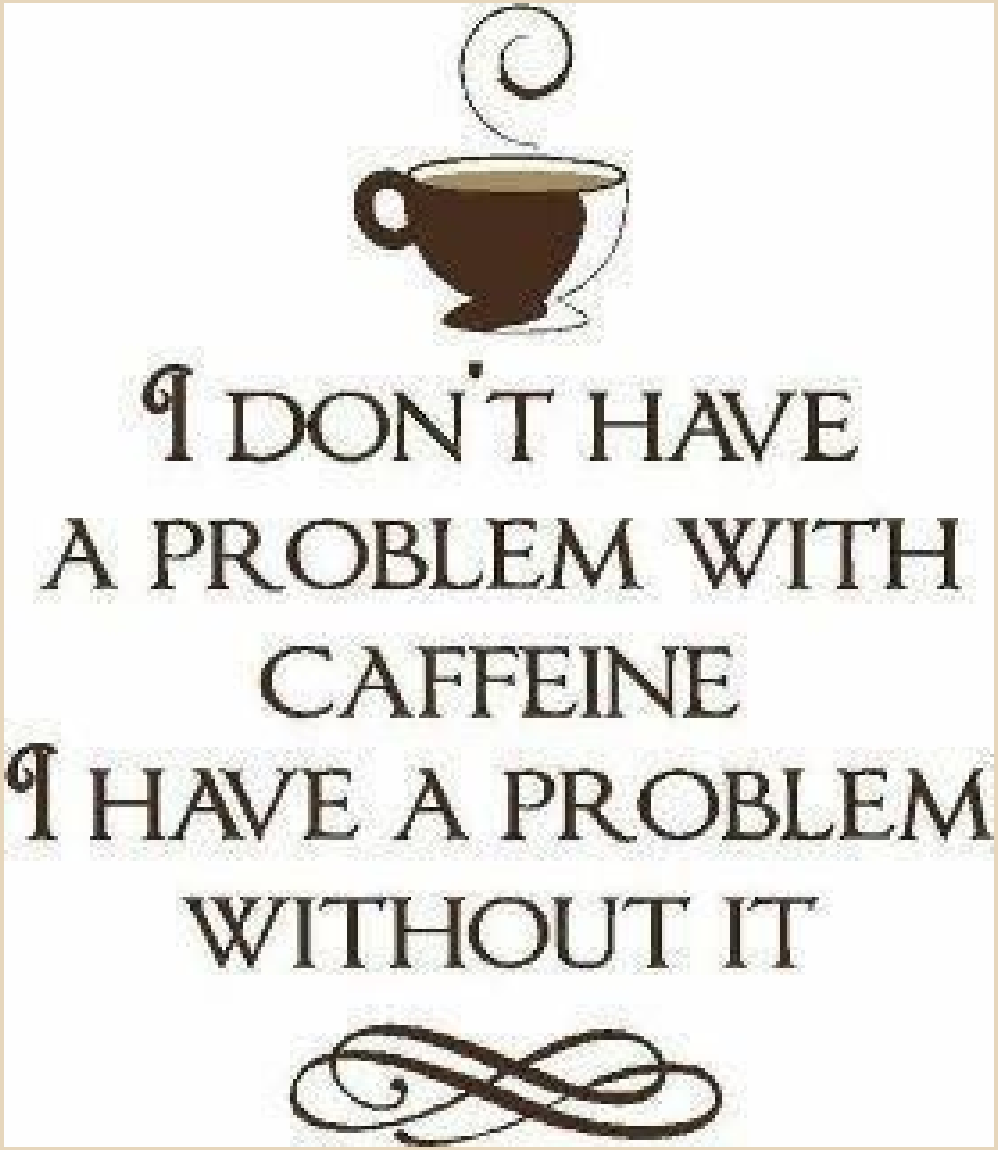
Let the `name` of the project be lower case, english word without any white space (unique in the level of the repository). Create a directory with the same name under the `/projects` directory. This will be the root folder of the build system.

The `members` array will create the connection between the student and the github user.

