VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

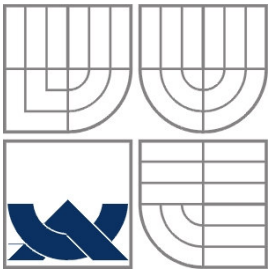
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VÍCEJAZYČNÁ PODPORA V SYSTÉMU OKBASE

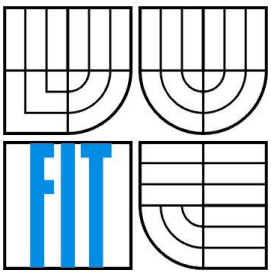
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE LUKÁŠ PODSEDNÍK
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VÍCEJAZYČNÁ PODPORA V SYSTÉMU OKBASE

MULTI-LINGUAL SUPPORT IN OKBASE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — LUKÁŠ PODSEDNÍK
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. JAROSLAV ZENDULKA, CSC.
SUPERVISOR

BRNO 2007

Zadání bakalářské práce/5900/2006/xpodse01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Podsedník Lukáš**
Obor: Informační technologie
Téma: **Vícejazyčná podpora v systému OKbase**
Kategorie: Databáze

Pokyny:

1. Seznamte se se systémem OKbase, včetně frameworku využitého pro implementaci.
2. Seznamte se s požadavky na rozšíření nové verze systému OKbase v oblasti podpory vícejazyčnosti.
3. Seznamte se se způsoby řešení podpory vícejazyčnosti softwarových produktů.
4. Navrhněte koncepci řešení pro systém OKbase.
5. Navržené řešení implementujte a ověřte jeho funkčnost.
6. Zhodnoťte dosažené výsledky a diskutujte další možná vylepšení.

Literatura:

- Programová dokumentace sytému OKbase. OKSystem.
- Spell, B.: Java Programujeme profesionálně. Computer Press, 2002, 1040 s. ISBN 80-7226-667-5.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zendulka Jaroslav, doc. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Lukáš Podsedník**
Id studenta: 84191
Bytem: Dukelská 138/60, 614 00 Brno
Narozen: 19. 03. 1985, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Vícejazyčná podpora v systému OKbase
Vedoucí/školitel VŠKP: Zendulka Jaroslav, doc. Ing., CSc.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

OKbase je kompletní softwarový produkt pro firmy zahrnující docházkový, mzdový a personální modul. Je implementován a provozován v českém jazyce. Cílem tohoto projektu je rozšířit OKbase tak, aby bylo snadné postupně doplňovat podporu dalších jazyků.

Klíčová slova

Vícejazyčnost, lokalizace, OKbase, docházkový systém, mzdový systém, informační systém, vícevrstvá architektura, VVA framework, databáze, Java, Swing, Enterprise Java Beans, klient, server, frontend, backend, číselník

Abstract

OKbase is a complete software product for firms including attendance, salary and human-resources module. It is implemented and conducted in Czech language. The main aim of this project is to extend OKbase to provide simple support of other languages.

Keywords

Multi-lingual support, localization, l10n, internationalization, i18n, OKbase, attendance system, wage-payment system, information system, multi-layer architecture, VVA framework, database, Java, Swing, Enterprise Java Beans, client, server, frontend, backend, code list

Citace

Lukáš Podsedník: Vícejazyčná podporav systému OKbase, bakalářská práce, Brno, FIT VUT v Brně, 2007

Vícejazyčná podpora v systému OKbase

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením doc. Ing. Jaroslava Zendulky, CSc.
Další informace mi poskytli Ing. Michal Dufek, Ing. Martin Kosa, Ing. Karel Miarka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Podsedník
14.5.2007

Poděkování

Rád bych vyjádřil své upřímné díky vedoucímu mé bakalářské práce doc. Ing. Jaroslavu Zendulkovi, CSc., mým nadřízeným Ing. Michalu Dufkovi, Ing. Martinu Kosovi, Ing. Karlu Miarkovi a všem dalším kolegům z firmy OKsystem s.r.o. za skvělou spolupráci, rodině a všem ostatním, kteří mě v průběhu mého studia podporovali a podporují.

Obsah

Úvod

Téměř v každém odvětví lidské činnosti se v současné době používají pro zefektivnění práce informační systémy. Dobře navržený a bezchybně implementovaný informační systém umožňuje snadný, rychlý a přehledný přístup k velkému množství různorodých dat, uživatel tedy může nasměrovat úsilí nikoliv na orientaci v datech, ale na jejich smysluplné využití. Jedním z odvětví oboru informačních systémů jsou firemní softwarové produkty.

Stále více firem na našem trhu navazuje mezinárodní obchodní styky a přijímá zahraniční zaměstnance. Z toho plyne vzrůstající potřeba vývoje firemních informačních systémů s podporou vícejazyčnosti.

Popis systému OKbase

Projekt OKbase je programové vybavení pro komplexní správu lidských zdrojů firmy nebo správní organizace a řízení jejích vnějších vztahů s partnery, zákazníky a institucemi. Je tvořeno funkčně propojenými moduly, které sdílí společnou databázi. Modulární architektura umožňuje transparentní rozšiřování funkcí systému kdykoli po jeho instalaci, v souvislosti s růstem potřeb společnosti a rozšiřováním nabídky modulů.

Projekt OKbase je softwarovým produktem společnosti OKsystem s.r.o.

Moduly projektu OKbase

V současné době jsou podporovány tyto moduly projektu OKbase:

- **Docházkový modul (D)** – nástroj pro evidenci pracovní doby zaměstnanců
- **Personální modul (H)** – evidence a spravování údajů o osobách
- **Mzdový a platový modul (P)** – kompletní zpracování mzdové agendy
- **Systémový modul (S)** – základní součást OKbase, poskytuje funkce pro běh všech modulů

Vícejazyčná podpora v systému OKbase

Produkt OKbase je implementován a provozován v českém jazyce. Je třeba ho rozšířit tak, aby bylo snadné postupně doplňovat podporu dalších jazyků.

Přeloženy by měly být statické texty, což znamená různá hlášení, nadpisy, popisky, položky v menu, ale také dynamické texty, což zahrnuje i data v databázi. Dynamická data (ta, která se mohou s časem měnit) se v klientovi zobrazují především v tzv. Combo Boxech (rozbalovací seznamy) a podobných grafických komponentách.

