
Firemný J2EE framework založený na open source produktech

Ján Šoltís, Peter Blšták
SOFTEC, s.r.o.

The Lifecycle of Software Technology [1]

- Invention
- Expansion and Innovation
- Consolidation
- Maturity
- Free Open-Source Software (FOSS) Domination
- The FOSS Era

V ktorom štádiu je J2EE?

Why J2EE projects fail? [2]

- J2EE is distributed, so everything is remote
- Technology is more important than the problem
- Too much code
- Over-engineering
- Not separating presentation logic from business logic
- RDBMSs are evil (Java is the center of the world)
- Building everything on your own

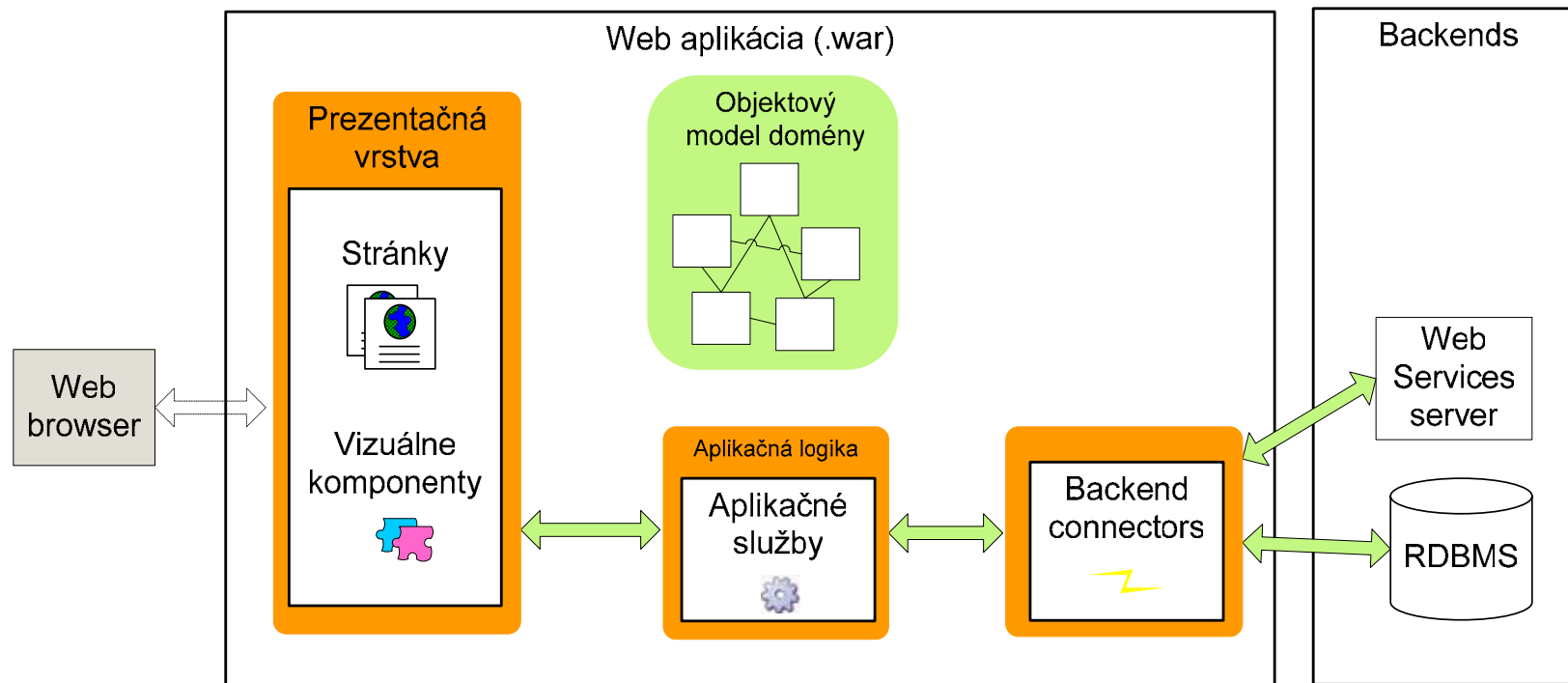
Zložitosť [3]

- **Essential complexity** – zložitosť samotného problému, ktorý riešime
- **Accidental complexity** – vlastnosť riešenia
- Accidental complexity is created when fine grained, general purpose abstractions like integers and strings are used to implement coarse grained, problem-specific concepts like Customers and Products
- Riešenia aj jednoduchých problémov môžu obsahovať enormnú „accidental complexity“

J2EE framework WAFT

- Založený na open-source produktoch
- Minimalizácia accidental complexity
- Zdravý rozum (vid' problémy J2EE projektov)

