VÝVOJ DISTRIBUOVANÝCH APLIKACÍ V SYSTÉMU PLAANT

Rudolf Pecinovský
Amaio Technologies, Inc., rudolf@pecinovsky.cz

ABSTRAKT:
Systém *Plaant* je nástrojem pro vývoj a následnou údržbu distribuovaných databázových aplikací. Nástrojem, který umožňuje, aby vývojář definovat pouze strukturu aplikace, a *Plaant* následně sám naprogramuje její kostru. Na vývojáře již zbývá pouze doplnit kód řešící některé specifické detaily konkrétní aplikace. Článek seznamuje s hlavními rysa nástroje *Plaant Tools*, který slouží k definici struktury podpůrné databáze a návrhu požadovaného uživatelského rozhraní.

KLÍČOVÁ SLOVA:
distribuované aplikace, J2EE, framework, vývoj

1 ÚVOD

Plaant je komplexní systém. Je to zároveň sada nástrojů pro vytvoření aplikace a zároveň prostředí, v němž na serveru aplikace běží, které komunikuje s koncovými uživateli prostřednictvím klientského GUI, a které umožňuje administrátorovi správu celé aplikace.

Vývojáře připravující aplikace běžící pod systémem *Plaant* bychom mohli rozdělit do dvou skupin:

- *návrháři aplikací*, kteří nemusejí být programátory, navrhují základní kostru aplikace a na ni navazující koncepci grafického uživatelského rozhraní – ti s výhodou využívají komfortu a možností nástroje *Plaant Tools*, jehož výstupem je sada XML dokumentů popisujících celou aplikaci a
- *návrháři komponent*, kteří navrhují kód, jenž ošetřuje některé aplikačně specifické funkce, a kteří programují v jazyce Java a pracují v některém standardním IDE.

Tento příspěvek se bude zabývat možnostmi, které nabízí *Plaant Tools* a na nichž je možno současně demonstrovat, jaký přínos může nasazení systému *Plaant* přinést.

2 ANALYTICKÁ PŘÍPRAVA – NÁVRH AGEND

Před tím, než-li se návrhář pustí do vlastního návrhu aplikace, je třeba celé zadání nejprve zanalyzovat a mimo jiné navrhnout databázovou strukturu budoucí aplikace, funkce, které bude aplikace plnit, a podobu uživatelského rozhraní.

Typický objekt databázové aplikace modeluje nějakou datovou entitu – tvoříme-li např. databázi pro knihovnu, budeme (mimo jiné) modelovat jednotlivé knihy, nakladatele, autory, čtenáře atd.; tvoříme-li databázi pro účtárnu, budeme modelovat účty, faktury, zákazníky, dodavatele, položky faktur atd.

Kolem každé modelované entity vzniká celá agenda činností, které jsou s ní spojeny. Tuto agendu má v aplikacích budovaných pod systémem *Plaant* na starosti samostatná komponenta, kterou označujeme termínem *agenda*.

Rozdíl mezi databázovou tabulkou používanou v relačních databázích a agendou je obdobný jako rozdíl mezi klasickým datovým typem ve strukturovaných jazycích a objektovým datovým typem v jazycích objektově orientovaných. Agenda představuje modelovanou entitu jako celek s jejím funkčním i datovým obsahem, a to včetně definice uživatelského rozhraní.

Agendy mají své přímé protějšky v kódu. Každá agenda je reprezentována třídou, která zabezpečuje její propojení s celým systémem. U rozsáhlejších aplikací je proto např. výhodné využít možnosti rozmístění agend do balíčků (packages). Celá aplikace se tak stane přehlednější a snáze spravovatelnou.

2.1 Data

Agendy bývají v relační databázi většinou reprezentovány jako tabulky, ale není to pravidlo. Agendy jsou totiž obecnější. Na jednu stranu mohou existovat tzv. agendy bez dat, které nemají v databázi žádnou přímou reprezentaci a jsou realizovány čistě programově, na druhou stranu však mohou existovat také agendy, které jsou v relační databázi reprezentovány celou skupinou tabulek (není to tak řídký případ, jak by se na první pohled zdálo).