Systém OKbase a framework využitý při implementaci

1.1 Bližší popis systému OKbase

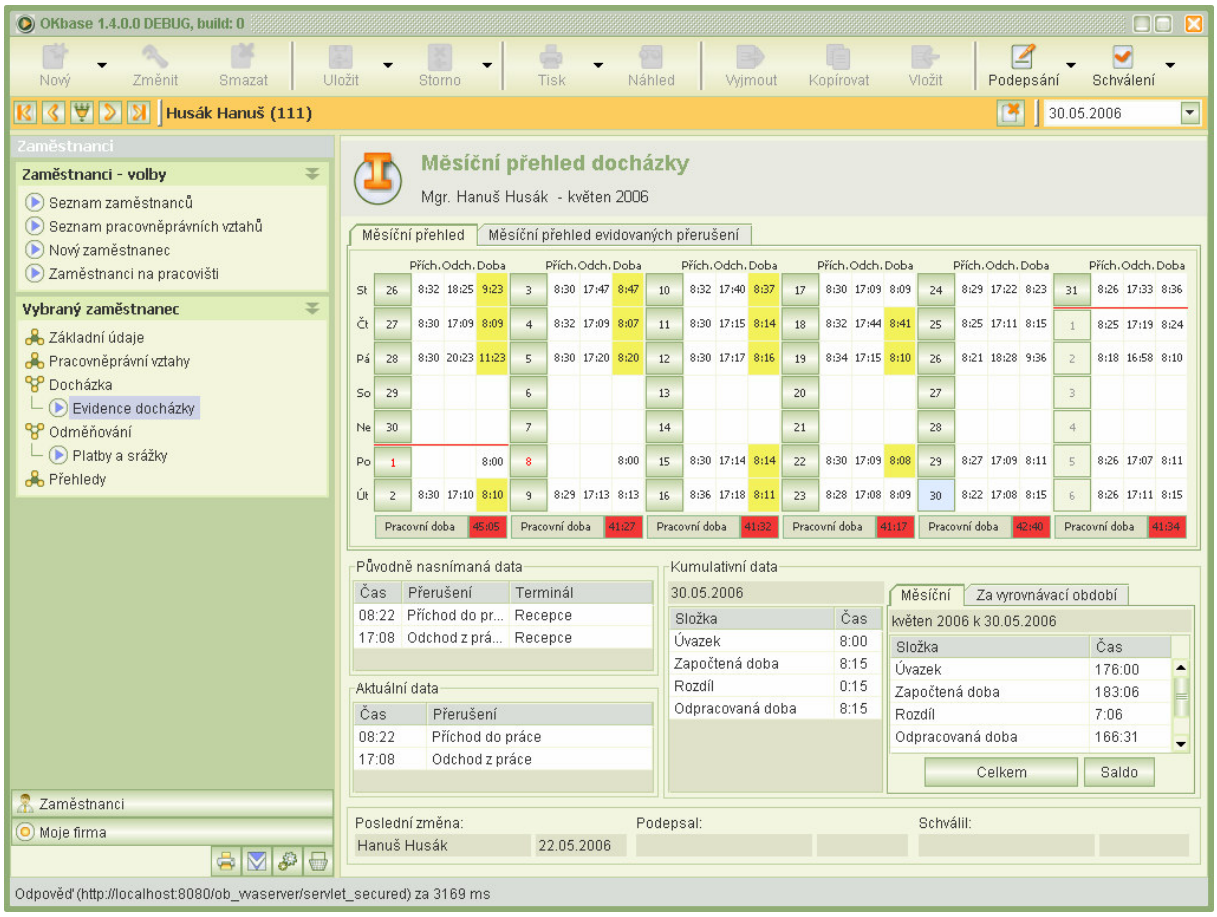
Základní rysy OKbase byly již shrnuty v úvodu. Nyní se budu zabývat bližším popisem OKbase a jeho modulů i z hlediska implementačního.

Ačkoli moduly OKbase sdílí společnou databázi, je systém úzpůsoben tak, aby bylo snadné přidání dalšího funkčního modulu.

OKbase je informační systém s vícevrstvou architekturou (VVA). Požadavky jednotlivých klientů zpracovává aplikační server. Aplikační server spolupracuje s databází OKbase umístěnou na databázovém serveru. Klient tvoří relativně tenkou vrstvu. OKbase podporuje jak webového (tenkého), tak bohatého klienta. Webový klient slouží docházkovému modulu (tzv. docházková samoobsluha). Protože bohatý klient bývá někdy také nazýván „bohatý tenký klient“ (protože v sobě integruje to nejlepší z uživatelské rozhraní moderního programu i komunikačních možnosti webových prohlížečů), abych se vyhnul nejasnostem, nebudu nadále používat pro webového klienta výraz „tenký.“ Bohatý klient má výrazně více funkcí a využívají ho všechny moduly OKbase.

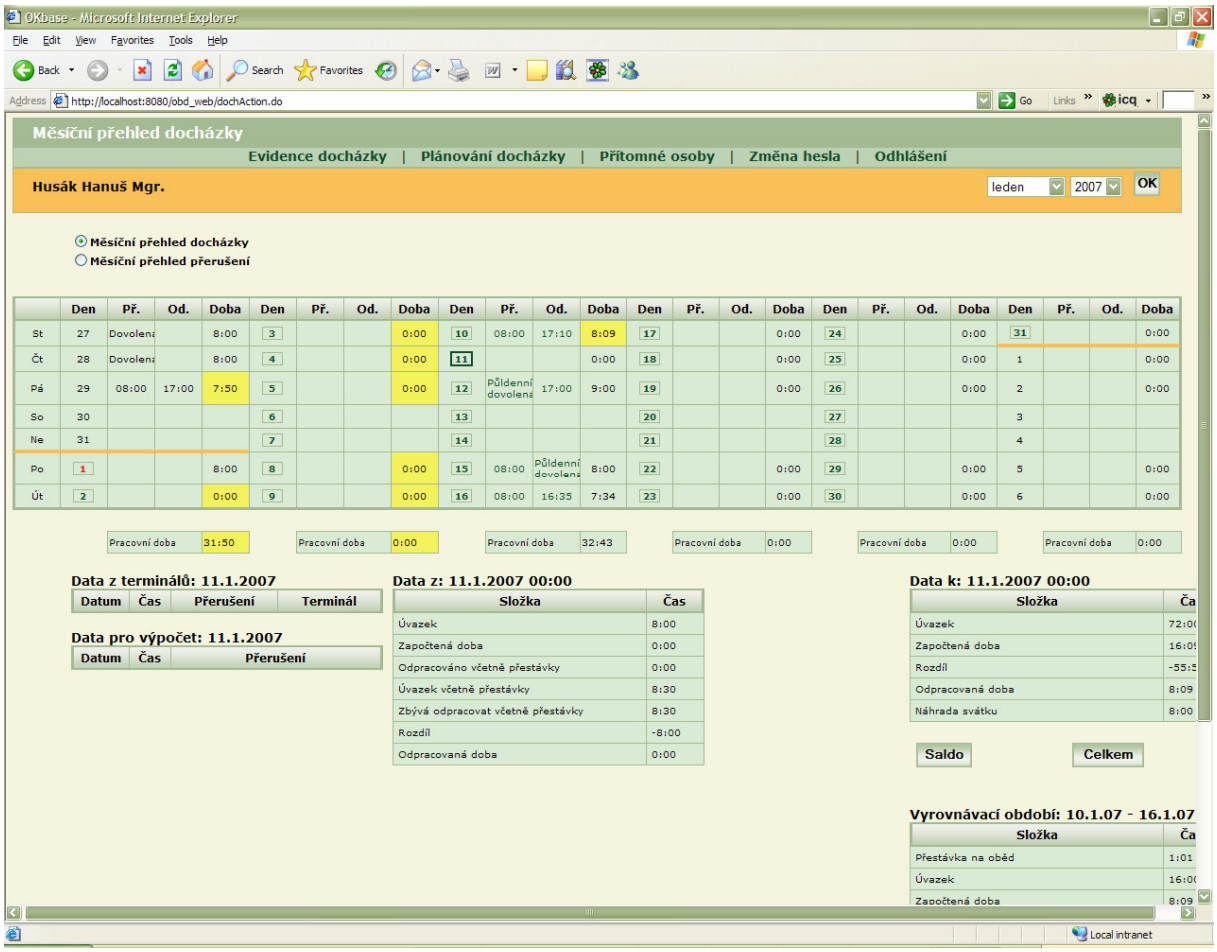
Klient implementuje v podstatě pouze grafické uživatelské rozhraní (GUI). Komponenty, které podléhají nějakým změnám (jsou to obvykle komponenty, které mají přiřazený tzv. Action Listener – zpracovávají události) jsou registrovány na serveru (míněno aplikační server) a server je tak schopen měnit jejich stav a zpracovávat události vyvolané uživatelem.