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

1. kód project.json

The origin of the Java word



Variations

- **Java Card**
  - for smartcards
- **Java Platform Micro Edition** (Java ME, earlier* J2ME)
  - limited resources, for mobile devices
- **Java Platform Standard Edition** (Java SE, earlier* J2SE)
  - for workstations
  - general purpose, for client machines
  - **JavaFX** (for rich desktop and Internet applications)
    - earlier it was a separate variation
- **Java Platform Enterprise Edition** (Java EE, earlier* J2EE)
  - distributed enterprise environment or wider spectrum demand

*: before 2006 the names were different, and it was very incommodious

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 version number after 1.4: 5, 6, 7, 8, ..
JDK version numbers left the original: 1.5.0, 1.6.0, 1.7.0, ..
JRE 8.x and JDK 1.8.x are GU, but JEE 8 is under construction

JavaEE - Content

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]
...	

Content by versions

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

Competitor of Java EE

- **Client and Server side** application framework
- Open source
- **Inversion of Control** (IoC) container of the Java Platform
  - dependency injection
  - reflection instead of direct instantiation

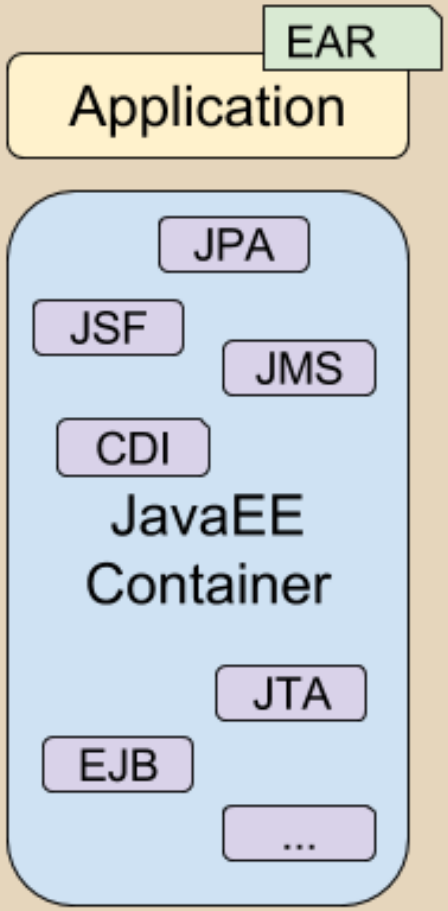
Content:

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

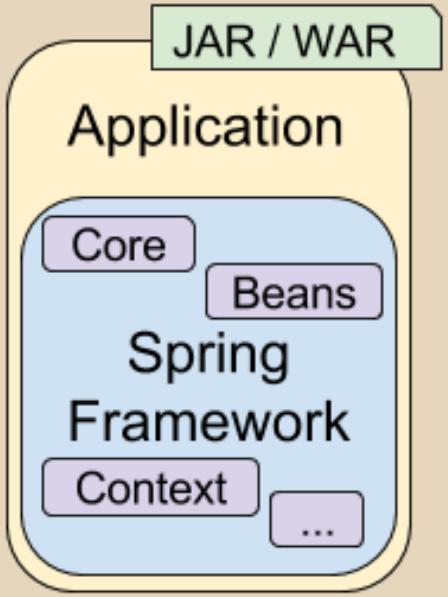
2002 October	**Rod Johnson**: Expert One-on-One J2EE Design and Development
2004 March	Spring Framework 1.0
2006 October	Spring Framework 2.0
2007 November	Spring Framework 2.5
2009 December	Spring Framework 3.0
2013 December	Spring Framework 4.0
2015 July	Spring Framework 4.2
2016 June	Spring Framework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

Spring Framework vs Java Enterprise Edition

There are several complete implementations. Modules are driven by **standards** (JSRs).



Applications are small, but you need a JavaEE compliant **heavy-weight container**.



There are some bridge libraries which joint the Spring libraries with the standards' implementations (e.g.: `spring-jms` or `spring-data-jpa`).

Applications are big, 'cos those contain the used Spring libraries, but you only need e.g. a **light-weight web-container**.

Software architectures

- Monolithic
- Multitier / n-tier
- Service-oriented
  - Message driven
  - Microservice
- Serverless

Achitecture for JavaEE

There is no 'silver bullet' architecture for software development. Each one has its own advantages and disadvantages. One of them is better for a JavaEE application, other than not.