Obdobně jako objektové datové typy mohou i agendy být potomky jiných agend. *Plaant* přitom, obdobně jako např. *Java*, podporuje pouze jednoduchou dědičnost.

Agendy mohou vytvářet *stromy dědičnosti*, v nichž dceřiné agendy přebírají do svých záznamů všechna pole svých rodičovských agend. Možnost využití mechanizmu dědičnosti výrazně zjednodušuje analýzu a na ni navazující realizaci některých typických zákaznických požadavků.

Jak bývá v relačních databázích zvykem, agendy mohou obsahovat odkazy na jiné agendy, přičemž *Plaant* rozeznává dva druhy odkazů: jednoduchý odkaz směřující na konkrétní záznam a tabulkový odkaz odkazující na celou množinu záznamů. Při vytváření odkazů typu M:N přitom nemusí vývojář explicitně vytvářet příslušnou mezitabulku, protože ji *Plaant* dokáže vytvořit a spravovat sám.

Agendy mohou obsahovat nejenom odkazy na jiné agendy, ale mohou také obsahovat jiné agendy jako své komponenty. To ulehčuje analýzu i realizaci častých úloh, které je tak možno předem připravit jako samostatné komponenty a poté je do vytvářené aplikace pouze vložit.

2.2 GUI

Součástí definice agendy je nejenom návrh jejích polí, ale také návrh formulářů, jejichž prostřednictvím budou uživatelé s agendou komunikovat. Formuláře pro plaantové aplikace jsou platformně nezávislé, takže jedna definice formuláře specifikuje jeho podobu jak např. pro desktopového klienta, tak pro webového klienta. Oba formuláře sice vypadají rozdílně, ale jsou postaveny nad stejnou definicí (viz obr. 1, 2).

2.3 Další součásti

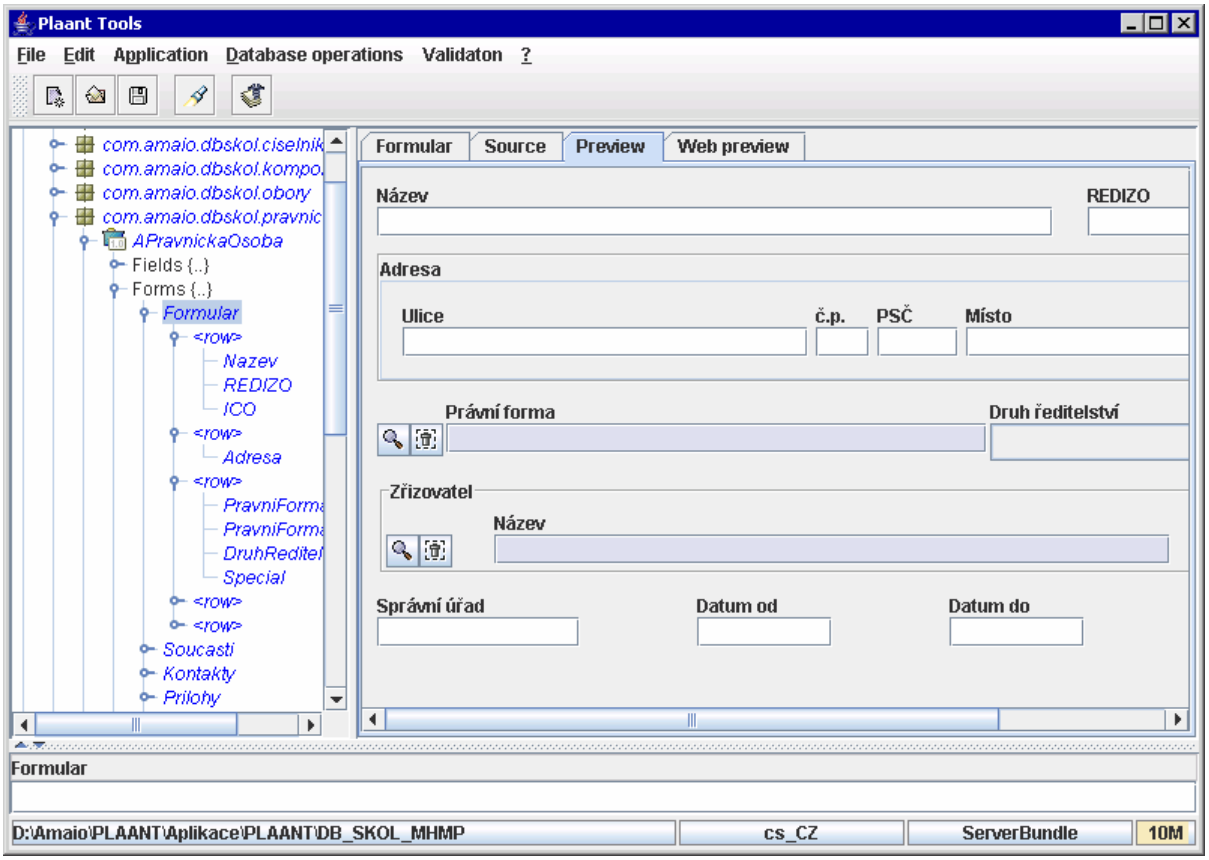
Součástí definice agendy jsou i definice:

