Frameworky

Ing. Tomáš Černý MS

Co známe

- Analýzu
- UML
- Popis architektury
- Známé architektury

Obsah

- Frameworky
  - Co to je
- Příklady
- J2EE
- EJB
- JSF
- Facelets
- ORM
- Seam
- Struts
- Spring

Framework

- Samotný název framework je často složité definovat a v různém kontextu znamená různé věci.
- **Framework** je rozšířitelná množina objektů pro funkce z dané oblasti.
- Například GUI framework je Java Swing.

Framework - hodnocení

Hodnocením kvality frameworku je do jaké míry poskytuje implementace služeb potřebných k řešení problému v dané oblasti, jaké obsahuje mechanismy dovolující vývojáři použít různé funkce k dosažení řešení celkového.

Framework

Java Swing framework poskytuje

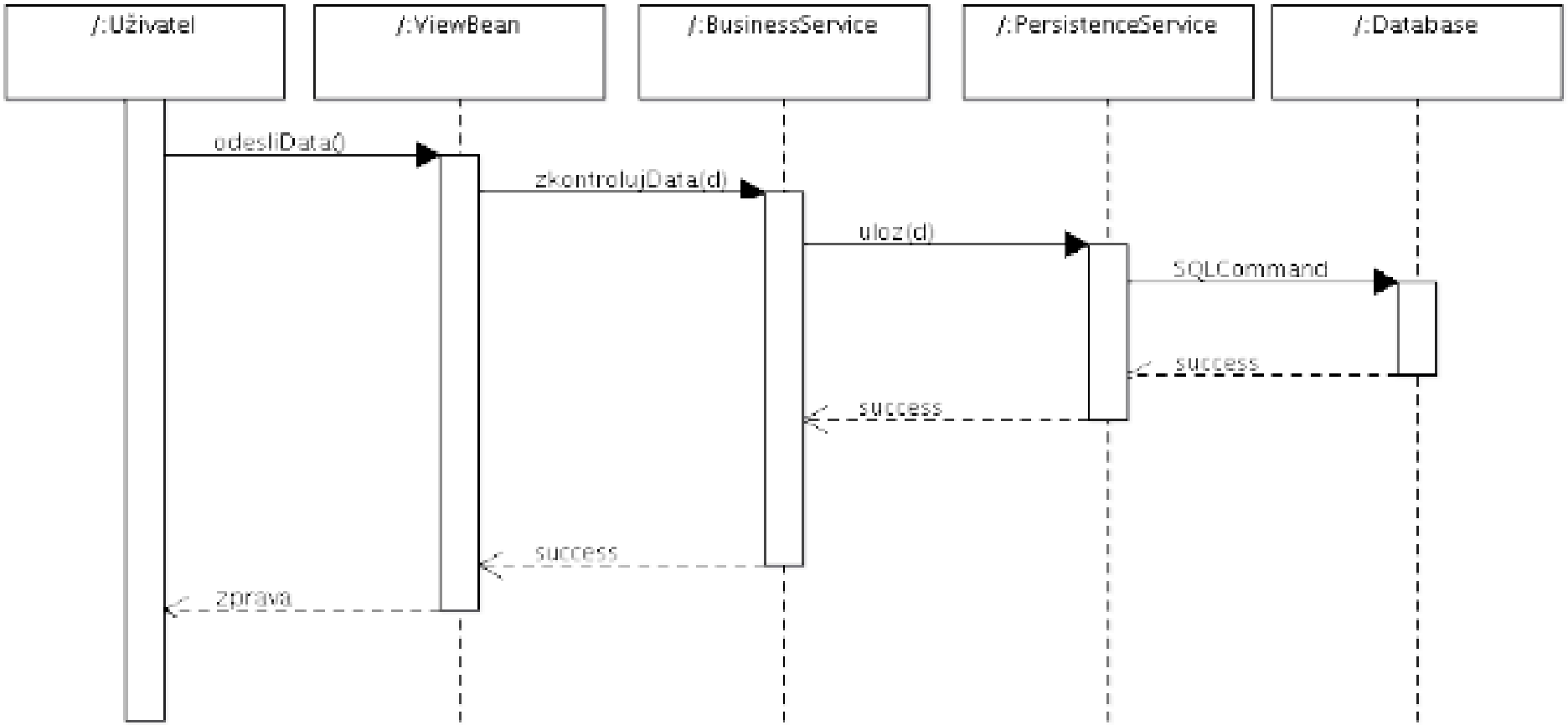
- mnoho tříd a rozhraní pro základní GUI funkce
- vývojář přidává grafické komponenty děděním ze swingovských tříd
- vývojář také využívá možností odpovědí na události (event), z předdefinovaných tříd, registrováním odposlechu (listener)
  - pomocí návrhové vzoru založeného na Observeru.

Framework - co a jak?

- soudružná množina rozhraní a tříd, které spolupracují k poskytnutí služeb logickému podsystému
- obsahuje konkrétní a abstraktní třídy definující rozhraní přizpůsobující k interakci objektů
- běžně vyžaduje, aby vývojář dědil třídy frameworku a upravil či rozšířil jejich služby
- obsahuje abstraktní třídy mající jak konkrétní, tak abstraktní metody
- Hollywoodský princip.. „**Nevolej nám, my zavoláme tobě**“. Tedy uživatelem definované třídy budou volány z předdefinovaných tříd frameworku

Framework - příklad

Pokusme se framework definovat na konkrétnějším případě



Framework - příklad

Nejsme první, kdo řeší takovéto problémy, které se opakují, máme možnost použít před-připravené funkce a knihovny, které nám s implementací systému pomohou.