Na obrázku 1 můžete vidět jednu z mnoha obrazovek bohatého klienta. Jedná se o obrazovku měsíčního přehledu docházky, která je součástí docházkového modulu. Na jednotlivé dny (tlačítka s čísly dnů) lze kliknout a tím dostat grafické zobrazení dne, kde je také umožněno zadat přerušení (např. příchod do práce nebo odchod k lékaři). Tlačítka Podepsat a Schválit využívají nadřízení k podepsání a následnému schválení docházky.



Obrázek 1 - Bohatý klient

Obrázek 2 ukazuje webového klienta – tzv. docházkovou samoobsluhu. Funkcionalitu má podobnou jako měsíční přehled docházky v bohatém klientovi. Navíc umožňuje plánovat některá přerušení (využívá se zejména při plánování dovolené).



Obrázek 2 - Webový klient

Bohatý klient má samozřejmě mnohonásobně více funkcí než klient webový, který slouží pouze jako docházková samoobsluha. Jsou to zejména funkce personální (modul H) a mzdové (modul P). Také umožňuje kompletní konfiguraci OKbase (např. editory číselníků).

1.2 Popis modulů OKbase

Docházkový modul (D)

Docházkový modul je nástroj pro evidenci pracovní doby zaměstnanců. Umožňuje jednoduchým způsobem evidovat příchody a odchody zaměstnanců, následně je vyhodnocovat, kontrolovat a schvalovat. Základem pro implementaci jsou směnové kalendáře zaměstnanců, docházkové terminály a identifikační karty.

Docházková část OKbase eviduje docházku osob a umožňuje průběžné monitorování přítomnosti osob na pracovišti. Data jsou do systému zadávána pomocí docházkových terminálů nebo pomocí běžných PC – užitím webového nebo bohatého klienta. Výpis poskytuje přehled odpracovaných i neodpracovaných hodin v potřebné struktuře.

Každá osoba má po přihlášení přístup k vlastním údajům a má možnost je modifikovat podle nastavených oprávnění. Docházkový modul umožňuje výpočet a schválení salda pracovní doby za celý měsíc – výsledkem jsou vstupní data pro mzdový a platový modul a následný výpočet mezd. Tento modul také obsahuje jednoduchý plánovací kalendář.

Personální modul (H)

Personální modul je nástroj pro optimalizaci využití lidských zdrojů organizace. Umožňuje evidovat a spravovat základní personální údaje o osobách v pracovněprávních vztazích (zaměstnancích nebo osobách ve služebních poměru), uchazečích o zaměstnání i bývalých zaměstnancích.

Dále umožňuje vybudovat systém pro operativní poskytování požadovaných personálních informací o všech zaměstnancích, uchazečích o zaměstnání a osobách udržovaných v evidenci i po odchodu z firmy, evidovat pracovní pozice, pracovní místa a činnosti, poskytovat informace pro tvorbu plánu lidských zdrojů včetně jejich vzdělávání a zefektivnit organizační strukturu firmy, poskytovat historické, evidenční, statistické a dokumentační informace a výstupy, zpřístupnit vybrané údaje všem úrovním vedení firmy a poskytnout manažerské prostředky pro práci s nimi a provozovat jedinou aktuální databázi i pro ostatní systémy vyžadující informace z oblasti lidských zdrojů.

Mzdový a platový modul (P)

Mzdový a platový modul slouží pro kompletní zpracování mzdové agendy. Zaručuje korektní operace a přesné výsledky podle posledních platných předpisů. Je určený pro řešení veškerých úloh v mzdové oblasti - od vedení základních údajů po tvorbu mzdových listů a odvod plateb. Mzdový a platový modul obsahuje údaje o pracovníkovi a jeho pracovním poměru, platové údaje a detaily o odměňování, sledování nepřítomností, zpracování srážek, výpočty daní, sociálního a zdravotního pojištění, výplatní lístky a roční mzdové listy, podklady pro prohlášení poplatníka k dani a výpočtu daně z příjmu, statistické výkazy, měsíční, roční a kumulované výstupy, elektronické podávání ELDP a Přihlášek a odhlášek zaměstnanců k nemocenskému pojištění a další.

Systémový modul (S)

Systémový modul je základní součástí OKbase. Vytváří prostředí a poskytuje funkce pro běh všech modulů. Systémový modul je správcem konfiguračních parametrů, uživatelských práv a funkcí, zabezpečuje i základní funkce související s ochranou osobných údajů a dalších uložených dat.

Základní komponenty jsou správa konfigurace systému, správa číselníků a systémových tabulek, správa uživatelů, oprávnění (přístupová práva uživatelů), události (Event log), zálohování a archivace, audit, tiskový generátor, historie, integrace s OS a další.

1.3 Implementace OKbase

Jak již bylo výše zmíněno, OKbase je naprogramována ve vícevrstvé architektuře, tedy odděluje datovou, aplikační a prezentační vrstvu. VVA klient OKbase je implementován na platformě Java, je tedy možné provozovat ho pod různými OS a architekturami.

Veškerá logika aplikace OKbase (tzv. Business logika) je soustředěna na aplikačním serveru. Business logika OKbase také využívá platformu Java, technologie J2EE (Java 2 Enterprise Edition) a EJB (Enterprise Java Beans). Tyto technologie jsou právě použity v nejspodnější vrstvě aplikace a využívá se jich při styku s databází. Technologie EJB ve spolupráci s Hibernate tools například umožňuje vygenerování entit z databáze a jejich mapování na objekty v jazyku Java. Každá entita v databázi je potom reprezentována Javovskou třídou. EJB také umožňuje elegantní užívaní dotazů a vkladů do databáze. Pro každou entitu si můžeme na jednom místě vytvořit různé předdefinované dotazy, které pak používáme v celé J2EE aplikaci.

Vyšší vrstvy Business logiky zajišťují výpočet. Pro ten se vesměs využívají standardní prostředky jazyka Java, např. pro ukládání dat Java Collections Framework (JCF). Ještě výše jsou potom umístěny prvky komunikující s klientskými (grafickými a datovými) komponentami.

Bohatý klient využívá standardní grafickou knihovnu jazyka Java – Swing. Webový klient využívá technologii JSP (Java Server Pages).

1.4 Implementační VVA framework

Při implementaci je použit speciální VVA framework z dílny firmy OKsystem. V následujících odstavcích se ho pokusím stručně popsat.

VVA framework je knihovna, která usnadňuje práci programátora při vytváření a dotváření klientské a serverové části (míní se část OKbase, která běží na klientovi nebo aplikačním serveru, nikoli tedy databázový server), a to zejména při zajištění komunikace klienta se serverem.

Původně měla být vícejazyčná podpora napsána pro OKbase na novém VVA frameworku 2.0, avšak z časových důvodů bylo přikročeno k implementaci pro OKbase běžícím na VVA frameworku 1.0. (tedy nikoli pro OKbase 2.x, ale pro OKbase 1.x). Vícejazyčná podpora by však měla být co nejméně závislá na implementačním frameworku, aby ji bylo možné využít i pro OKbase 2.x. Dále budu popisovat **VVA framework 1.0**.