- *speciálních funkcí*, které realizují některé nestandardní činnosti,
- *stavových diagramů* popisujících jednotlivé fáze, kterými některé objekty procházejí spolu s množinou operací, které jsou v jednotlivých stavech povolené,
- *výstupních sestav*, které umožňují definovat zveřejňovaná data, jejich formáty a cílové zařízení, na něž sestava směřuje.

Těmito součástmi se ale příspěvek nezabývá.

3 NÁVRH STRUKTURY DAT

Po analýze nastupuje návrh kostry aplikace. Návrhář aplikace navrhne za pomoci programu *Plaant Tools* základní objektové schéma, tj. definuje, které agendy bude aplikace obsahovat a jaké údaje v nich budou uloženy. Součásti zadávaných informací je i to, zda se má vyžadovat zadání hodnoty, v jakém rozsahu hodnot se smí zadávané hodnoty pohybovat a případně i vzor (šablona) pro vstup i zobrazování dat.



Obr 1: Aplikační okno nástroje *Plaant Tools* při návrhu GUI

Celý návrh probíhá prostřednictvím vyplňování formulářů, do nichž vývojář zadává požadované hodnoty. Při definici odkazů do jiných agend návrhář pouze zadá odkazovanou agendu ze seznamu. Jedná-li se o odkazy typu M:N pak nemusí definovat žádné pomocné tabulky, protože stačí, když oznámí, že se jedná o násobný odkaz, a o zbytek se postará *Plaant Tools*.

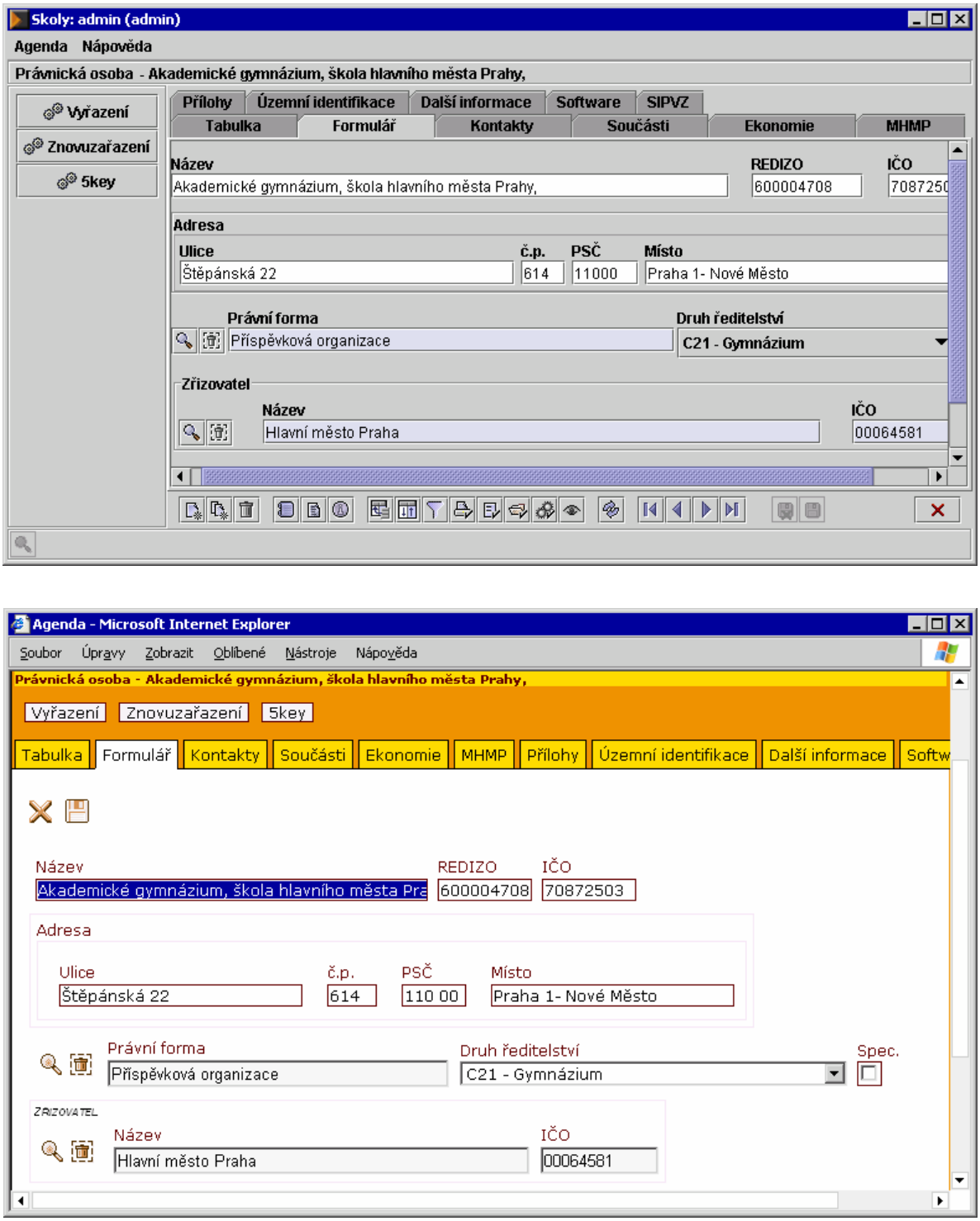
V mnoha případech existují zvyklosti, jak odvodit typické zadávané hodnoty z hodnot dříve zadaných. V takovém případě má návrhář možnost požádat *Plaant Tools*, aby aplikoval toto zvyklostní pravidlo a potřebné hodnoty doplnil.

Součástí definice datových polí je i zadání jejich typických reprezentací v GUI, které usnadňuje následný návrh uživatelského rozhraní.

Při návrhu kostry aplikace definuje vývojář také případného předka popisované agendy, resp. vložení obsahu jiné agendy jako komponenty. Jak jsem již řekl, návrhář se při tomto návrhu nemusí zabývat definicí nějakých pomocných tabulek, protože ve většině případů mu stačí pouze odsouhlasit, že zadání vyhovují ty tabulky, které pro takovéto účely definuje *Plaant* automaticky. Má-li však návrhář nějaké nestandardní požadavky, může zadáním příslušných parametrů tvorbu těchto tabulek samozřejmě ovlivnit.

Po ukončení návrhu *Plaant Tools* vytvoří v databázi příslušnou strukturu tabulek. Do této etapy tvorby aplikace návrhář prakticky nezasahuje. Veškerou starost o vytvoření potřebné struktury na sebe přebírá *Plaant Tools*. Návrhář musí řešit pouze situace, kdy *Plaant Tools* odhalí nějakou nekonzistenci v definici.

Velice příjemnou vlastností nástroje *Plaant Tools* je to, že dokáže nejenom vytvořit novou strukturu „na zelené louce“, ale je schopen při pozdějších modifikacích upravit stávající strukturu tak, aby odpovídala novým požadavkům, a ve většině případů navíc tak, že se při této modifikaci struktury neztratí data.



Obr 2: Porovnání desktopové a webové podoby formuláře

4 NÁVRH GUI

Současně s návrhem struktury dat může vývojář navrhovat i podobu budoucích formulářů. Pro návrh GUI však *Plaant Tools* nenabízí žádný WYSIWYG návrhář. Je tomu tak proto, že žádný současný WISIWYG návrhář nedokáže připravit vzhled formuláře pro několik výrazně odlišných platforem současně (např. pro desktopovou aplikaci, pro webovou aplikaci a pro PDA). Tato potřeba však stála přímo u zrodu systému *Plaant*.