- My se pak můžeme soustředit pouze na vývoj konkrétní aplikace a ne na vývoj kontrolních modulů.
- Ušetříme tedy dost času, práce, ale především údržby.
- Pokud zjistíme chybu v dané knihovní funkci, tak ji předáme poskytovateli a on ji opraví za nás.

Framework - příklad

Možná definice:

Ucelená množina kompatibilních knihoven, které nám pomohou s řešením vývoje aplikace je framework.

Ponaučení jest:

neřešme již vyřešené problémy. Otázka má vždy být. Jsme první kdo řeší tento problém?

Pokud ne tak zkusme najít knihovny, které nám usnadní práci.

Frameworky

Testing frameworks
- Junit
- TestNG
- PHPUnit

Cache frameworks
- Jboss Cache

View components
- Ajax4JSF
- ICEFaces
- Trinidad
- Tomahawk

Další frameworky

Templating frameworks
- **Facelets**
- Velocity
- Tiles

Ajax frameworks
- Prototype
- Scriptacuo.us
- Jquery
- MooTools

Security frameworks
- Jboss Drools
- Acegi

Další frameworky

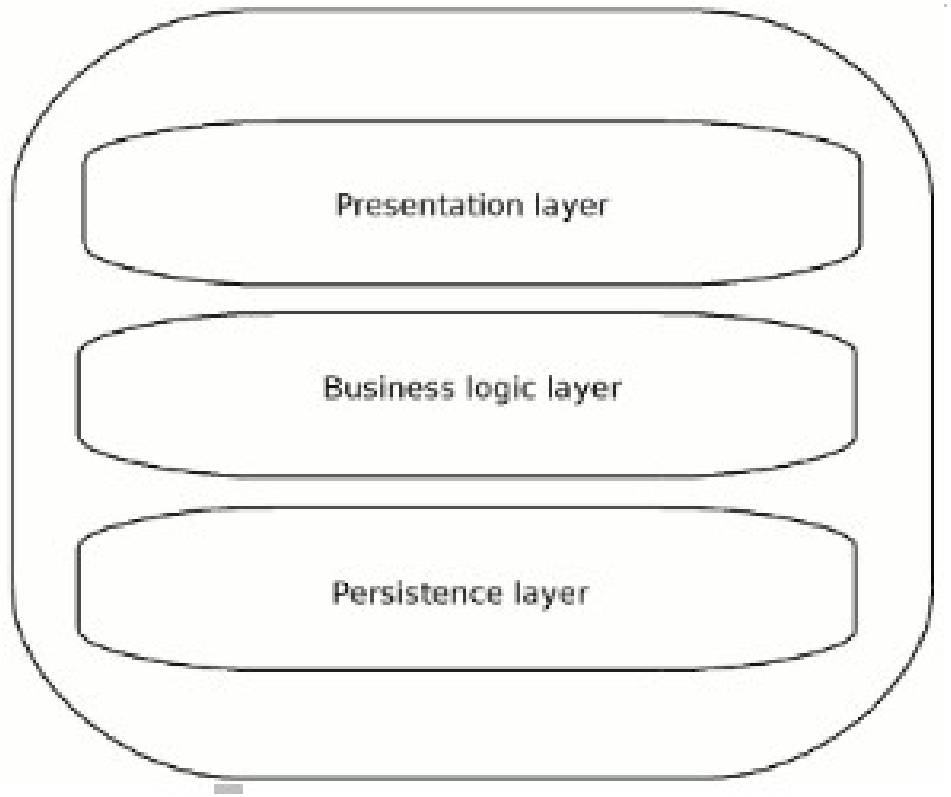
View Frameworks
- **JSF**
- JSP

Objektově relační mapování (ORM)
- **Hibernate**
- Cayenne

Aplikační frameworky
- **Seam** (web)
- **Struts** (web)
- Shale (web)
- Tapestry (web)
- **EJB**
- **Spring**

Java Platform, Enterprise Edition (Java EE)

Průmyslový standard pro implementace podnikových (enterprise) servis-orientovaných architektur SOA. Tato architektura umožňuje rozdělit aplikaci na tři hlavní části.



Prezentační vrstva - J2EE

Prezentační vrstva

- je vlastní uživatelské rozhraní, které má na starosti zobrazení dat ve vhodném rozložení
- odesílání akcí a reakce na ně
- předávání dat do nižší vrstev pomocí volání služeb.
- zajišťuje datovou konverzi a validaci,
- zobrazení zpráv uživateli,
- má na starosti přechody mezi sekcemi a další podobné operace.
- logiku uživatelského rozhraní.

V této vrstvě běžně najdeme Java server pages, Java třídy, HTML soubory, obrázky, PDF soubory a další. Tyto soubory jsou archivovány ve web archívu (WAR).

Prezentační vrstva - Jinde?

Prezentační vrstva

- Jak se liší ostatní frameworky of Javy?
- Co zde najdeme?

Business logika - J2EE

Business logika

- má na starosti business pravidla,
- požadavky systému z hlediska domény,
- transakce a dohlížení na správnost dat vzhledem k dalším datům aplikace.
- může zde být další webové služby či řešení závislostí.

Běžně zde lze nalézt kontrolní komponenty volající business pravidla. Tato vrstva se často liší mezi různými systémy.

Co najdemeve v J2EE?
Co najdme jinde?

Persistence - J2EE

Persistence

- je nejnižší vrstvou
- má na starosti serializaci, tedy uložení, dat.
- zodpovídá za získání dat z úložiště.
- zahrnuje návrhový model dat, který je běžně mapován na modelrelační.

triviální přiblížení : Java Database Connector (JDBC).