Zaměřím se na komunikaci klienta se serverem, což je nejpodstatnější úloha této knihovny. Při spuštění bohatého klienta před sebou vidíme celou řadu komponent. Nejedná se pouze o viditelné, grafické komponenty, ale také o některé datové komponenty, které nevidíme. Ty komponenty, které se za běhu OKbase mohou měnit, nebo ty komponenty, které jsou pro každého uživatele různé, mají také svůj obraz na serveru. Tento obraz je potomkem abstraktní třídy **`AbstractServerComponent`**, která pochází z naší VVA knihovny.

Musí tedy existovat mechanismus, jak na sebe namapovat komponentu z klienta a ze serveru. Tento mechanismus zajišťují třídy s koncovkou –Map – ty obsahují konstanty, které se využívají při registraci komponent.

Chci-li tedy přidat komponentu, která není statická, musím:

Na klientovi:

- přidat instanční proměnnou odpovídajícího typu grafické (nebo datové) komponenty (např. JComboBox)
- provést inicializaci komponenty (např. pomocí konstruktoru – new JComboBox())
- provést registraci komponenty pomocí speciální registrační funkce (právě z VVA frameworku), která má dva parametry: identifikátor komponenty a konstantu z třídy *Map
- případně provést umístění komponenty do okna (pokud se jedná o grafickou komponentu)

Ve třídě *Map:

- doplnit instanční proměnnou – konstantu, která bude sloužit jako identifikátor při registraci komponenty

Na serveru:

- přidat instanční proměnnou do obsluhující třídy, potomka třídy `AbstractServerComponent`
- do konstruktoru předat parametr o identifikaci příslušné komponenty (opět pomocí konstanty z třídy *Map)

2 Požadavky na rozšíření OKbase o vícejazyčnost

2.1 Vícejazyčnost v OKbase

Požadavky na rozšíření projektu OKbase v oblasti podpory vícejazyčnosti již byly shrnuty v kapitole 1.3. Nyní se jimi budu zabývat podrobněji.

Produkt OKbase je implementován a provozován v českém jazyce. Je třeba ho rozšířit tak, aby bylo snadné postupně doplňovat podporu dalších jazyků. Jedná se pouze o vícejazyčnou podporu, nikoli o úplnou lokalizaci. Implementace vícejazyčné podpory by však měla zohledňovat a neměla by bránit možnosti následného zavedení podpory úplné lokalizace, a to zejména podpory veškeré zahraniční legislativy související s personálními, mzdovými, docházkovými a dalšími oblastmi firemních informačních systémů.

Cílem tohoto projektu je zavést vícejazyčnou podporu pro tuzemské uživatele projektu OKbase. Tuzemskými uživateli jsou míněny firmy, které mají zaměstnance v České republice a řídí se zákoníkem práce České republiky. Při zavádění vícejazyčné podpory je nutné postupovat tak, aby bylo snadné rozšířit systém OKbase pro zahraniční uživatele. Zahraničními uživateli jsou míněny firmy, které mají zaměstnance jinde než v České republice a které se řídí zákoníkem práce jiné země než České republiky.

2.2 Co musí být vícejazyčné

Které komponenty by měly být vícejazyčné, to už bylo také zmíněno v kapitole 1.3. Požadavky rozdělím a shrnu do dvou seznamů:

Statické texty:

- nadpisy oken a komponent
- popisky (= nápisy – statické texty v oknech)
- položky v menu
- dialogová okna
- chybová hlášení programu
- texty, které se zobrazují po najetí kurzoru myši nad určitou komponentu

Dynamické texty:

- číselníky (= data, která jsou měnitelná, pokud má na to uživatel dostatečná práva – např. číselník přerušení: sdružuje jednotlivé typy přerušení (a jejich přidružené údaje), jež mohou nastat během dne)
- další data v databázi podléhající překladu

2.3 Další požadavky

Dále je nutné zachovat zpětnou kompatibilitu. Bude tedy potřeba zachovat databázovou strukturu původního OKbase, pouze přidávat tabulky, nebo provádět jen mírné změny.

Dalším argumentem, proč zachovat původní databázovou strukturu je ten, že se k databázi s cizími jazyky bude podle předpokladů přistupovat spíše ve vyjímečných případech. OKbase bude nasazen primárně u českých firem. Vícejazyčnost tedy bude sloužit pro zahraniční pracovníky pracující v českých firmách nebo firmách řídících se českou legislativou.

3 Návrh řešení vícejazyčné podpory

3.1 Statické a dynamické texty, rozdělení

Dříve, než se budeme zabývat návrhem řešení, je třeba si definovat některé pojmy. Vícejazyčnost informačních systémů bývá řešena na dvou úrovních – pro statické a dynamické texty. Pro tyto dvě úrovně se vžil název frontend a backend.

Frontend

Statická data; data, která žádným uživatelem nejdou změnit. Měnit je může pouze programátor (např. pro další verzi systému).

Backend

Data v databázi, která podléhají překladu

Při návrhu vícejazyčnosti frontendu a backendu se nám nabízí celá řada možností. Je možné je je třídit buď z hlediska toho, jakou technologii při návrhu použijeme, nebo z hlediska toho, na jaké vrstvě aplikace budeme vícejazyčnost řešit (tedy jestli na klientovi, v některé servisní vrstvě aplikace, nebo přímo v databázi). V následujících řádcích se budu zabývat tříděním podle druhého hlediska, následně uvedu, jaké technologie (nebo programovací techniky) se při vybrané možnosti dají použít.

Nejdříve ale řešení rozdělím na vícejazyčnost frontendu a backendu, kterými se budu zabývat odděleně.

3.2 Návrh vícejazyčnosti frontendu

U vícejazyčnosti frontendu se nabízí následující možnosti:

- řešení vícejazyčnosti přímo na klientovi
- řešení vícejazyčnosti v aplikační logice (běžící na aplikačním serveru)
- uložení informací do databáze; řešení vícejazyčnosti na úrovni databáze a aplikace

Při umístění na klienta by se mohlo stát, že by klient zabíral neúnosné množství paměti, což je potřeba při návrhu zohlednit a zvážit. Čím více jazyků bude OKbase podporovat, tím tím bude bohatý klient rozsáhlejší. Podle specifikace požadavků bude většinou potřeba pouze český jazyk, proto by mohlo být uchovávání takového množství dat na klientovi ve většině případů zbytečné.

Umístění dat na serveru nebo v databázi by zase zvýšilo komunikační tok mezi klientem a serverem, čímž by zajisté došlo ke zpomalení. V návrhu je proto nutné zvážit, jestli je toto zpomalení podstatné, nebo zanedbatelné. Při uložení do databáze by navíc mohlo dojít ke zbytečnému zvětšení databáze (jedná se o statické texty, které se s časem nemění, proto by v databázi vůbec nemusely být).

Pokud by byla data uložena někde jinde než na klientovi, zachoval by se koncept tzv. „bohatého tenkého klienta“ (zachoval by se co nejmenší rozsah klienta).

