<embed/it>

PA165: Introduction to Java EE

Petr Adámek
Tomáš Pitner

Lecture 1
Tue Sep 17, 2013

Content

- Course profile
  - Learning style
  - Assessment
  - Outline
- Java EE applications
- Java EE application architectures
- Technology around Java EE
- Basic concepts

ORGANIZATION OF THE COURSE

Course composition

- Lectures
  - Recommended (slides in English, given in Czech)
- Lab Sessions
  - Compulsory
  - Examples to the matter from lectures
  - Consulting the projects
- Team Project
  - 4-member teams
  - Checkpoints
  - Work throughout the semester

Assessment

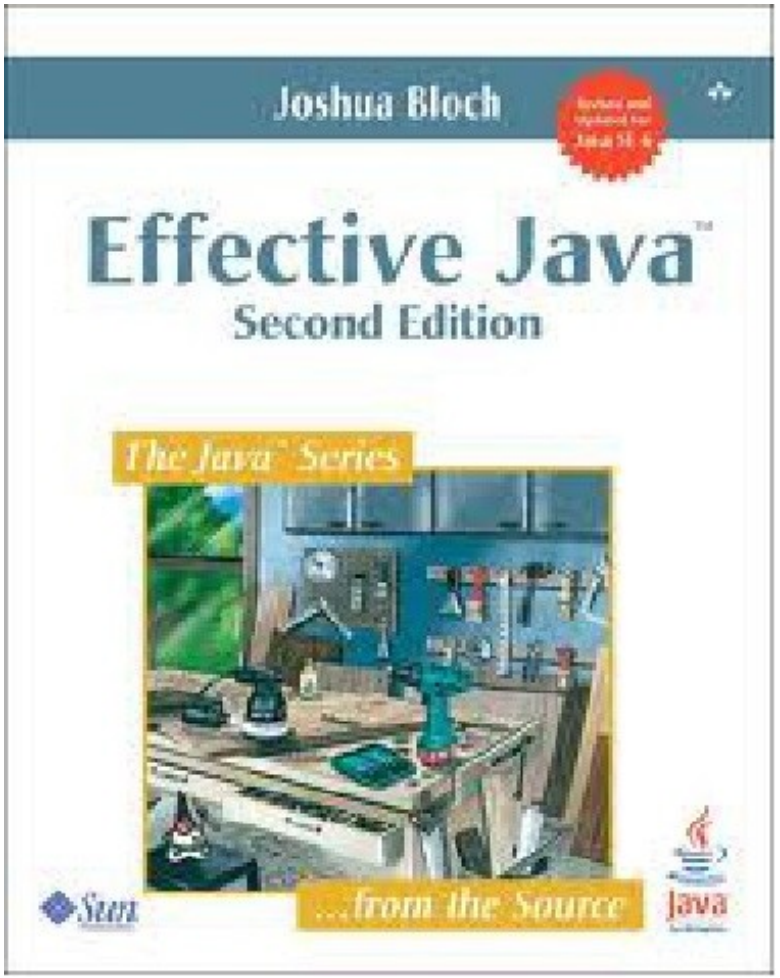
- Project: 70 points
  - Checkpoints: 4x10 points
  - Defense: 30 points
- Final exam (written on paper): 30 points

- Completion:
  - Credit: min. 60 points
  - Exam: min. 70 points

Course outline

- Intro to Java EE (architecture, technology, concepts)
- Data Persistence (ORM, JPA, Spring JDBC, iBatis, Testing)
- Application logic (IoC, AOP, Transactions, Security, Testing)
- Presentation layer (web frameworks, Stripes, Spring MVC, Wicket, JSF, Safety)
- Integration technologies (Web services SOAP, REST, JMS, RMI, IIOP, ESB)
- Testing (unit, integration, functional, acceptance, user-friendliness, efficiency, safety)

Recommended reading



- **Effective Java (2nd Edition)**
- Joshua Bloch
- http://amazon.com/dp/0321356683/
- For more info see the course outline in IS

JAVA EE PLATFORM

What is Java EE Platform?