Monolithic architecture

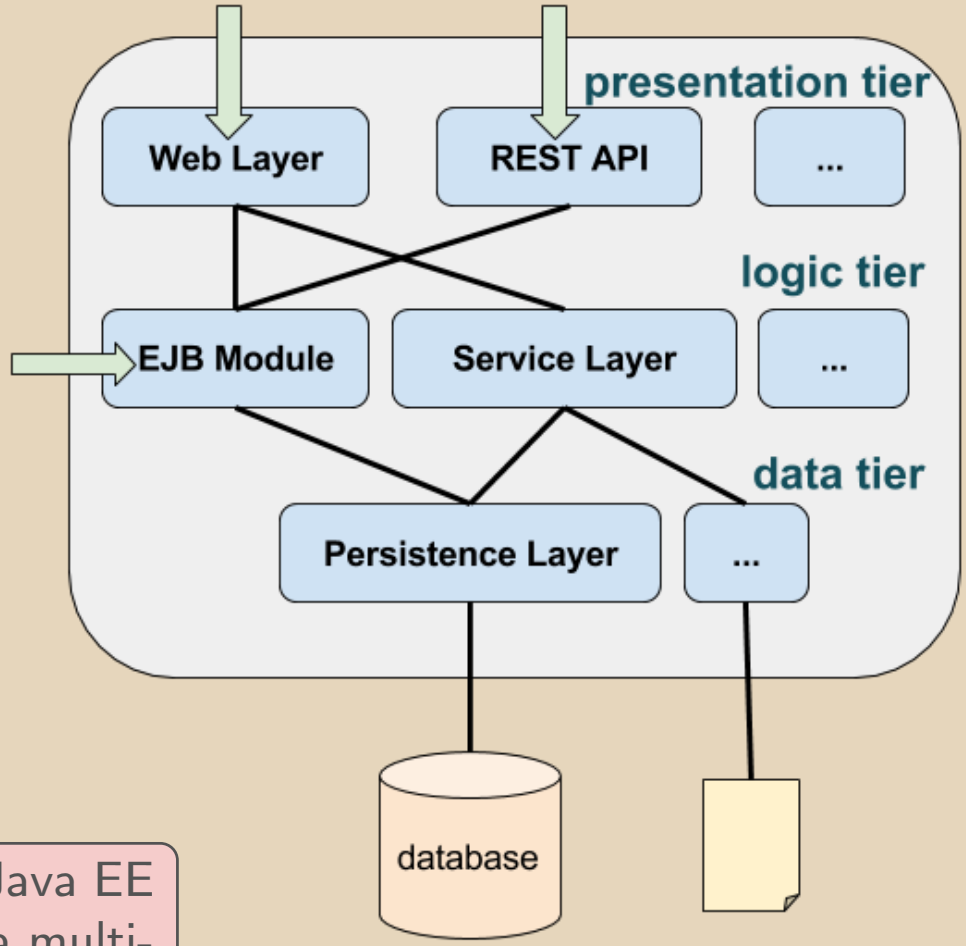
designed without modularity

A **monolithic application** is self-contained, and independent from other computing applications. The design philosophy is that the application is responsible not just for a particular task, but can perform every step needed to complete a particular function.

Monolithic JavaEE application?

A complex Java EE based EAR artifact could be considered as a monolithic application, but Java EE is much more than a monolitic application builder.

Multitier / n-tier architecture



Presentation, application processing, and data management functions are physically separated.

Most of the cases a Java EE based application is a multi-tier application.

Service oriented architecture (SOA)

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

Disassemble a JavaEE monolithic multitier application to smaller pieces.

Advantages:

- improved modularity
- makes the application easier to understand, develop and test

Different approaches based on the communication between the pieces:

- **Message based** (e.g. JMS, etc.)
- **RESTful** (e.g. Microservices, etc.)

Microservices

Microservices are much more than just a kind of example of SOA.

- collection of loosely coupled services
- small services - fine-grained to perform a single function (FaaS)
- each service is elastic, resilient, composable, minimal, and complete.

[1] Unix philosophy

Serverless computing is a **cloud computing execution model** in which the cloud provider dynamically manages the allocation of machine resources.

Without server?

Serverless computing still requires servers. The name "serverless computing" is used because the server management and capacity planning decisions are completely hidden from the developer or operator.

Java design environment

- 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

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

Version: **8u144**

System Environment Variables:

- **JAVA_HOME** →`j:\java\jdk1.8.0_102`
- **Path** modification → `%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)

- Open source distributed source control manager
- **Linus Torvalds**
- Version: **2.14.1**
- Download: 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

- Free for Public repositories
- Primarily for open-source communities
- https://github.com/
- Sign uo!
- https://github.com/davidbedok

BitBucket:

- https://bitbucket.org/
- Free for `Private repositories` as well up to 5 developers

OEJEE Repository

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

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