3.2.1 Kde jsou uloženy řetězce nyní

Popisky, položky menu a tlačítka jsou v OKbase reprezentovány třídami `JLabel`, `JMenu`, `JMenuItem`, `JButton` (standardní knihovna Swing jazyka Java) a `VvaSpeedButton` (VVA framework OKbase). Zjišťujeme tedy, kde jsou těmto komponentám přiřazovány řetězcové popisky. Pro ilustraci uvádím několik příkladů:

Komponenty `JLabel`

Příklad: `ObmWelcomePanel.java`

```
jLabel_OKsystem_title.setText("Docházkový modul");
jLabel_OKsystem_text.setText(
  "<HTML><BODY>"+
  "Docházkový modul je nástroj pro evidenci pracovní doby zaměstnanců.
Aplikace je svojí variabilitou určená pro všechny druhy organizaci i
firem. Umožňuje jednoduchou a účinnou formou evidovat příchody a odchody
zaměstnanců, následně je vyhodnocovat, kontrolovat a schvalovat. Základem
pro implementaci jsou směnové kalendáře zaměstnanců, docházkové terminály
a identifikační karty."+
  "</BODY></HTML>"
);
jLabel_cile_title.setText("Personální modul");
```

Inicializace textu do prvku `JLabel` probíhá na klientovi.

Komponenty `JMenu` a `JMenuItem`

Příklad: `ObdmMenuPanelMenuBuilder.java`

```
jMenu_obsluha.setText("Obsluha");
jMenuItem_obsluha_prihlaseniUzivatele.setText("Přihlášení uživatele");
jMenuItem_obsluha_vzhledStranky.setText("Vzhled stránky...");
jMenuItem_obsluha_nahled.setText("Náhled...");
jMenuItem_obsluha_importAExport.setText("Import a export...");
jMenuItem_obsluha_konec.setText("Konec");
```

Inicializace textu v menu tedy opět probíhá na klientovi.

Komponenty `JButton` (tlačítka)

Inicializace textu na tlačítkách opět probíha v klientovi.

Komponenty `VvaSpeedButton`

Jedná se o speciální tlačítka z VVA knihovny. Jejich nadpisy se také inicializují na klientovi.

Závěr: Inicializace většiny komponent se statickými texty (i těch, které jsou přes VVA knihovnu propojené se serverem) se odehrává na klientovi. To může být argument, proč umístit vícejazyčnost frontendu právě na klienta.

3.2.2 Vícejazyčnost na klientovi

Pro vícejazyčnost implementovanou přímo na klientovi je vhodné využít standardní prostředky Javy – třídy týkající se lokalizace (zkratka l10n) a implementace mezinárodní podpory (anglicky internationalization – zkratka i18n) z balíku `java.util.*` (především potomky tříd `java.util.ResourceBundle`).

Statické texty na tlačítkách, v menu a na popiskách by se potom neinicializovaly přímou hodnotou, jak je tomu u klasické aplikace využívající knihovnu `javax.swing`. Získání lokalizovaných řetězců demonstrují následující úseky kódu převzaté z Literatury [1]. Java nabízí dvě možnosti, jak ukládat vícejazyčné texty:

```
import java.util.*;

protected class MyResources extends ListResourceBundle {
  protected static Object[][] resources = {
    {"WhatIsJava", "Jaka je Java?"}
    // …
  };

  public Object[][] getContents() {
    return resources;
  }
}
```

Nyní jsme vytvořili třídu, kde je uložen požadovaný řetězec. Třída `MyResources` je potomkem třídy `ListResourceBundle`. Metoda `getContents()` vrací dvojrozměrné pole párů klíč/prostředek. Tři tečky v příkladu naznačují, že pole „resources“ obvykle obsahuje více párů klíč/prostředek. Rovněž může existovat více tříd-potomků `ListResourceBundle`, např. `MyResources_de` pro německou jazykovou mutaci.

Následující kód ukazuje, jakým způsobem z `ListResourceBundle` hodnotu lokalizovaného řetězce získáme:

```
import java.util.*;

public class JavaQuestion {
  protected static ResourceBundle resources =
    ResourceBundle.getBundle("MyResources");

  public static void main(String[] args) {
    System.out.println(resources.getString("WhatIsJava"));
  }
}
```

Podotýkám, že program je značně zjednodušen, aby jasně demonstroval použití internacionalizace. Třída `ListResourceBundle` (její potomek) slouží jako úložiště lokalizovaných dat a hlavní funkce třídy `JavaQuestion` vypíše na standardní výstup lokalizovaný řetězec.

To, v jakém jazyce řetězec bude, určuje nastavené prostředí v operačním systému. To v OKbase nebude to pravé. Budeme potřebovat dostat ty řetězce, které souhlasí s národním prostředím definovaným v uživatelském profilu uživatele OKbase.

Následující úsek kódu objasňuje, jak si lze vynutit, kterou jazykovou mutaci vícejazyčného textu získáme.

```
protected static ResourceBundle resources =
  ResourceBundle.getBundle("MyResources", new Locale("de", "CH"));
```

Java nabízí ještě jednodušší způsob, jak ukládat lokalizované řetězce. Jedná se o třídu `PropertyResourceBundle`, která je také potomkem `ResourceBundle`. U `PropertyResourceBundle` nemusíme vytvářet novou třídu pro uložení vícejazyčných textů – stačí vytvořit stejnojmenný soubor s příponou .properties, do něhož vložíme požadované informace:

```
WhatIsJava=Was ist Java?
```

Soubor .properties je textový, obsahuje řádky klíč=hodnota oddělené znakem pro konec řádku. Tento zápis má dvě výhody:

- je výrazně jednodušší
- při přidání další „property“ se nemusí třída s řetězci překládat

Výhody tohoto řešení:

- data jsou uložena na klientovi, není tedy třeba měnit dosavadní koncepci uložení. Tyto klientské komponenty nebude třeba registrovat na serveru pomocí VVA knihovny a nebude třeba zatěžovat spojení posíláním dat z klienta.

Způsob uložení statických dat (frontendu) na klientovi má ale své nevýhody:

- čím více bude jazyků, tím větší množství paměti bude klient zabírat
- do jisté míry odporuje koncepci „tenkého“ bohatého klienta popsaného v kapitole 1.1
- při používání webového klienta je uložení jakýchkoli dat na klientovi nerealizovatelné. Bude tedy nutné zabývat se vícejazyčností bohatého a webového klienta zvlášť.

3.2.3 Vícejazyčnost na aplikačním serveru

Je nutné se také zabývat řešením, které bude více koncepčně příbuzné dosavadní implementaci systému OKbase a nebude tolik zatěžovat klienta.

Nic nebrání tomu, použít prostředky jazyka Java z balíku `java.util.*` i části OKbase umístěné na aplikačním serveru. Mechanismus popsaný v předchozí kapitole to dovoluje.

Některé statické komponenty (jedná se především o komponenty `JLabel`) nejsou pomocí VVA knihovny na serveru zaregistrovány (pomocí funkce `registerComponent()`). V případě tohoto řešení bude tedy nutné pracovat na úkolech:

- vytvoření `ListResourceBundle` (java tříd) nebo `PropertyResourceBundle` (tzv. .properties souborů) v části OKbase umístěné na aplikačním serveru
- registrace všech komponent (bude nutné upravit jak bohatého klienta, tak server) pomocí funkcí VVA knihovny
- smazat inicializaci komponent na klientovi a umístit inicializaci komponent na server

Pro inicializaci komponent je vhodné vytvořit speciální servisní třídu, která bude zajišťovat služby vícejazyčnosti.

Výhody tohoto řešení:

- poměrně snadná implementace, nemusí se měnit databázové schéma
- nedojde ke zvýšení zatížení klienta

Nevýhody:

- data nejsou uložena v databázi, ale v textových souborech (nemusí být nevýhoda, jedná se o statická data, která se nebudou měnit)
- zvýšení komunikačního toku mezi klientem a serverem

Podotýkám, že ke zvýšení přenášeného objemu dat mezi klientem a serverem dojde každopádně, pokud budeme preferovat řešení, které nebude založeno na uložení výcejazyčných dat

přímo na klientovi. Bude k němu ale docházet především jednorázově při spouštění klienta pomocí Java WebStart, nemusíme se tedy obávat výrazného zpomalení běhu aplikace.

3.2.4 Vícejazyčnost frontendu uložená v databázi

Nabízí se řešení, které uloží statická vícejazyčná data přímo do databáze. Je diskutabilní, jestli je nutné ukládat vícejazyčná data právě do databáze, zvláště když se jedná o statická data, která by se s časem neměla měnit.

Pozn.: Data by se neměla rozhodně neměla měnit u uživatelů (nebude existovat žádný editor dat), ke změnám může docházet při přidávání podpory dalších jazyků do OKbase nebo při modifikaci těch stávajících. Všechno toto by měli zajišťovat programátoři manuálně.

Při ukládání dat do databáze nelze použít nástoje pro vícejazyčnost z balíku `java.util.*`. Bylo by nutné napsat vlastní nástroje. Dále navrhněme, jak by takováto data mohla být uložena v databázi:

- pro každý jazyk nová tabulka (sloupce klíč, hodnota) – složitější na implementaci, vybírání tabulky podle id jazyka by se muselo řešit dynamickým tvořením SQL dotazů v Java kódu, případně dynamickým SQL
- jedna velká tabulka se sloupci (klíč, hodnota, id_jazyka) – je možné vybrat jazyk

Dále se nabízí dvě možnosti, jak vícejazyčná data použít:

1. Klient si při spuštění a připojení k serveru vyžádá jazykovou mutaci, ta bude načtena z databáze a bude mu poslána jako pole klíč/hodnota. Klient si pak sám inicializuje komponenty se statickými texty (nebude tedy docházet k registraci těchto komponent na serveru).
2. Klient si při spuštění a připojení k serveru vyžádá jazykovou mutaci, ta bude načtena z databáze a podle ní budou incializovány obrazy klientských statických komponent na serveru. Tyto komponenty bude tedy nutné před inicializací vícejazyčným textem registrovat a přidat na server (jak bylo popsáno v kapitole 1.4).

Výhody možnosti 1.:

- neregistrované komponenty klienta nebude třeba registrovat na serveru pomocí VVA knihovny

Nevýhody možnosti 1.:

- pomalejší spuštění klienta, posílání velkého objemu dat na začátek, ikdyž třeba nebude potřeba
- dojde k dalšímu zatížení klienta

- příliš se neshoduje s koncepcí OKbase postavené na využívání VVA knihovny při komunikaci klienta a serveru

Poslední nevýhoda by se dala vyřešit registrací „komponenty vícejazyčnosti“ – pole (nebo kolekci) párů klíč/hodnota.

Výhody možnosti 2.:

- nedojde k přílišnému zatížení klienta

Nevýhody možnosti 2.:

- nutnost registrovat všechny grafické komponenty, i ty se statickými texty

3.2.5 Závěr

Hlavním kritériem při rozhodnutí, kam vícejazyčnost umístit tedy bude fakt, kolik paměti by zabíraly vícejazyčné texty na klientovi. Provedeme tedy experiment a zkušební .properties soubor vytvoříme. Vynásobením jeho velikosti počtem jazyků získáme navýšení paměti, kterou bude klient zabírat.

Jelikož klient používá technologii Java WebStart, dochází k automatickým aktualizacím. Properties soubory se budou měnit, pokud nebudou aktuální.

Pro ukládání dat do .properties souborů byla původně stanovena tato pravidla pro pojmenování identifikátorů (klíčů) property:

- identifikátory budou bez háčků a čárek
- první písmeno identifikátoru bude velké
- u víceslovných názvů bude každé počáteční písmeno dalšího slova velké
- jinak budou identifikátory stejné jako české texty
- pokud má český název více než 5 slov, dojde ke zkrácení – další slova se neuvedou
- pokud se jedná o delší text (např. text úvodní text popisu jednotlivých modulů OKbase), dojde k výstižnému pojmenování identifikátoru (nemusí souhlasit s textem). V takovém případě je nutné si identifikátor textově vyhledat v .properties souboru podle hodnoty (řetězce), kterou reprezentuje
- pokud je v českém názvu dvojtečka („:“), přijde místo ní do identifikátoru „Dvojtecka“
- taktéž „Tecka“, extra mezera =„Mezera“, tři tečky = „Trojtecka“

Během vývoje se však ukázalo, že vývojové prostředí Eclipse umožňuje částečně automatizované generování .properties souborů, jež značně usnadní práci. Nevýhodou ovšem je, že identifikátory (klíče) property nesplňují výše stanovená pravidla.

Zkušební .properties soubor pro docházkový modul zaujímá 5,57KB. Vzhledem k tomu, že soubor bude v JAR archivu, bude jeho velikost po kompresi 2,29KB. Pokud by například OKbase podporoval 10 jazyků, bylo by to 10 properties souborů o velikosti 22,9KB. Když k tomu přidáme

.properties soubory dalších modulů OKbase, několikanásobně se velikost zvýší. Avšak ani to nestačí k tomu, abychom museli pro extrémní velikost možnost uložení na klientovi zavrhnout.

Nejvýhodnější možnost uložení vícejazyčných dat frontendu pro bohatého klienta je tedy přímo na klientovi. Webový klient bude mít svá vícejazyčná data uložena v .properties souborech na serveru.

3.3 Návrh vícejazyčnosti backendu

Data k vícejazyčnosti backendu budou určitě uložena do databáze, protože sama dynamická data jsou tamtéž uložena.

Při řešení je ale nutné splnit požadavek zpětné kompatibility. Znamená to, že původní databázové schéma musí být zachováno, do databáze se smí pouze přidávat tabulky (nebo sloupce tabulek). Z tohoto požadavku plyne hodně omezení, která se projeví při návrhu řešení.

Například z docházkového modulu se bude muset vyřešit vícejazyčnost u těchto tabulek:

D_C_CHYBA, D_C_PRERUSENI, D_C_PRERUSENI_R, D_C_ZADOST_STATUS, D_HIST_KUMULAT, D_TERM_PARAM, D_TERMINAL_KALENDAR.

Ve mzdovém modulu jich bude mnohonásobně více.

Při vícejazyčnosti backendu je potřeba vyřešit tyto úkoly:

- návrh databázového schématu pro uložení vícejazyčných dat
- návrh celku aplikace, který bude tato data z databáze načítat a poskytovat jednoduché rozhraní pro vyšší vrstvy aplikace

V následujících kapitolách popíši jednotlivé možnosti uložení vícejazyčných dynamických dat do databáze.

3.3.1 Řešení více tabulkami

Demonstrujme řešení na konkrétní tabulce *D_C_PRERUSENI*. Tato tabulka obsahuje různé typy přerušení, které si zaměstnanec v docházkovém systému může zadat. Např. příchod do práce, odchod z práce, odchod na přestávku, odchod k lékaři, dovolená, nemoc, atd.

