Tvorba informačních systémů na platformě J2EE

Petr Hetmánek (xhetman@fi.muni.cz)
Masarykova Univerzita, Fakulta Informatiky, Botanická 68a, Brno

Abstrakt

Rostoucí dostupnost internetu vede ke vzniku stále nových a komplexnějších informačních systémů. Značná část těchto systémů je implementovaná ve skriptovacích jazycích jako je PHP, ASP nebo JSP. Tyto technologie přímo neposkytují nástroje pro oddělení prezentační vrstvy od aplikační logiky nebo možnost modulárního návrhu systému. Bez využití těchto technik je rozsáhlejší informační systém značně nepřehledný a správa a rozšiřování vyžaduje obvykle velké úsilí.

Řešení těchto problémů poskytuje platforma Java 2 Enterprise Edition. J2EE definuje standardy pro vývoj vícevrstvých modulárních aplikací. Obsahuje podporu Enterprise JavaBeans, Java Server Pages (JSP), Java Servlety a technologie XML.

Článek popisuje možnosti, které nabízí tato platforma při tvorbě webových informačních systémů. Hlavní zaměření bude na architekturu Model 2, kterou poskytuje framework Struts. Ta společně s využitím technologie Enterprise JavaBeans poskytuje velice silný nástroj pro vytvoření robustní aplikace, kde je důraz kladen na znovupoužitelnost, přehlednost a modulárnost.

1. Úvod

V dnešní době jsou na funkce vznikajících aplikací kladeny stále větší požadavky. Společností, které se vývojem takových aplikací zabývají, stále přibývá a v rostoucí konkurenci je třeba mít nějakou výhodu. Klíčem k úspěchu je rychlý a kvalitní vývoj. Proto se snaží vytvořit architekturu, která by jim jej pomohla urychlit. V dnešním různorodém prostředí je třeba, aby aplikace měly v sobě implementováno velké množství služeb od různých výrobců. Implementace těchto zdrojů obvykle zabere až 50 procent vývojového času celé aplikace. Pro urychlení vývoje je třeba navrhovat aplikace tak, aby jednotlivé jejich části byly na sobě co nejméně závislé, a tedy bylo možné určité moduly použít i při vývoji jiných aplikací.

Architektury aplikací můžeme rozdělit do tří hlavních kategorií:

- **Monolitické aplikace**
  Jedná se o aplikaci složenou pouze z jednoho modulu obsahujícího veškerá data, která aplikace potřebuje, funkce pro manipulaci s těmito daty a také samotné zobrazení výsledku klientovi pomocí uživatelského rozhraní.
  Nevýhodou takto navržených aplikací je prakticky nulová znovupoužitelnost jak uložených dat, tak funkcí, které s těmito daty pracují.

- **Klient-server aplikace**
  Využití sítí zapříčinilo vznik další kategorie. Jedná se o aplikace, které se skládají ze dvou částí. Serverová část je obvykle databázový server sloužící pouze pro uložení dat, které je potom možné sdílet více klienty. Část na straně klienta se potom stará o funkce, které s těmito daty pracují a o zobrazení výsledku uživateli.
  Tento návrh již umožňuje znovupoužitelnost dat, a to protože jsou uložena na serveru. Avšak stále je tu omezení ve využití funkcí pro manipulaci s těmito daty, které obvykle bývají využívány více aplikacemi. Při návrhu více klientských rozhraní (např. klasická aplikace a webový klient) je nutné znovu implementovat funkčnost, která je pro oba totožná.

- **Vícevrstvé aplikace**
  Tento návrh přidává do klasické klient-server aplikace ještě jednu vrstvu, která se nazývá Business Logic a obsahuje veškeré funkce, která pracují s daty. Funkci této vrstvy obsluhuje aplikační server. Samotný klient se tedy již nezabývá prácí s daty, ale pouze volá požadované funkce aplikačního serveru a obdržené výsledky prezentuje uživateli. Pokud chceme vytvořit

více klientských rozhraní (např. klasická aplikace a webový klient) v této architektuře, již stačí pouze vytvořit několik variací, které volají stejné funkce aplikačního serveru a rozdílně je zobrazují uživateli.