V nástroji *Plaant Tools* se proto nezadává přesná podoba vytvářeného formuláře, ale zadávají se pouze parametry definující sdružování zobrazovaných objektů do větších skupin a

vzájemnou relativní pozici těchto objektů a skupin. Návrhář zadává rámcové uspořádání, které následně dolaďuje úpravou hodnot některých parametrů. *Plaant Tools* mu k tomu poskytuje příslušné formuláře, takže si návrhář nemusí pamatovat záplavy klíčových slov a jim příslušející sady parametrů. Zároveň mu *Plaant Tools* umožňuje průběžně kontrolovat, jak bude navržený formulář vypadat jak v desktopové, tak ve webové verzi GUI (viz obr 1).

Zkušenost ukazuje, že při tomto způsobu návrhu je produktivita práce vyšší, než při využívání WYSIWYG pro samostatný vývoj formulářů pro jednotlivé platformy. V první etapě není úspora ještě tak zřejmé, ale při následných modifikacích (a ty pokaždé zákonitě přijdou) je možnost jednotné definice formulářů pro všechny používané platformy velice výhodná.

4.1 Lokalizace

Součástí základního návrhu je i lokalizace používaných pojmů pro jednotlivé jazykové mutace. Aplikace pak při přihlášení každého klienta připraví svoji lokalizovanou mutaci podle lokality nastavené v klientském systému, případně zadané ve spouštěcím příkazu.

4.2 Výstupy *Plaant Tools*

Na základě informací zadaných ve výše popsaných fázích vytvoří *Plaant Tools* sadu XML souborů, do nichž vloží informace o jednotlivých funkčních segmentech aplikace, odpovídajících datových strukturách a klientském uživatelském rozhraní. Tyto soubory aplikace při svém spouštění načte a podle získaných informací vše potřebné nastaví.

Současně *Plaant Tools* připraví skripty pro použitý RDBMS (Relational Database Management System). Tyto skripty zabezpečí vytvoření potřebných tabulek v databázi, a to jak tabulek s informacemi potřebnými pro vlastní činnost aplikace, tak tabulek pro uložení uživatelských dat.

Nejedná-li se o počáteční návrh, ale modifikaci nějakého staršího návrhu, generuje *Plaant Tools* skripty, které pouze modifikují stávající databázi, aniž by přitom došlo ke ztrátě dat.

5 ZÁVĚR

Používání systému *Plaant* či jemu podobných systémů transformuje vývojářskou práci na zcela novou úroveň. Odpadá spousta „hlavoruční dřiny“, při které je potřeba vkládat do počítače kód, o kterém každý ví, jak má vypadat, avšak neexistují pro něj nějaká obecná pravidla, podle nichž by jej mohly připravit nějaké jednoduché generátory. Tento systém umožňuje relativně jednoduchou a snadnou tvorbu aplikací, které na jednu stranu nabízejí komfort srovnatelný s uživatelsky optimalizovanou aplikacemi typu MS Access, a na druhou stranu umožňují zpracování velkých objemů dat, nasazení v rozsáhlých sítích. Jeho hlavní výhodou je však nejspíš ekonomický přínos, protože vývoj a následná správa aplikace vyvinuté pod systémem *Plaant* je nejenom výrazně rychlejší, ale oproti klasicky vyvíjeným aplikacím vyžaduje zhruba poloviční investice.

LITERATURA

[1] PECINOVSKÝ, Rudolf; LAHODA, Vladimír: Systém pro vývoj distribuovaných aplikací Plaant. *Objekty 2005 – sborník příspěvků devátého ročníku konference*, VŠB, Ostrava 2005. ISBN 80-248-0595-2.

[2] LAHODA, Vladimír; PECINOVSKÝ, Rudolf: Architektura systému Plaant. *Objekty 2005 – sborník příspěvků devátého ročníku konference*, VŠB, Ostrava 2005. ISBN 80-248-0595-2.

[3] Sun, Java 2 Platform, Enterprise Edition (J2EE), *http://java.sun.com/j2ee*.