Pokusím se stručně vysvětlit některé sloupce tabulky *D_C_PRERUSENI*:

kod – primární klíč, jednoznačný identifikátor přerušení

text – text přerušení, např. „Odchod k lékaři“

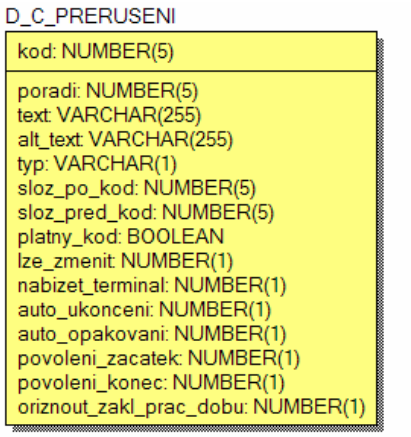
alt_text – text přerušení, který se zobrazuje na docházkovém terminálu

typ – např. P, O, nebo C – příchodové, odchodové, nebo celodenní

sloz_po_kod – mzdová složka, která následuje po tomto přerušení (např. „Odpracovaná doba“ po příchodu do práce)

sloz_pred_kod – mzdová složka, která předchází toto přerušení (např. „Odpracovaná doba“ před odchodem z práce)

Další sloupce jsou příznaky používané při výpočtu docházky. Obrázek 3 ukazuje tabulku *D_C_PRERUSENI* zobrazenou pomocí ER diagramu.



Obrázek 3

Sloupce, u kterých se bude řešit vícejazyčnost:

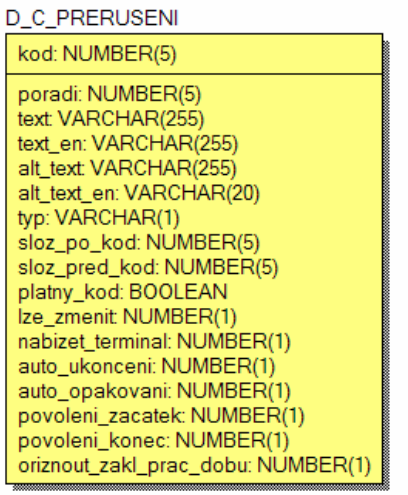
- *text*
- *alt_text*

U sloupce *typ* vícejazyčnost řešit nebudeme, protože je používán především interně (uvnitř aplikace, navenek se projevuje pouze v důsledku).

Nyní se zabývejme metodou, jak vícejazyčná data uložit.

Pomiňme některá řešení s redundancí dat nebo jinými nevýhodami:

- duplicita sloupců – pouhé přidání atributů do stejné tabulky, např. pro angličtinu *text_en* a *alt_text_en*. Nevýhoda tohoto způsobu je především v tom, že při přidání dalšího jazyka se musí měnit databázová struktura. Výsledné schéma můžete vidět na následujícím obrázku. Má své opodstatnění pro vícejazyčnost, kdy přesně víme, že hodláme přidat jenom jeden nebo dva jazyky. Výhodou je především jednoduchost, kvůli které se toto řešení často používá. Pro nás ale není vhodné.



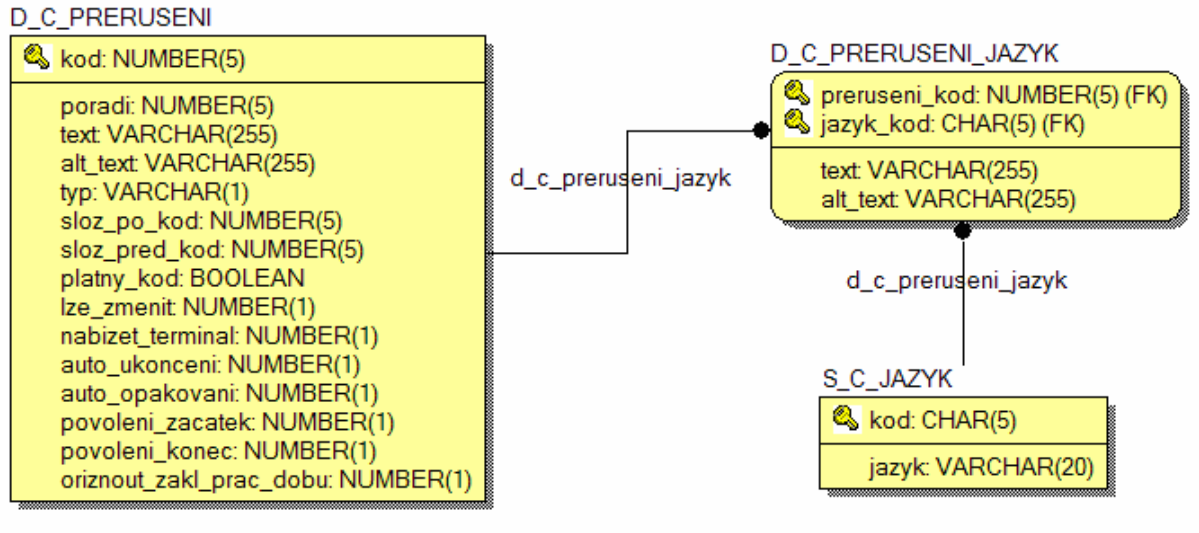
Obrázek 4

- duplicita řádků – pro každý jazyk je zduplikován řádek tabulky. Klíč je složený z původního primárního klíče, je ještě přidán klíč jazyka, který nazveme *lang_id*. Nevýhoda tohoto způsobu nastane, pokud jsou v tabulce atributy, které překladu nepodléhají. Např. u *D_C_PRERUSENI* je jich celá řada: *sloz_po_kod*, *sloz_pred_kod*, *platny_kod*, *lze_zmenit*, *nabizet_terminal*, a další. Dojde pak k nežádoucí duplicitě a redundanci dat – data budou v tabulce tolikrát, kolik bude jazyků (místo jedenkrát).

Při návrhu se tedy musíme vyvarovat:

- toho, aby bylo nutné při přidání jazyka měnit databázové schéma
- redundance

Ke splnění požadavků musíme tedy tabulku *D_C_PRERUSENI* rozdělit na část podléhající překladu a část nepodléhající překladu. Všechny atributy, které potřebujeme překládat, osamostatníme do další tabulky, která bude k základní tabulce ve vztahu 1:N, kde N bude počet jazyků. Řešení by mohlo vypadat takto:



Obrázek 5

Identifikátor jazyka v tabulce *S_C_JAZYK* nebude dvojznakový (např. „cs“, „en“, „de“ pro češtinu, angličtinu, němčinu), ale pětiznakový typu „cs_CZ“ nebo „en_US“. Příslušnost ke státu (národu) v OKbase sice nebudeme uvažovat, ale zkratky pětimístného typu jsou obvyklejší a používanější (zejména v Jazyku Java).

V tabulce *D_C_PRERUSENI* zůstaly překládané sloupce *text* a *alt_text*. Vyplývá to z požadavku z kapitoly 2.3 (Další požadavky, Specifikace požadavků), kdy je nutné zachovat zpětnou kompatibilitu, tedy není možné v databázi nějaké sloupce mazat. Vzhledem k tomu, že čeština jako výchozí jazyk v tabulce S_*C_JAZYK* ani *D_C_PRERUSENI_JAZYK* nebude, nedojde k žádné redundanci.