Jak vypadá vrstva J2EE?
Jak to je s dalšími frameworky

JavaServer Faces (JSF)

JSF je framework založený na komponentách a také na vzoru Model-View-Controller MVC

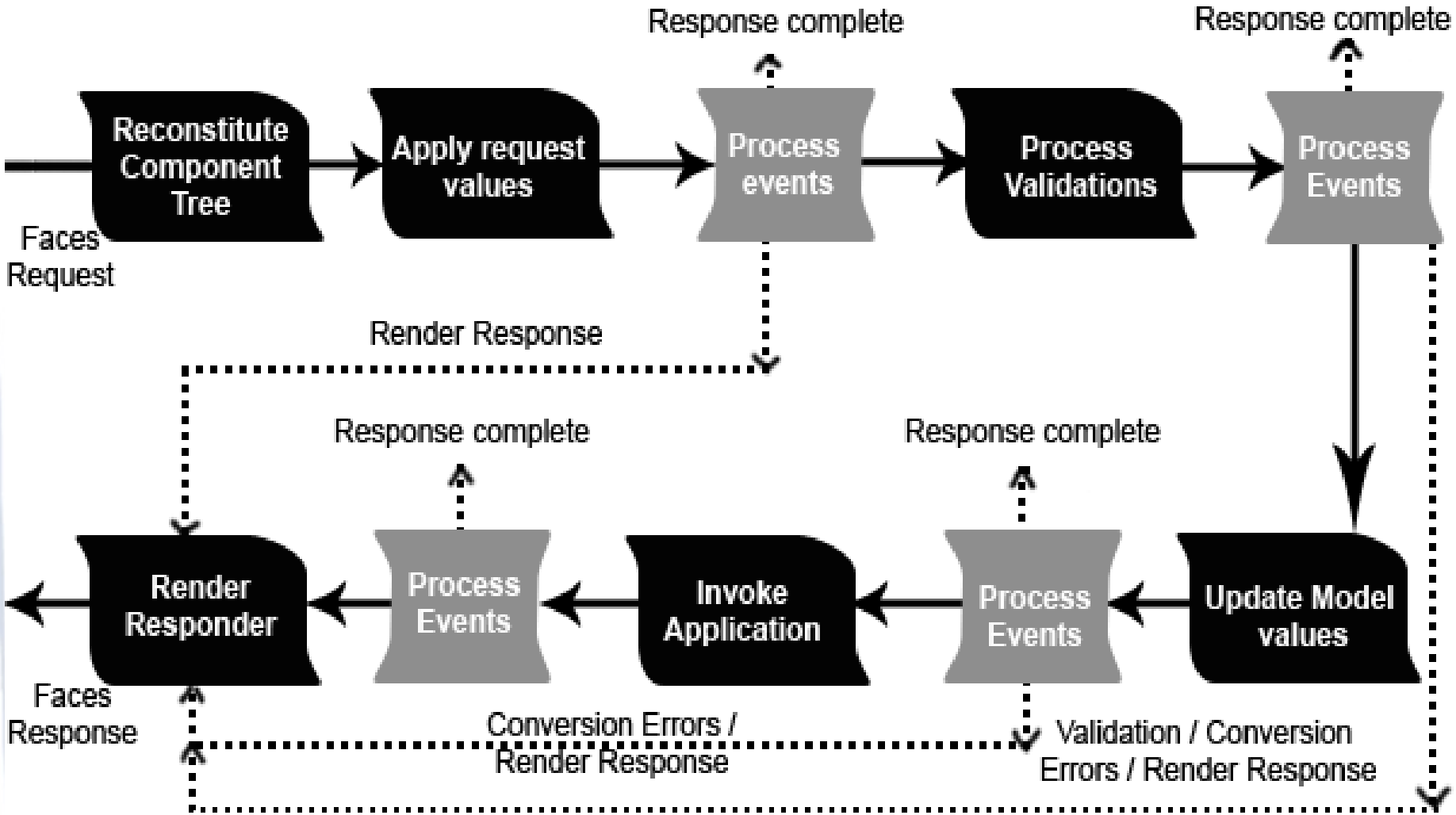
- zjednodušuje vývoj uživatelských rozhraní pro J2EE

Uživatelské rozhraní je svázáno s životním cyklem požadavku

Komponenty uživatelského rozhraní jsou nabízena o různých poskytovatelů (RichFaces, ICEFaces, Trinidad, Tomahawk..)

Co je MVC? Více v předmětu SI3
Co je návrhový vzor? Více v předmětu SI3, ASS

JavaServer Faces (JSF) – life cycle



JavaServer Faces (JSF)

Framework poskytuje

- API pro reprezenraci komponent a jejich vnitřních stavů,
- mezinárodní jazykovou podporu a přístup.
- Zobrazování komponent není limitováno jen do jazyka HTML.
  - Jelikož je uživatelské rozhraní definováno a postaveno na serverové straně, klientský prohlížeč pouze určí podporované formáty a uživatelské rozhraní je zobrazeno ve vybraném formátu.
  - HTML i pro WML...

JavaServer Faces (JSF)

Hlavní výhody jsou:

- použití komponent
- vestavěná validace datových vstupů,
- navigace mezi stránkami,
- řízení událostí,
- automatická datová konverze,
- veliký výběr poskytovatelů komponent,
- čisté dělení business logiky od prezentace,
- kratší vývojový cyklus, standard,
- IDE (integrated development environment – eclipse, netbeans),
- podpora a nezávislost zařízení
- a další.

Facelets

Rozšíření Facelets

Technologie rozšiřující JSF a možnosti vývoje.

- budování stromu komponent
- přehledné informace o chybách v systému při ladění
- snadná templetizace datových struktur či fragmentů
  - snadná templetizace datových struktur či fragmentů
    - snadná templetizace datových struktur či fragmentů

View komponenty

RichFaces

Je knihovna komponent pro aplikace psané v JSF, JSP.
Nejlépe ilustrováno zde:
http://livedemo.exadel.com/richfaces-demo/index.jsp

Trinidad

Je knihovna komponent pro aplikace psané v JSF, JSP.
Nejlépe ilustrováno zde:
http://myfaces.apache.org/trinidad/trinidad-api/tagdoc.html

ICEfaces...

Enterprise JavaBeans technology (EJB)

Technologie EJB je architektura komponent na serverové straně pro Java EE.

Umožňuje rapidní a zjednodušený vývoj distribuovaných, transakčních, bezpečných a přenositelných aplikací postavených na Javě.

V porovnání s Java RMI či CORBA je EJB jednodušší tím, že skrývá většinu implementace typické pro distribuované aplikace.

Enterprise JavaBeans technology (EJB)

Každá distribuovaná aplikace má mechanismus na vytvoření klientských a serverových proxy objektů ke komunikaci a mechanismus k informování o odstranění komponent.

EJB má základní myšlenku oddělení business logiky od persistence

Enterprise JavaBeans technology (EJB)

Entity bean

- modeluje koncepty tedy objekty z reálného světa.
  - Tedy například osoba, auto či letiště.
- Tyto objekty jsou pak uloženy do relační databáze.

Enterprise JavaBeans technology (EJB)

Session bean

- řídí procesy a úkoly, či koordinují aktivity.
- kontrolují správnost dat a jejich závislostí, časové konflikty a provádí volání rutin k uložení či načtení dat.
- řídí transakce.
- rozeznáváme dva typy **Stateful a Stateless**, tedy stavové a bezstavové.
  - Stavové si udržují konverzační stav při použití klientem. Metody jsou tedy závislé na stavu beany. Stav uložen v session serveru a nebo v kombinaci session serveru a parametru konverzace.
  - Stateless beany si neudržují žádný stav a metody jsou tedy plně nezávislé na stavu.

Enterprise JavaBeans technology (EJB)

Jelikož session i entity beany jsou cílené jen pro synchronní volání, tedy volání akcí od klienta, nemohou sloužit asynchronním zprávám.

Pro **asynchronní** řízení zpráv je zde třetí typ bean tzv. **Message-driven beans**.

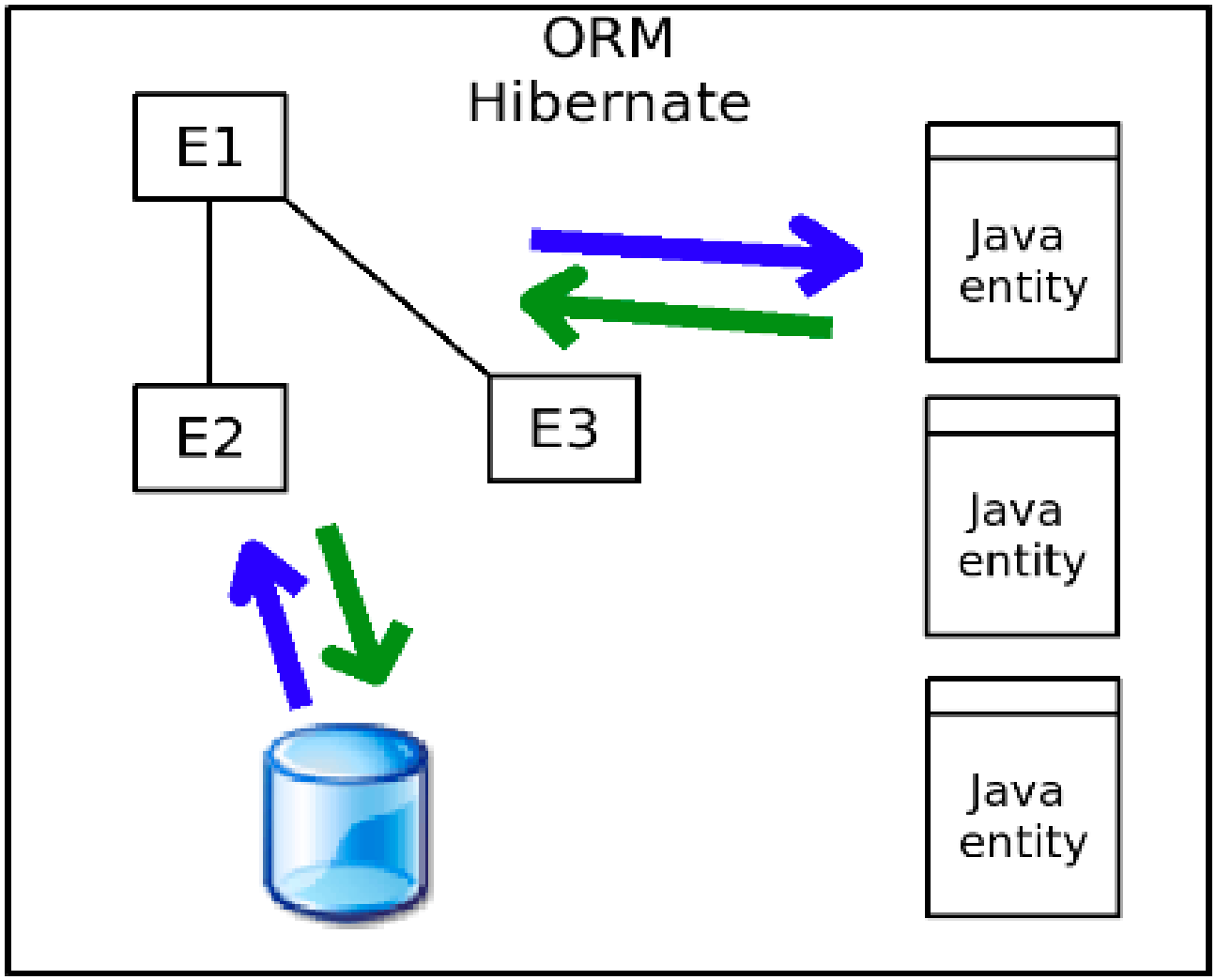
Hibernate – Object Relational Mapping (ORM)

Hibernate je open-source implementací služby objektově relačního mapování.