Architektúra WAFT aplikácie



Prezentačná vrstva

- Založená na Tapestry
- Obsahuje len prezentačnú logiku
- Vrstva vykonáva svoju činnosť s pomocou objektov, ktoré implementujú rozhrania z aplikačnej vrstvy
- Skutočné objekty, ktoré tieto rozhrania implementujú vrstva nepozná
- Dáta dostáva vo forme objektov *doménového modelu*

Tapestry

- Komponentový framework
- Programátor pracuje s objektmi, metódami a properties týchto objektov
 - nie s URL a query parametrami
- Skutočný objektovo-orientovaný vývoj
- Výzor v čistom HTML
- Tapestry umožňuje jednoducho vytvárať vlastné komponenty
 - „Everything is a reusable component“
- Jednoduchosť a efektivita

Doménový model

- Hierarchia tried špecifická pre danú doménu
- Triedy zvyčajne zodpovedajú business konceptom – zmluva, klient,...
- Len prenáša dáta medzi vrstvami (Data Transfer Objects), aplikačná logika nie je súčasťou modelu
- Aj keď býva často mapovaný do databázy, nemá zvyčajne znalosť o ďalších vrstvách
 - A teda môže byť testovaný nezávisle
 - Môže mať rôzne mapovania (produkcia vs. testovanie)

Metadáta - anotácie

```
public class Pripomienka
{
    @Required
    @TextConstraints(max = 100)
    private String text;

    @DatePrecision(DatePrecisions.YEAR_TO_DATE)
    private Date datumOdoslania;

    private Instalacia instalaciaZistenia;

    @Enumeration("StavPrip")
    private CTEnumeration stavRiesenia;
```

Aplikačná vrstva

- Aplikačne závislý kód
- Pracuje s doménovými objektmi
- Používa Mapper API na získanie a uloženie týchto objektov z/do repository (napr. databázy)
- Realizovaná formou služieb
- Služba
 - Skupina metód
 - Bežný Java interface a Java metódy

Výhody servisnej architektúry

- Na metódy je možné definovať prístupové práva
- Auditovanie a logovanie volaní metód
- Jednoduchá zámena implementácií
 - Paralelný vývoj
 - Testovanie
- Možnosť implementovať služby distribuovane (remoting, EJB, WS)

Backend connectors

- Mapper API: API používané na „sperzistnenie“ doménového modelu do repository (zvyčajne DB, ale môže byť aj súbor, XML, pamäť)
- Mapper API implementácie:
 - Hibernate
 - DAO používajúce priamo JDBC
 - Castor – XML súbory

Open-source infraštruktúra

- Tapestry
- Hibernate
- Spring (DI, AOP, ACEGI Security, remoting)
- Tomcat/JBoss
- MySQL/ProgreSQL
- JUnit, CanooWebTest, Luntbuild

WAFramework

- Integruje jednotlivé technológie
- Pridáva navyše:
 - Autentifikáciu
 - Autorizáciu
 - Interpretáciu metadát
 - Silné vizuálne komponenty
 - Konzistentné spracovanie chýb
 - Riadenie DB konekcií a transakcií
 - Práca s číselníkmi
 - ...

Procesný framework

- Nie všetko sa dá naprogramovať do frameworku
- Procesný framework = postupy vývoja, metodika
- Častý problém: metodiky sú príliš abstraktné a formálne (a teda nepoužiteľné)
- WAFT metodika je praktická, obsahuje okrem iného:
 - konfiguráciu nástrojov
 - prípravenie vývojového prostredia
 - odkazy na dokumentáciu
 - konfiguračné riadenie, defect tracking, builds
 - projektové štandardy
 - coding style, code review

WAFt framework

- **Produktívny** – aplikačný programátor pracuje na vyššej úrovni abstrakcie
- **Stručný** – výsledný kód je jednoduchý a čitateľný, nie je potrebné žiadne generovanie kódu
- **Silný** – v prípade potreby môže aplikačný programátor zostúpiť aj na nižšiu úroveň abstrakcie
- **Flexibilný** (nie je monolitický) – prakticky každú jeho časť je možné zameniť v prípade, že je v danom prostredí vhodnejšia iná implementácia
- **Pragmatický** – využíva postupy a technológie, ktoré fungujú, nie tie, ktoré najviac počúť

A bottom-up, developer-focused approach to developing applications

Referencie

- [1] James, A.C.: The care and feeding of FOSS
 - http://www.moonviewscientific.com/essays/software_lifecycle.htm
- [2] Johnson, R.: Why J2EE Projects Fail
 - http://www.theserverside.com/news/thread.tss?thread_id=34532
- [3] Greenfield, J., Short, K.: Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools