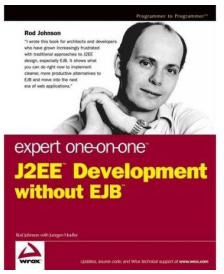
PA 165 – Enterprise Java

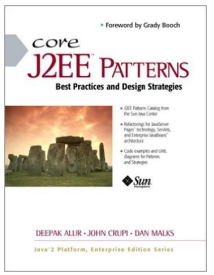
Organizational Details

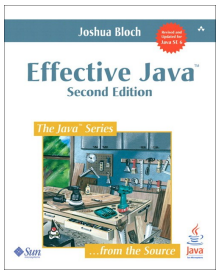
22nd Sept 2015

Course Aim

- to understand selected chapters from **advanced Java-based system design** and **implementation**;
- To be aware of **methodological issues** of **high-quality system design** and **implementation** and related topics;
- to be able to work with the **most important APIs** from Java SE/EE, Spring framework, Java EE and Javascript frameworks for UI;
- To **get used to team work** within large enterprise software development and with system design by applying enterprise patterns;







Summarized Content

- Intro to **large (enterprise) Java-based application** and **systems**
- **Development tools** (Netbeans, Maven, Git)
- **Enterprise patterns** (DTO, DAO)
- **Persistence/ORM** (JPA/Hibernate)
- **Internet applications** (servlets, JSP, taglibs, Java web containers)
- **Web application layers**, security (authentication, authorization, main attacks), Spring MVC, client-side javascript frameworks (AngularJS), HTML, CSS, DOM
- **Spring framework** (AOP, dependency injection, security, transactions, Spring Boot)
- **Web services** (REST, WS-* standards), Spring-WS, JAX-RS
- **Messaging Systems** (JMS)

Seminars & Project

- Working on an **assignment** based on what seen during lectures
- Will be useful for the **team project**, developed in teams of four
- **Evaluation:**
  - 60 points for the project (3 milestones + final defence)
  - 40 points for the final exam;

- More details by your instructor during the first seminar

The Team

PA 165 – Enterprise Java

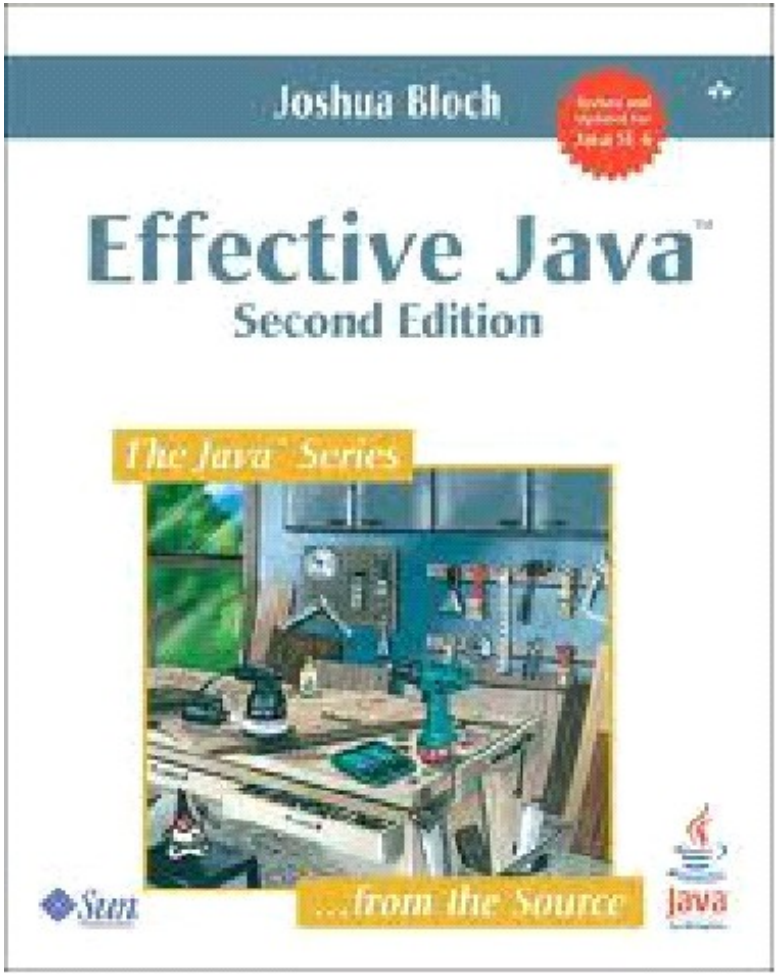
Introduction to Java EE
Tools

22nd Sept 2015

Content

- Java EE Introduction
  - Java EE Technology
  - Basic concepts
  - Architecture
- Tools
  - Maven
  - Git

Resources



- Effective Java (2nd Edition)
- Joshua Bloch
- http://amazon.com/dp/0321356683/

Java EE Platform

What is Java EE

- Platform for modern enterprise system development
- Industry standard (JCP)
- Current version is Java EE 7 (JSR 342)

Characteristics of modern IS

- Complex and large systems
- Integrated with other systems inside and outside the organization
- Adaptibilty to different requirements of various customers
- Multiple platforms support
- Support for big amount of users (especially in case of web applications)
- Security
- Quality and reliability

Development process requirements

- Fast development
- Easy maintainance
- Easy extensibility and adaptibiy
- Easy integration with other systems
- Support for agile development
- Support for team and multi-team development
- Portability and compatibility (various HW, OS, DB, tools, application servers, etc.)
- Scalability
- Security
- Easy to test

Basic Concepts

- Infrastructure
- Modularity
- Independance and low invasivness
- Declarative approach
- Convention over configuration
- Following basic rules for mainainable code

Infrastructure

- Developer should be focused on problem domain he should not be spending time and effort with general problems which are not specific for given application
  - Architecture, authentication, authorization, transaction managament, data persistence, communication and integration, remote access, presentation layer architecture, localization, etc.
- Java EE platform and frameworks are providing such infrastructure
- Never implement your own framework!

Modularity

- Application should be assembled from cooperating components
- Components should be
  - Loosely coupled
  - Reusable
  - With well designed and separated interface
  - Well tested
  - Well documented (contract described with JavaDoc)
- With well desigend components, we can easily change and adjust application behaviour
  - By changing the component
  - By changing the component configuration
  - By changing connection between components

Independance and low invasivness

- Coponents should be as less dependant as possible not only between each other, but also on specific technologies and frameworks (at least at interface level)
- It helps to reusability and maintainance
- POJO (plain old java object) concept

Imperative TX management

```
public void someMethod() {

   UserTransaction transaction = context.getUserTransaction();

   try {
      transaction.begin();
      doSomething();
      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());
    }
}
```

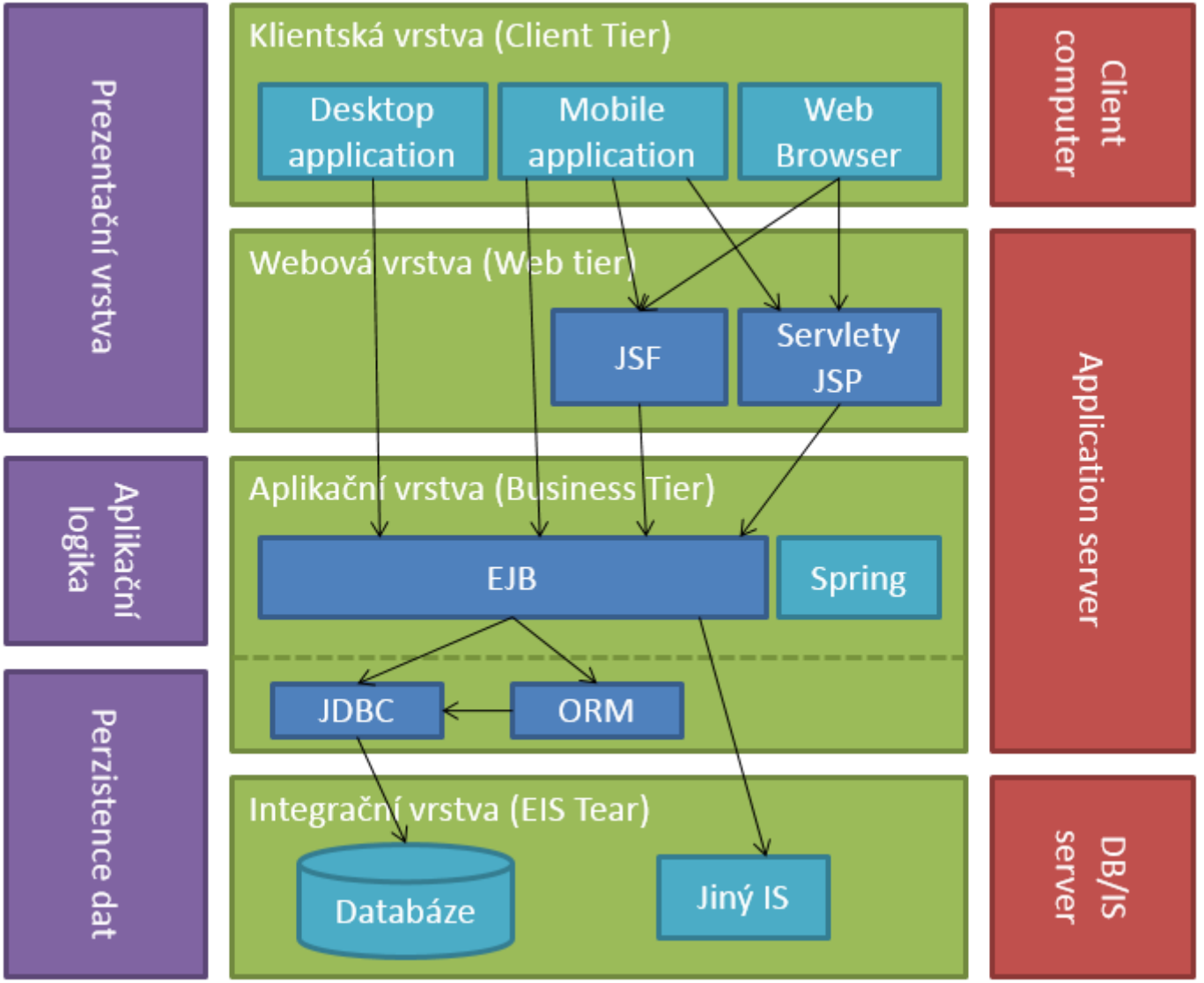
Declarative TX management

```
@TransactionAttribute(TransactionAttributeType.RequiresNew)
public void someMethod() {
     doSomething();
}
```

Convention over Configuration

- Propagated by Ruby on Rails

Java EE Architecture



Tools

Maven

- Software project management and comprehension tool.
- Embedded Tomcat
- Demo

Git

- Source management tool
- Demo