Mapuje se mezi objektovým doménovým modelem použitým pro Java aplikaci a relačním modelem databáze.

Zjednodušeně se na mapování lze dívat jako mapování mezi Java třídou a tabulkou databáze, tedy Javovský typ versus SQL typ.

Hibernate – Object Relational Mapping (ORM)



Object Relational Mapping (ORM)

Obecné problémy

- Jak implementovat UDT na úrovni SQL
  - (UDT User-defined datatype)

```
public class User {
    private String username;
    private String password;
    private Address address;
    // getters / setters
}
```

- Co takhle podtypy - jak implementovat polymorfní typy a asociace

Object Relational Mapping (ORM)

Obecné problémy

- Jak na identitu
  - a==b // místo v paměti
  - a.equals(b) // yup
    - ani jedno není ekvivalentem primárního klíče!!!
    - implementace korektní metody equals() pro entity objekt
- Asociace jako objektový odkaz vs. cizí klíč
- Jak navigovat v datech – minimalizace SQL dotazů
- Cena mapování JDBC/SQL

Object Relational Mapping (ORM)

Hibernate ale řeší i další věci jako:

- Mapování ORM
  - mapování objektů na tabulky
  - mapování asociací one to many, many to many
  - podtypy
- Persistence
  - Lazy initialization
    - Nahraj jen potřebné objekty z objekt grafu a ostatní nahraj až pokud jsou třeba

Object Relational Mapping (ORM)

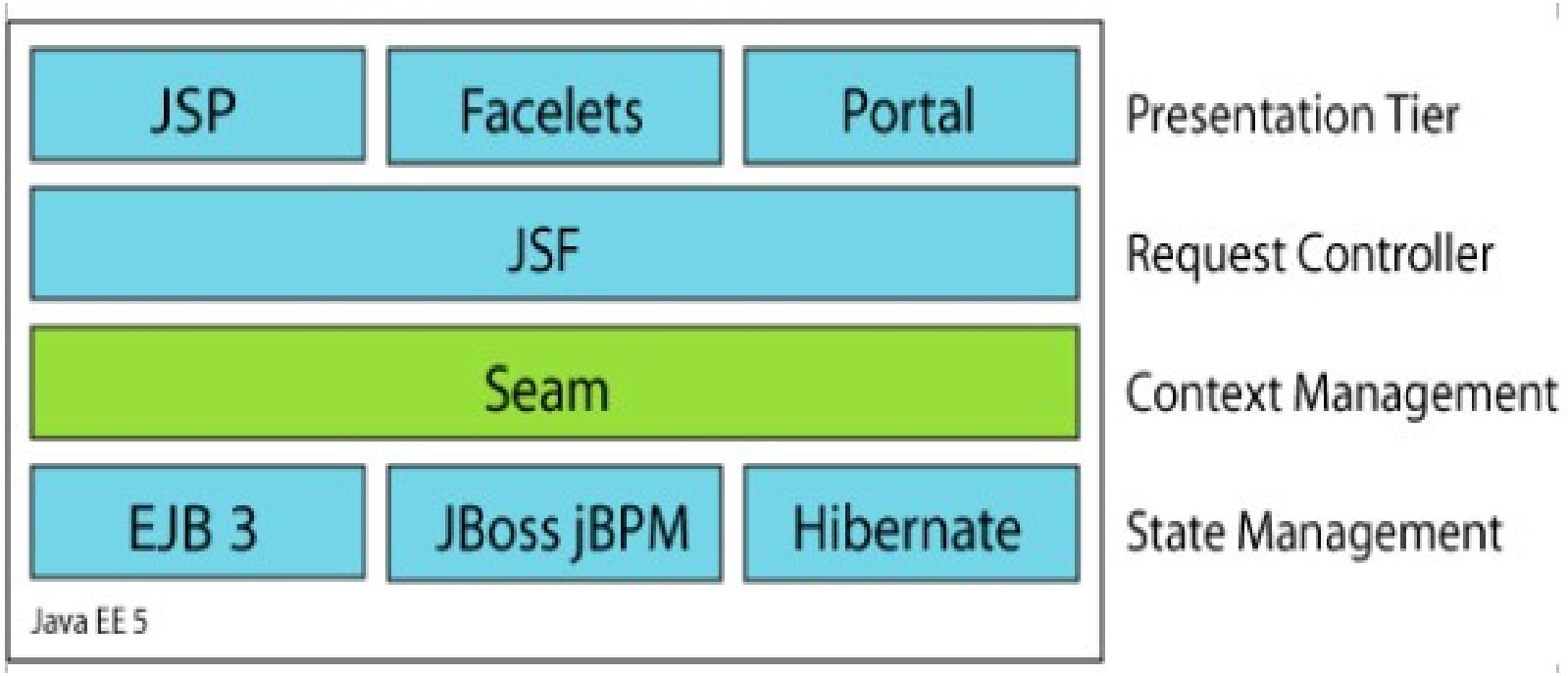
- Hibernate Query Language HQL
- Integrace
  - standalone aplikacemi
  - Java EE
    - servlets
    - EJB
- Mapování na různé databáze
- Konfigurace strategií
- Pomoc při ladění
- Dotazová a datová cache
- Nativní SQL
- Filtry