- Platform for modern IS development
  - Provides the infrastructure
  - Industry standard (JCP)
  - Current version: Java EE 7 (since June 2013)

- Support for
  - Web applications
  - Web services
  - Multitier-applications

Modern Information Systems

- Complex and large systems
- Require integration with other systems
- Adaptability to different customer requirements
- Deployment on different platforms
- Support for a large number of clients (especially for Web applications)
- Security
- Quality and reliability

IS Developer Needs

- Rapid development
- Easy Maintenance
- Easy extensibility and customization
- Easy Integration with other systems
- Support for Agile
- Support for the team and multi-team development
- Portability
- Various software and hardware platforms, different tools and application servers
- Scalability
- Security
- Easy to test

FUNDAMENTAL CONCEPTS

Fundamental concepts

- Infrastructure
- Modularity
- Independence and low invasiveness
- Declarative access
- Convention over Configuration
- Adherence to the guidelines for the development of maintainable code

Infrastructure

- The developer should focus on your problem domain and should not be forced to deal with general issues that must be addressed in any application.
- Application architecture, security, transaction management, data persistence, communications and integration, remote access, infrastructure presentation layer, localization, etc.
- Java EE platform and the built application framework (frameworks) therefore provide the necessary infrastructure.

- Never implement your own framework!

Modularity

- The application is developed as a set of cooperating components
- Components should
  - Be loosely connected (loosely coupled), which between them should be as little dependent
  - Being reusable (whether only in the project, or even beyond)
  - Having a well designed and a separate interface (among other things, reduce the level of dependence, especially those in transition)
  - Being well-tested
- If we have a set of well-designed components, it is easy to modify and adapt application behavior
  - Replacement of components
  - By changing the configuration of components
  - By changing the connections between components

Independent and less invasive

- Components should be independent not only among themselves but also to specific technologies and application frameworks
  - At least at the level API
- This simplifies maintenance and increases reusability
- The concept of POJO (Plain Old Java Object) component
  - A common class that does not implement any specific interfaces or extend any particular class
  - It is therefore independent of any part or class library
  - Simple, clear understanding of the business does not require any special knowledge
  - You can easily create an instance, you can easily test

Declarative Approach

- Certain aspects of program behavior are not defined by traditional imperative code (*sequence of commands*), but the specifications of the *intent* (what to do).
- This leads to simplification and streamlining code.
- Recommended for transaction management, security management and access rights, automated conversion, various automatic mapping, etc.
- Self declaration desired behavior can be placed
  - In the deployment descriptor (deployment descriptor)
  - Directly in code via annotations (modern and preferred approach)

Imperative transaction control

```
public void someMethod() {

    UserTransaction transaction = context.getUserTransaction();

    try {
        transaction.begin();
        [redacted]
        transaction.commit();
    } catch (Exception ex){
        try {
            transaction.rollback();
        } catch (SystemException syex){
            throw new EJBException
                ("Rollback failed: " + syex.getMessage());
        }
        throw new EJBException
            ("Transaction failed: " + ex.getMessage());
    }
}
```

Declarative transaction control

```
@TransactionAttribute(TransactionAttributeType.RequiresNew)
public void someMethod() {
    [redacted]
}
```