Výhodou této architektury je tedy znovupoužitelnost nejen dat, ale také funkčnosti celé aplikace. Tím se podstatně redukuje čas při vývoji různých klientů. Další výhodou je také jistota, že funkce, které manipulují s daty na cílovém serveru, jsou bezpečné, neboť neběží přímo u klienta.

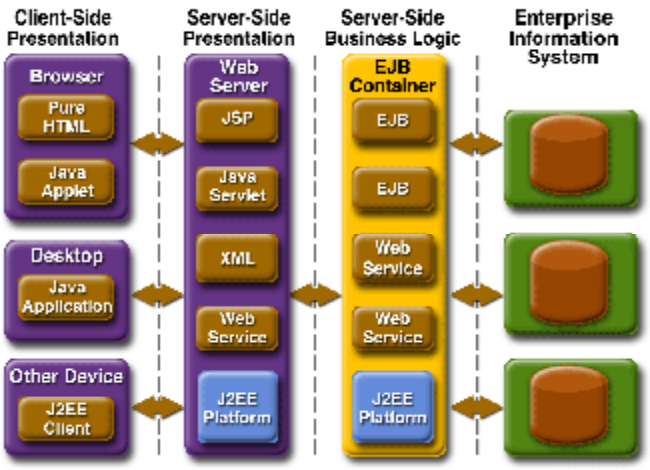
2. Platforma J2EE

Platforma J2EE (Java 2 Enterprise Edition, [1]) definuje standardy pro vývoj rozsáhlých vícevrstvých distribuovaných aplikací. Vývoj těchto aplikací zjednodušuje použitím komponent a služeb, které nabízí. Zároveň také využívá výhod programovacího jazyka Java, které spočívají zejména v přenositelnosti mezi různými operačními systémy.

Součástí této platformy je plná podpora Enterprise JavaBeans (které budou popsány později), Java Servlets, JavaServer Pages a technologie XML.

Hlavní myšlenkou J2EE je tvorba samotné aplikace z komponent. Proto je možné práci rozdělit mezi několik samostatných vývojových týmů a každý se může zabývat jinou části aplikace. Například komponenty pro interakci s uživatelem nebo komponenty, které naplňují hlavní činnost aplikace.

Pro své komponenty nabízí platforma J2EE řadu služeb, které jsou zajišťovány automaticky – jako např. správa životního cyklu komponent, správa transakcí nebo management zdrojů. Proto se vývojář nemusí zabývat tvorbou věcí, které jsou pro většinu aplikací téměř totožné, a může se plně soustředit na vývoj samotné aplikace.



Obrázek 1: Aplikační model platformy J2EE (převzato z [1])

Na obrázku 1 vidíme J2EE aplikační model, který slouží jako doporučení pro návrh aplikací s plným využitím možností, které tato platforma nabízí. Myšlenka spočívá v rozdělení částí aplikace do několika vrstev. První je prezentační vrstva na straně klienta. Tato část se zabývá grafickým zobrazením výsledných dat přímo uživateli. Může být tvořena například HTML stránkou, Java Appletem nebo aplikací, která zobrazuje data. Druhou vrstvu tvoří prezentační vrstva na straně serveru. Ta se stará o přípravu dat do formátu, kterému porozumí prezentační vrstva klienta. K vytvoření takového formátu se obvykle používá servletů nebo JSP. Může se však také jednat jen o předpřipravená data ve formátu XML. V další vrstvě se již dostáváme k samotnému jádru aplikace. Pojmem Business Logic je označována ta část, která přímo manipuluje s daty a obsahuje hlavní funkce aplikace. V této části platforma J2EE nabízí a doporučuje použití Enterprise JavaBeans. Poslední, nejspodnější, vrstvu obvykle tvoří některý z databázových serverů.

Výhodou vícevrstvé architektury je vzájemná nezávislost jednotlivých vrstev. Důležité je to hlavně u vrstvy prezentační a Business Logic. Toho lze využít při návrhu robustní aplikace, která bude mít

jak webový interface s využitím JSP a servletů, tak klientskou aplikaci. V těchto případech stačí funkce naprogramovat v rámci EJB a pouze v prezentační vrstvě oddělit zobrazení výsledných informací. Při použití klasické klient-server architektury by bylo nutné jednu věc programovat dvakrát.

3. Enterprise JavaBeans

EJB [2, 3] jsou objekty, které v sobě obsahují hlavní funkčnost celého systému a neběží přímo u klienta, ale distribuovaně na straně serveru. Takový server se nazývá EJB kontejner nebo také aplikační server.

EJB kontejner je zodpovědný za distributivnost komponent a správu služeb jako např. transakce, bezpečnost, souběžnost a stálost. Díky tomu, že se o to stará server, programátor se může soustředit pouze na implementaci funkčnosti daného systému. Tyto důležité služby pro distribuované systémy nemusí již programovat, pouze u každé komponenty nastaví parametry, které ovlivňují chování výše zmíněných služeb. Tím se stává vývoj mnohem jednodušším a rychlejším.

Mezi služby, nabízené EJB patří například:

- **Transakční model** - umožňuje vývojáři specifikovat vztahy mezi metodami, které spolu tvoří jednu transakci.
- **Bezpečnostní model** – prozměnu umožňuje specifikovat role různých oprávnění, pod kterými do systému přistupují různí klienti. Pouze klient, který volá funkci některého objektu z aplikačnímu serveru s požadovanou rolí oprávnění, dostane zpět odpověď.
- **Služba JNDI** (Java Naming and Directory Service) – ta umožňuje EJB komponentu vystavit pod určitým jménem, aby ji později klient mohl najít a přistupovat k ní. Jedná se o jednotné rozhraní, které umožňuje přistupovat k objektům umístěným na aplikačním serveru pomocí programátorem zvoleného jména.
- **Model vzdáleného přístupu** – zajišťuje takovou službu, která umožňuje to, aby ve chvíli, kdy je na aplikačním serveru vytvořen objekt a klient na něj získá odkaz, tak již dále pracoval s tímto objektem, jako by byl umístěný přímo u něj a o samotnou komunikaci se nemusel starat.

Proč používat EJB?

- Nezávislé na architektuře aplikačního serveru.
- Naprogramováno v Javě a nezávislé na operačním systému.
- Zavádí role oprávnění při přístupu k jednotlivým objektům a jejich metodám.
- Stará se o transakční zpracování.
- Nabízí podporu pro distribuované transakce.

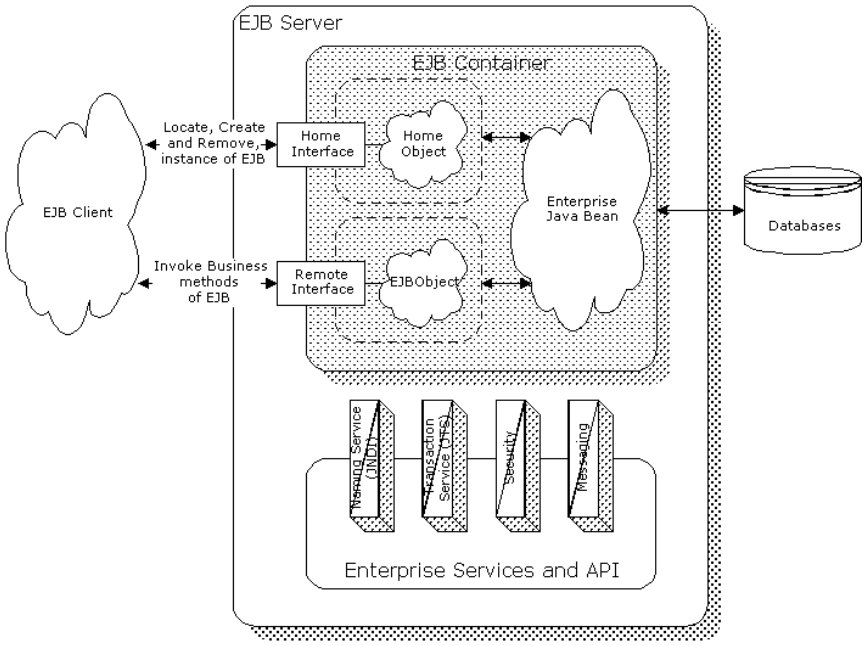
Architektura Enterprise JavaBeans

Obrázek 2 zachycuje celou architekturu Enterprise JavaBeans, kterou si popíšeme. *EJB server* poskytuje prostředí, ve kterém mohou běžet EJB kontejnery. Primárně se stará o multiprocessing, kontrolu zátěže a přístupu EJB kontejneru k disku. Zajišťuje, aby běžící kontejner byl přístupný z vnějšího světa. Může také obsahovat další součásti, jako např. podporu SSL, služby pro správu transakcí a další. Obvykle se jedná o servery s operačními systémy Linux nebo Windows.

Hlavní součástí architektury je *EJB kontejner*, který funguje jako rozhraní mezi EJB komponenty a funkcionalitou na straně klienta. Stará se o běh samotných EJB objektů a poskytuje jim potřebné služby přes standardní rozhraní, které je definováno ve specifikaci. Většina výrobců ke kontejnerům přidává také další služby, jako např. webový server.

Home Interface je rozhraní, které definuje metody pro nalezení, vytvoření a odstranění instance EJB objektu. *Home Object* je implementace metod tohoto rozhraní. Vývojář EJB objektů pouze definuje rozhraní Home Interface a kontejner automaticky vygeneruje příslušný objekt.

Remote Interface je rozhraní definující metody, které poskytuje samotný EJB objekt. *EJBObject* implementuje metody tohoto rozhraní pro přístup k těmto funkcím v Enterprise JavaBean. Tyto metody jsou automaticky generovány kontejnerem.



Obrázek 2: Architektura Enterprise JavaBeans (převzato z [2])

Hlavní objekt ***Enterprise JavaBean*** je umístěn uvnitř kontejneru a nikdo jiný kromě kontejneru samotného by k němu neměl přistupovat. Díky tomuto nepřímému přístupu má kontejner možnost pracovat s důležitými službami, jako je oprávnění přístupu nebo transakční zpracování.

EJB klient potom pomocí JNDI najde specifický kontejner, který vyvolá příslušné metody pro vytvoření EJB objektu. Klient dostane pouze odkaz na EJBObject, nikdy na samotnou instanci Enterprise JavaBean. Kdykoliv klient vyvolá nějakou metodu, dostane tento požadavek EJBObject, který ji pošle dále příslušné metodě samotného objektu až po provedení nezbytných funkcí týkajících se bezpečnosti a dalších služeb. Klient používá Home Object k nalezení, vytvoření a zrušení instance Enterprise JavaBean.

Entity Beans

Enterprise JavaBeans jsou tří druhů.Prvním z nich jsou Entity Beans, které slouží pro modelování dat. Reprezentují samotná data uložená v databázi. Jedna instance takového objektu odpovídá jednomu záznamu v databází. Entity beans bývají přímo synchronizovány s databází, takže při vytvoření nové instance automaticky vznikne nový záznam a naopak při smazání se záznam odstraní.

Entity Beans můžeme rozdělit na dva typy. Bean-Managed Persistence (BMP) je objekt, kde trvalost je plně ve správě Enterprise JavaBean. Když kontejner uzná za vhodné, že je třeba data synchronizovat s obsahem databáze, zavolá jednu z metod pro uložení nebo obnovení dat. Vývojář zajišťuje, jak budou tyto metody implementovány. Stejně tak musí zajistit, aby při vytvoření nové instance objektu byl vytvořen nový záznam v databázi a také při jejím zrušení příslušný objekt smazán.

Druhým typem jsou Container-Managed Persistence (CMP), kde se o trvalost stará kontejner. Vývojář pouze definuje tabulky a přiřazení parametrů objektu pro danou tabulku a o zbytek už se postará kontejner. Když uzná za vhodné, tak automaticky naplní objekt daty uloženými v databázi, případně data zpět uloží nebo vytvoří nový záznam, či některý zruší. Tento typ má výhodu zejména v tom, že není závislý na konkrétní databázi. V příslušném konfiguračním souboru stačí pouze určit typ databáze a kontejner už sám bude vědět, jak data ukládat.

Session Beans

Session Beans slouží pro modelování uživatelských cílů aplikace. Jedná se o akce, které aplikace vykonává. Session beans jsou dvojího typu. Stavové se používájí v případech, kdy je nutné si během

jednoho sezení pamatovat některé údaje nebo sledovat akce klienta. Používají se pro obsluhu procesů, které se dělí do volání více metod nebo transakcí. Při změně stavu ve volání jedné metody je tento stav přenesen i do volání ostatních metod. V praxi se může jednat například o nákupní košík internetového obchodu, kde si ve stavu udržujeme jeho obsah.

Procesy, které lze uskutečnit pomocí jediného požadavku, nepotřebují mít uložený stav. Pro tento typ komunikace se používají bezstavové session beans. Definujme si například objekt pro sčítaní dvou čísel. Na vstup dostane dvě čísla, ty sečte a vrátí výsledek. V rámci této akce je jakékoli ukládání stavu zbytečné.

Rozdíl mezi entity beans a session beans je v době jejich trvání. Zatímco entity beans přetrvávají stále ve formě záznamů v databázi, trvání session beans je obvykle omezeno na jedno sezení, kdy klient komunikuje s aplikačním serverem.

Message-Driven Beans

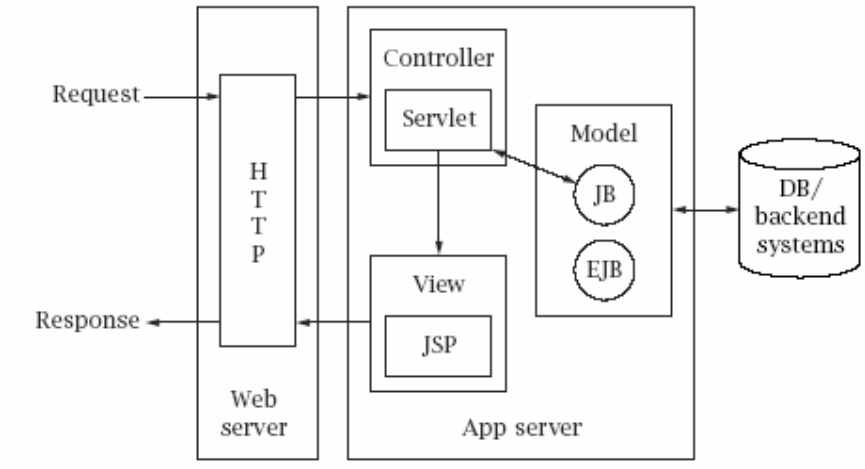
Posledním typem jsou Message-Driven Beans. Jedná se o objekty, které jsou svým účelem velice podobné session beans, ale lze s nimi pracovat pouze pomocí zasílání zpráv. Tento typ je možno využít, pokud je požadován asynchronní přístup. Zprávy jsou postupně zasílány objektu označovanému jako Middleman, který leží mezi klientem a serverem. Pouze poslouchá, dostává zprávy a posílá je dál – na místo jejich určení. Pokud příjemce zprávy není v danou chvíli dostupný, zpráva zůstává u tohoto zprostředkovatele a je mu zaslána později. Nevýhodou takového systému však je zvýšená režie, potřebná pro zpracovávání těchto zpráv a nemožnost zpracování výjimek, pokud k nim dojde.

4. Struts

V poslední části se budeme zabývat frameworkem Struts [4, 5] od Apache Software Foundation. Jedná se o rámec, který vypomáhá vývojáři při návrhu webových systémů s využitím výše zmíněného aplikačního modelu platformy J2EE. Pod pojmem rámec (framework) se rozumí částečně hotová aplikace, kterou po specializaci můžeme použít na naše konkrétní účely. Aplikační rámec nabízí vývojářům znovupoužitelnou strukturu, na které mohou jednoduše vystavět vlastní aplikace.

Rámec Struts je postaven na architektuře Model 2, která vychází z architektury MVC (Model-View-Controller) a dále ji specifikuje přímo pro vývoj webových systémů. Pro prezentační vrstvu, tedy vrstvu, která se zabývá zobrazení informací klientovi, se používají Java Server Pages (JSP) a pro práci s Business Logic vrstvou se používají Java servlety.

Celý proces popisuje obrázek 3. Klient pošle požadavek na řadič (Controller), který podle konfiguračního souboru struts-config.xml rozhodne, jak bude požadavek zpracován. Nejjednodušší způsob je přesměrování na JSP stránku, která pouze provede zobrazení informace.



Obrázek 3: Architektura frameworku Struts (převzato z [5])

V případě, že požadavek potřebuje pracovat s Business Logic systému, řadič předá řízení některé akci ve vrstvě Model. Samotná operace může být provedena přímo v akci nebo může zavolat metodu některé Enterprise JavaBeans. Jak jsme popisovali dříve, takový přístup zvyšuje znovupoužitelnost celého systému. Jakmile se provede požadovaná akce, aplikace se dostane do dalšího stavu a servlet, který obsluhoval akci vyvolá opět JSP stránku, která zobrazí uživateli výsledek.

Rozdělení na vrstvy umožňuje zejména rozdělení práce na projektu mezi více programátorů nebo programátorských týmu. Zatímco jeden se může zabývat pouze hlavním jádrem aplikace, kterou tvoří vrstva Business Logic, druhému stačí pouze znát formát dat, který dostane prezentační vrstva, a ten zobrazit. Z toho také přímo plyne, že změnu designu aplikace může provést i programátor, neznalý funkčních detailů (protože vůbec nebude zasahovat do kódů zajišťujících funkci aplikace). A stejně tak, pokud se provádí nějaká změna ve funkcích aplikace, není třeba zasahovat do zobrazování dat uživateli, stačí pouze upravit metody v Enterprise JavaBean.

Asi největší výhodou použití tohoto návrhu je to, že pokud se programátor v budoucnu rozhodne k aplikaci přidat další rozhraní (např. aplikaci pro Windows nebo Linux), nemusí již znovu programovat hlavní funkce, ale stačí naprogramovat uživatelský interface a rozhraní, které bude používat již hotové objekty EJB.

Struts kromě architektury nabízí navíc také několik tříd a plug-inů, které vývojářům usnadňují některé zdlouhavé procedury. Jedním z těchto nástrojů je například jednoduše implementovatelný způsob pro validaci formulářů. Programátor pouze vytvoří potomka třídy ActionForm, kterému přetíží metodu validate() a v ní může implementovat testy, které ověří, zda byly položky formuláře správně vyplněny. O samotné volání této třídy se již stará řadič rámce Struts.

Druhou možností je použít plug-in Validator, který tyto validace umožňuje nastavovat přímo v konfiguračním souboru typu XML, takže vývojář nemusí vytvářet potomky třídy a programovat metody, ale pouze vybraným prvkům formuláře v tomto konfiguračním souboru nastavit vlastnosti, které musí před odesláním splňovat.

Dalším užitečným prvkem je vícejazyčná podpora, kdy se všechny texty ukládají do odděleného souboru a programátor používá identifikátory, které těmto textům odpovídají. Pro jiný jazyk potom stačí vytvořit další soubor a v něm uvést překlady těchto textů.

Velice užitečným plug-inem při návrhu uživatelského rozhraní jsou dlaždice (Tiles). Ty umožňují použít šablony, kterými se definuje vzhled stránek bez uvedení jejich obsahu. Konkrétní obsah se do stránek doplňuje až za běhu

5. Shrnutí

Cílem článku bylo čtenáře informovat o možnostech, které nabízí platforma J2EE a framework Struts pro vývoj robustních distribuovaných systémů. Při použití těchto technologií dojde ke zrychlení vývoje aplikací. Díky vícevrstvé architektuře je u takto navržených aplikací snadnější údržba a rozšiřitelnost. Současně architektura umožňuje snadnější a přehlednější rozdělení práce mezi více programátorů nebo týmů.

6. Literatura

1. Java Technology [online]. Sun Microsystems. Dostupný z WWW: <http://java.sun.com>
2. SULLINS, Benjamin G.; WHIPPLE, Mark B. EJB Cookbook. Manning, 2003. ISBN 1930110944.
3. MONSON-HAEFEL, Richard; BURKE, Bill; LABOUREY, Sacha. Enterprise JavaBeans. 4. vyd. O'Reilly, 2004. ISBN 0-596-00530-X.
4. HUSTED, Ted N.; DUMOULIN, Cedric; FRANCISCUS, Georgie; WINTERFELDT, David. Struts In Action. Manning, 2003. ISBN 1930110502.
5. SPIELMAN, Sue. The Struts Framework: Practical Guide for Programmers. Morgan Kaufmann, 2003. ISBN 1558608621.