Seam web app. framework

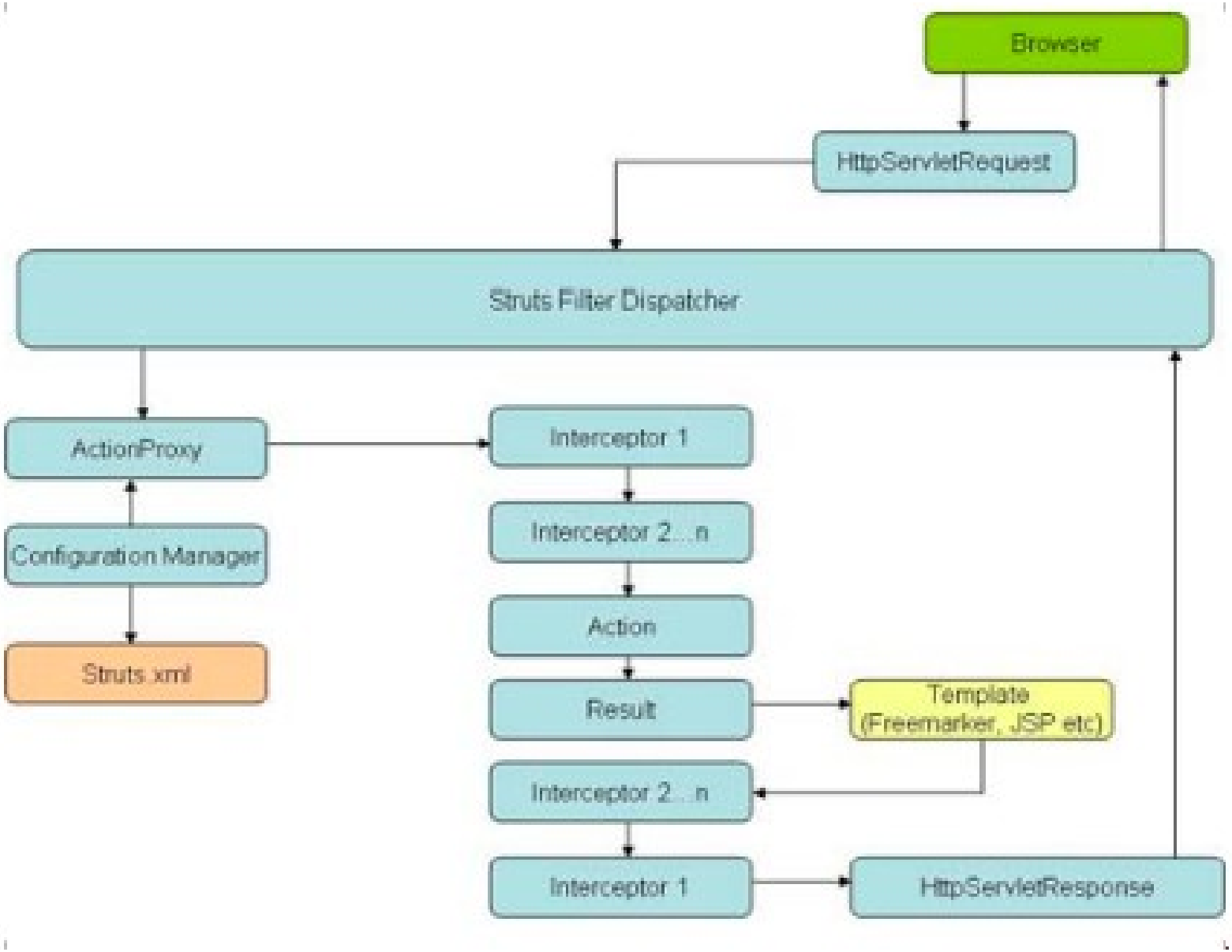
Seam je poměrně moderní framework postavený na definicích Java EE.

- open-source vývojová platforma
- budování Rich web aplikací v Javě
- integruje technologie jako
  - Asynchronous JavaScript and XML (AJAX),
  - JavaServer Faces (JSF),
  - Java Persistence (JPA),
  - Enterprise Java Beans (EJB 3.0)
  - Business Process Management (BPM) do jednotného řešení.

Seam


JSP
Facelets
Portal
JSF
Seam
EJB 3
JBoss jBPM
Hibernate
Java EE 5
Presentation Tier
Request Controller
Context Management
State Management

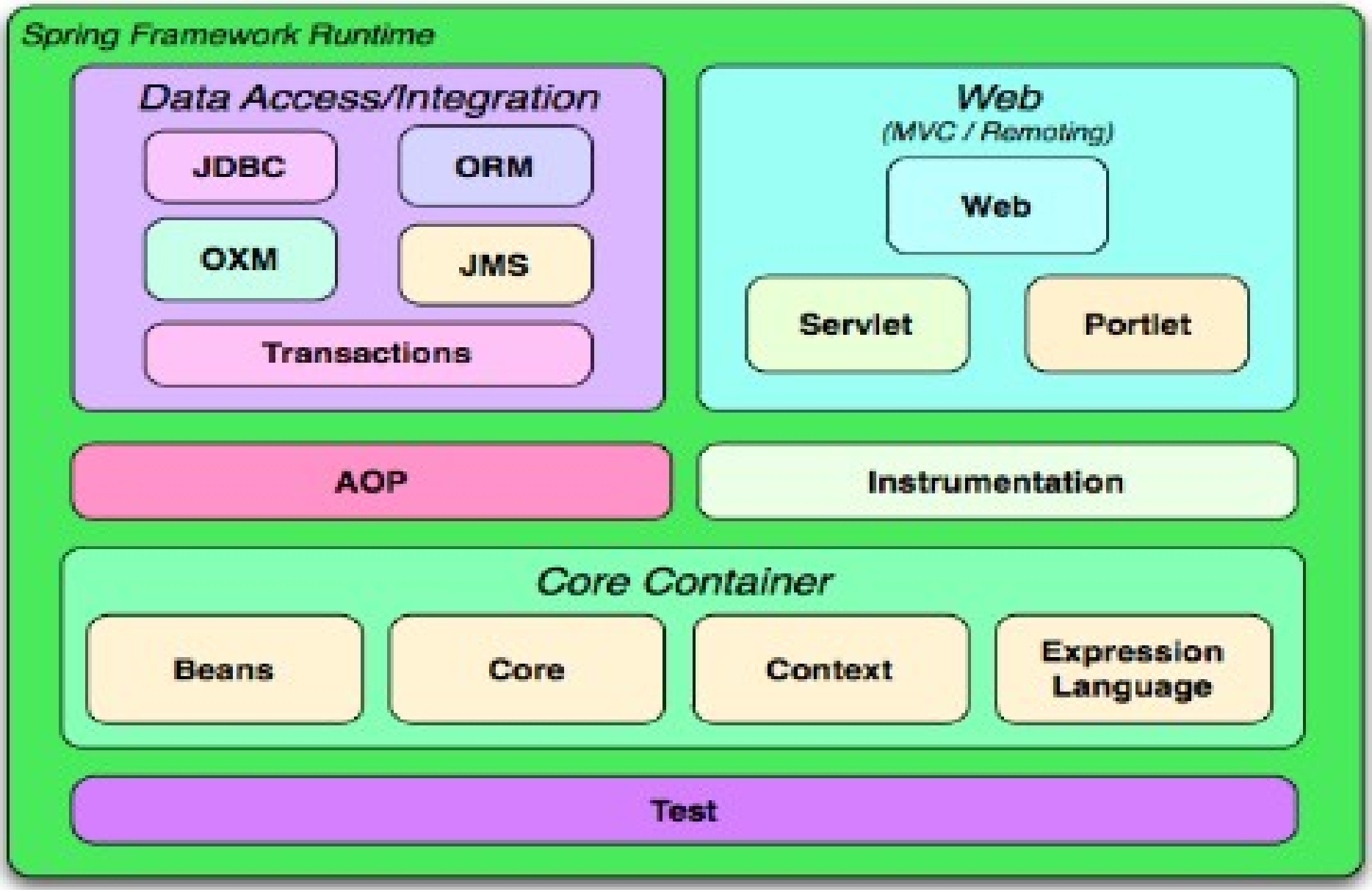
Struts 2


Browser
HttpServletRequest
Struts Filter Dispatcher
ActionProxy
Configuration Manager
Struts.xml
Interceptor 1
Interceptor 2...n
Action
Result
Template
(Freemarker, JSP etc)
Interceptor 2...n
Interceptor 1
HttpServletResponse

Spring

- Často se používá v kombinaci s Hibernate a JSF
- Existuje i verze Spring pro vývoj web aplikací
  - Je to jednodušší alternativa pro J2EE
  - Spring přináší kontext aplikace a tedy je možné stavět aplikace mimo kontejner (web server)
  - Spring se používá pro
    - vkládání závislostí (dependency injection)
    - pro AOP
    - řízení transakcí
    - remote procedure call
    - management operations
    - message handling.

Spring


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
(MVC / Remoting)
Web
Servlet
Portlet
AOP
Instrumentation
Core Container
Beans
Core
Context
Expression Language
Test

Co je Dependency Injection pattern (DI)?

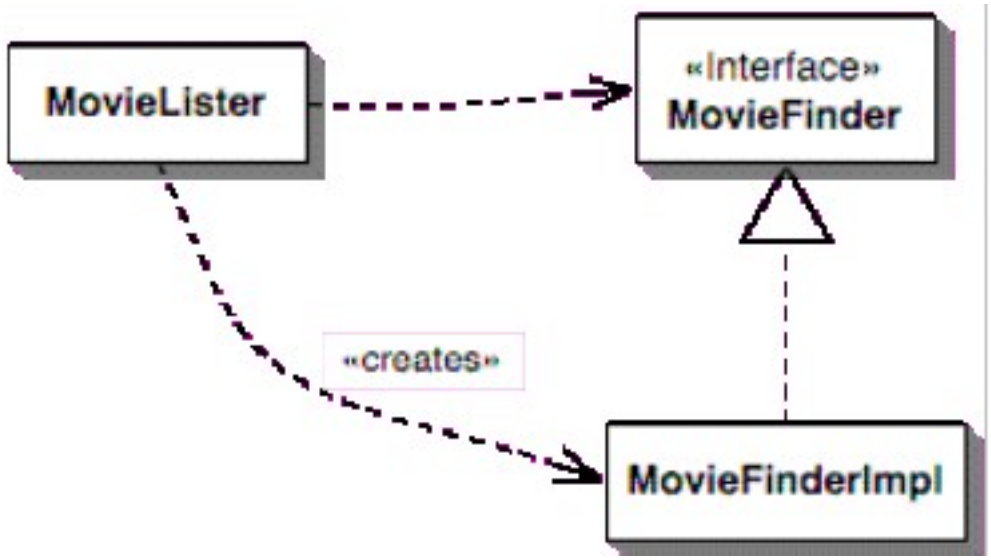
Běžně jsou Java aplikace poskládány z mnoha objektů které spolu spolupracují. Tyto objekty mají mezi sebou závislosti.