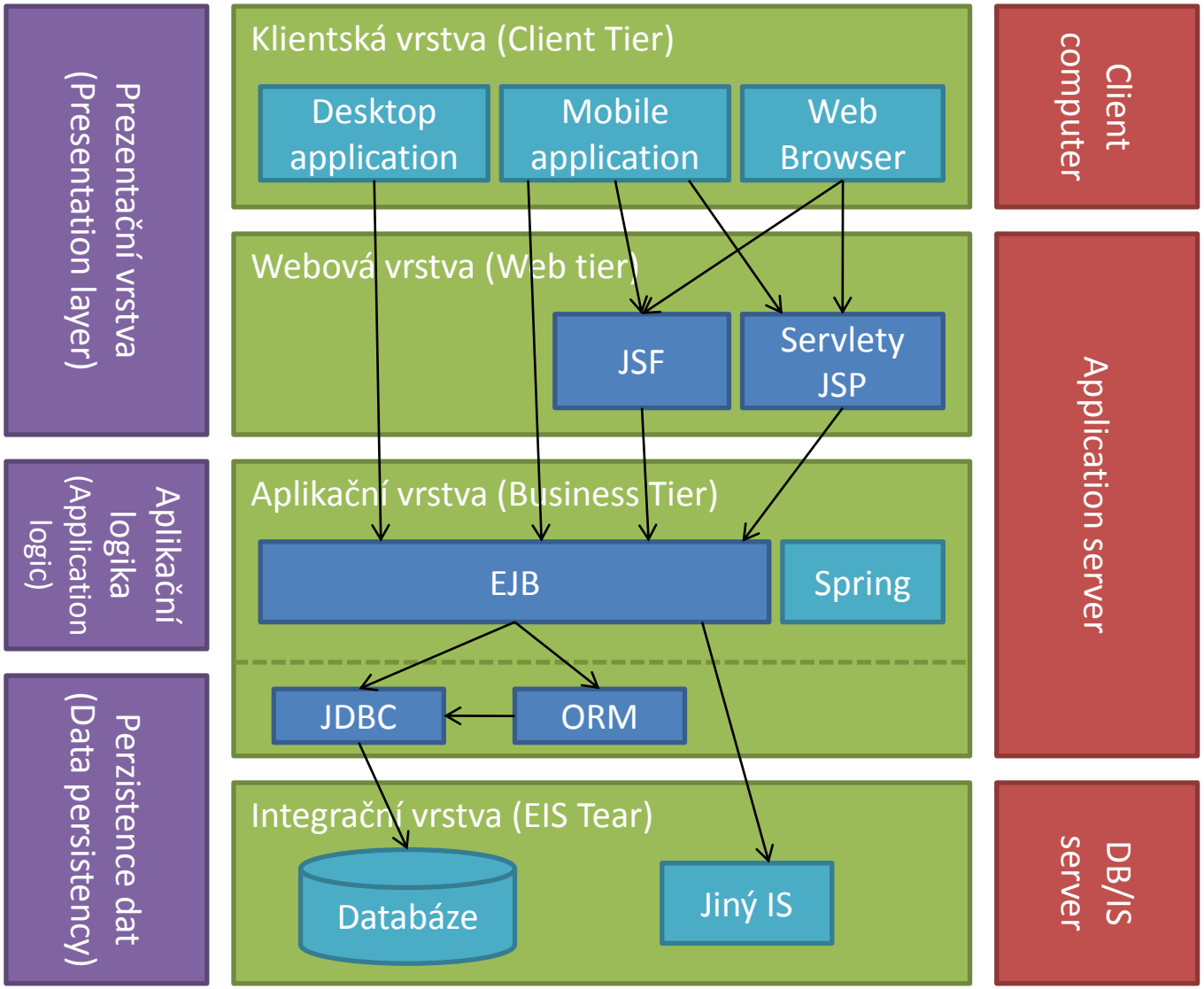
Convention over Configuration

- Concept … Ruby on Rails

In previous versions

- The first version of the Java EE platform focused mainly on infrastructure and technology
  - Ease of development was underestimated
  - Complex technology is a complex application
  - Steep learning curve
  - The need to use complex tools
- This led to the frustration developers and the emergence of alternative approaches and technologies (Hibernate, Spring)
- The change came with Java EE 5
  - Strong inspiration tool Spring, Hibernate, etc.
  - Annotations
  - POJO components

ARCHITECTURE & TECHNOLOGY


Prezentační vrstva (Presentation layer)
Aplikační logika (Application logic)
Perzistence dat (Data persistency)
Klientská vrstva (Client Tier)
Desktop application
Mobile application
Web Browser
Webová vrstva (Web tier)
JSF
Servlety JSP
Aplikační vrstva (Business Tier)
EJB
Spring
JDBC
ORM
Integrační vrstva (EIS Tear)
Databáze
Jiný IS
Client computer
Application server
DB/IS server

Presentation Layer

Desktop applications
- Swing
- AWT
- SWT
- Java Web Start

Mobile applications
- Java ME
- Android/iOS/BlackBerry OS/Windows Phone

Web applications
- Servlets, JSP, JSTL
- MVC frameworks
  - Request based (Struts, Stripes, Spring MVC)
  - Component based (JSF, Tapestery, Wicket)
- Portlets
- Applets

Application Logic

Plain class library

- Not suitable for larger applications

EJB

- Requires an application server supporting *EJB* or *EJB lite*

Spring framework

- 3rd party (community) products, not part of Java EE
- Very popular
- Non-invasive

Data Persistence

JDBC

- Universal API for DB access
- Cumbersome (too low-level) when used directly, so we use:
  - Template Method
  - Spring JDBC
  - Commons DB
  - RowSet

ORM

- Standard JPA (currently JPA 2.0)
- Hibernate, TopLink, Eclipse Link

Obsolete

- EJB 2.x
- JDO

Application servers

Open Source - full

- JBoss
- Glassfish

Open Source - only servlet container

- Tomcat
- Jetty

Commercial

- WebSphere (IBM)
- WebLogic (Oracle, formerly BEA)

Questions

?