Propojení těchto objektů, by nemělo být násilné, jelikož může existovat více implementací služeb, které používáme. Závislosti tedy část řešíme na úrovni XML konfigurace, případně pomocí anotací.

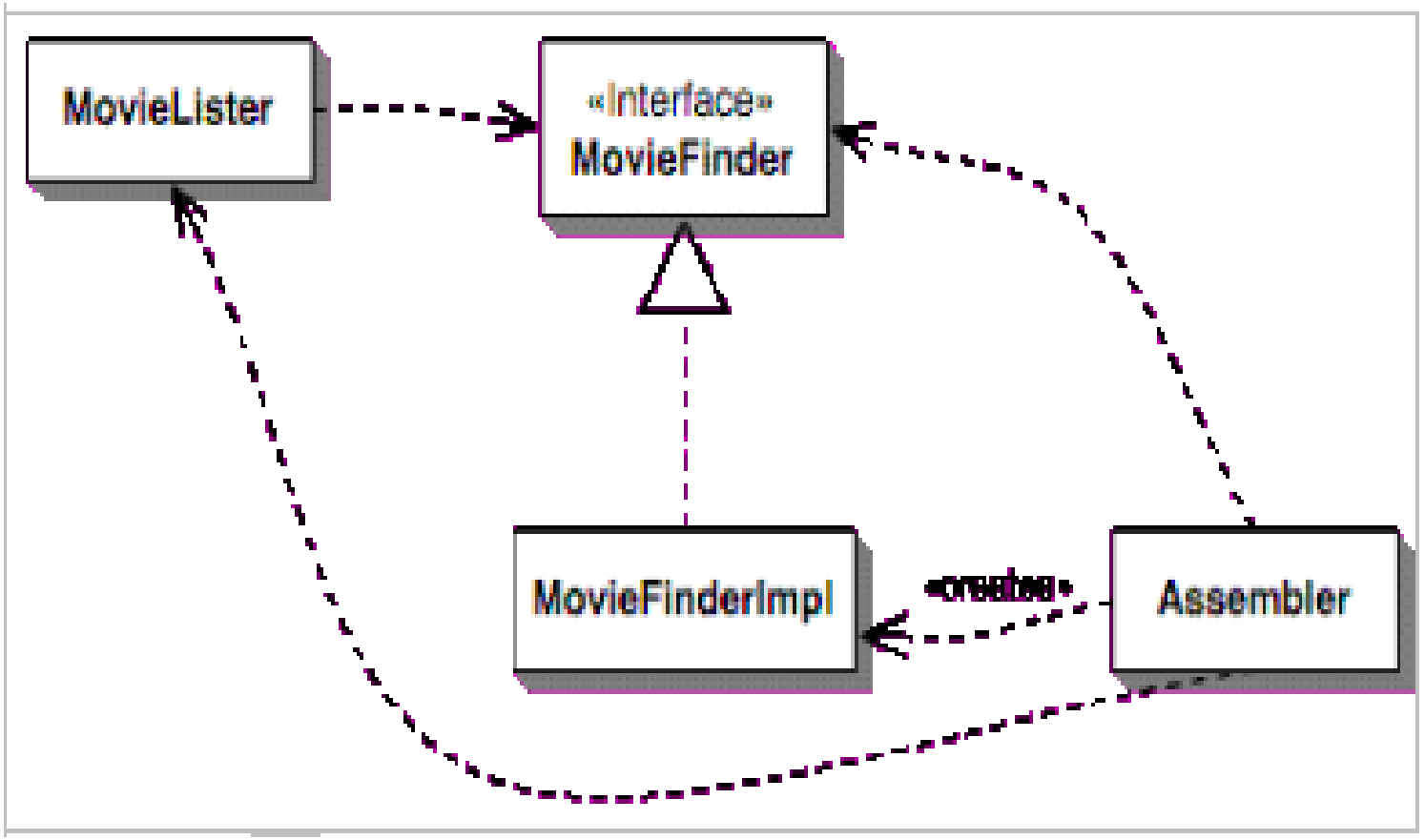
Příkladem je přepnutí do implementace pro testování tzv Dummy či Mock implementace.

S tímto vzorem dále úzce souvisí Inversion of control pattern (IoC).

Příklad


MovieLister
«Interface»
MovieFinder
«creates»
MovieFinderImpl

Příklad


MovieLister
«Interface»
MovieFinder
MovieFinderImpl
Assembler

Příklad

Setter Injection (Spring)

```
class MovieLister...
  private MovieFinder finder;
  public void setFinder(MovieFinder finder)
{
    this.finder = finder;
  }
```

Příklad

Setter Injection (Spring)

```
class ColonMovieFinder...
    public void setFilename(String filename)
{
        this.filename = filename;
    }
```

Příklad

```
<beans>
      <bean id="MovieLister" class="spring.MovieLister">
        <property name="finder">
              <ref local="MovieFinder"/>
          </property>
      </bean>
      <bean id="MovieFinder"
                    class="spring.ColonMovieFinder">
          <property name="filename">
            <value>movies1.txt</value>
          </property>
       </bean>
   </beans>
```

Příklad

```
public void testWithSpring()
                throws Exception {
    ApplicationContext ctx = new
         FileSystemXmlApplicationContext("spring.xml");
    MovieLister lister = (MovieLister)
                              ctx.getBean("MovieLister");
    Movie[] movies =
                lister.moviesDirectedBy("Sergio Leone");
    assertEquals("Once Upon a Time in the West",
                                    movies[0].getTitle());
    }
```

Service Locator