Tatáž transformace se provede se všemi tabulkami v databázi, které obsahují texty podléhající překladu. V kapitole 3.3 jsou vyjmenovány tabulky docházkového modulu, které takové texty obsahují. Je jich 7. Jenom v docházkovém modulu by tedy vzniklo navíc 7 dalších tabulek a 7 by se muselo modifikovat. Vezmeme-li v úvahu celý systém OKbase, vznikly by navíc desítky databázových tabulek.

Ačkoli je toto řešení pravděpodobně návrhově nejčistší, vznikají zde problémy s přílišnou složitostí a neúměrně velkým počtem nových tabulek, které vícejazyčná podpora do systému OKbase přináší.

Shrňme tedy do bodů výhody a nevýhody tohoto řešení:

Výhody:

- návrhová čistota

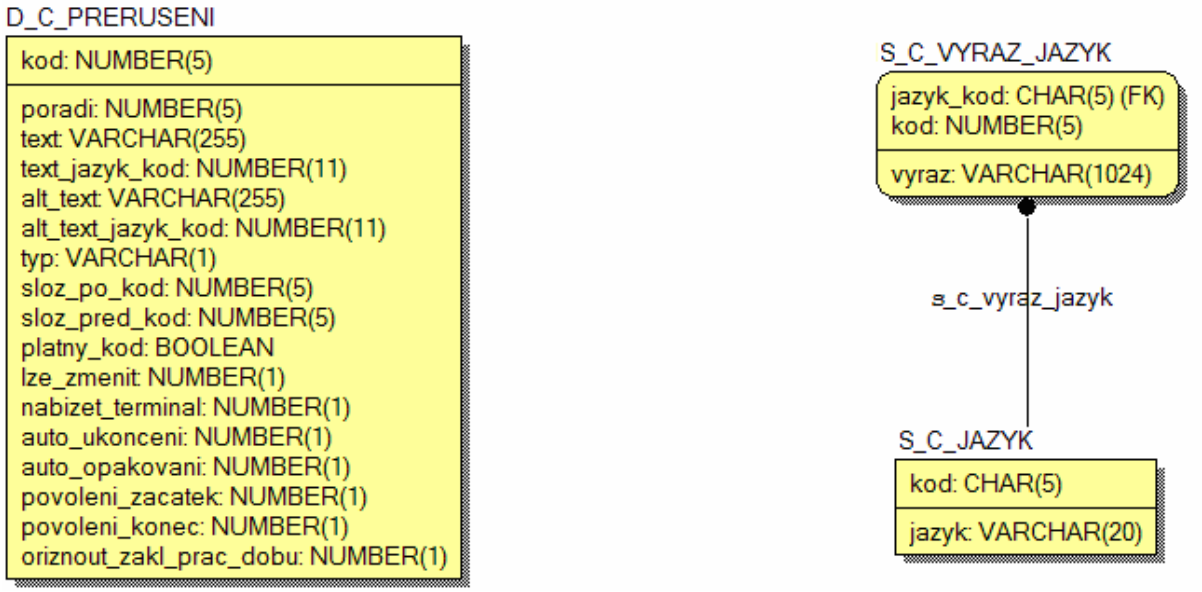
Nevýhody:

- velké množství tabulek

Předcházející řešení vícejazyčnosti backendu byla převzata z Literatury [3].

3.3.2 Řešení jednou tabulkou

Data backendu je též možné uložit do jedné tabulky pro celý OKbase, případně do jedné tabulky v rámci modulu OKbase. Tato tabulka bude mít sloupce *jazyk_kod*, *kod*, *vyraz*. Sloupec *vyraz* bude hledaný textový řetězec patřící ke sloupci určité tabulky. Vícejazyčné sloupce budou zdvojené, přičemž další sloupec bude obsahovat část primárního klíče do tabulky *S_C_VYRAZ_JAZYK*.



Obrázek 6

Výhody:

- poměrně jednoduché, do databáze se přidají pro každý modul jenom dvě tabulky

Nevýhody:

- mezi tabulkami *D_C_PRERUSENI* a *S_C_VYRAZ_JAZYK* není možné udělat vztah, záznamy v tabulkách se tedy musí mazat, upravovat a přidávat programově
- není příliš vhodné kvůli jazykové podpoře do tabulky *D_C_PRERUSENI* přidávat sloupce

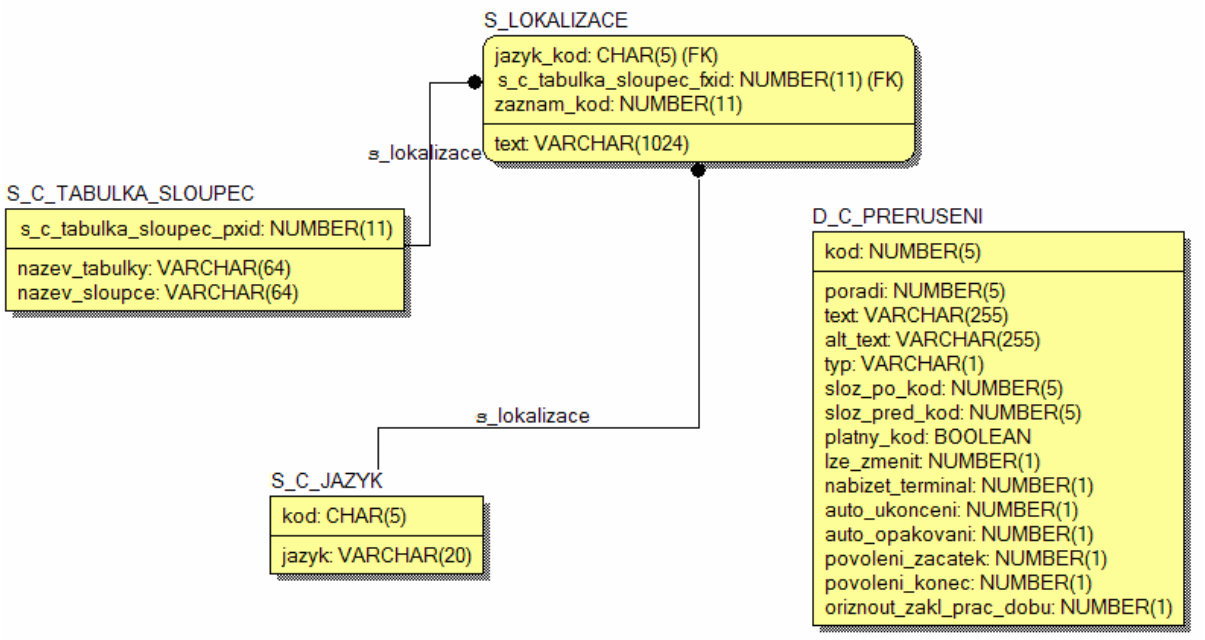
3.3.3 Dynamické řešení

Je možné i jiné řešení, které nazveme dynamické. Dynamické proto, že umožňuje nejenom přidat jazyk bez změny databázové struktury, ale také přidat nějaký atribut, sloupec jakékoli tabulky, který se dříve nepřekládal a nyní je ho potřeba přeložit.

Nevýhodou je ale opět to, že s překládanou tabulkou (v našem případě *D_C_PRERUSENI*) není možné vytvořit vztah.

Na první pohled sice vypadá, že vztah může vzniknout. Je ale potřeba si uvědomit, že na atribut *zaznam_kod* by musely být navázány všechny tabulky, u kterých je potřeba provést lokalizaci, nejenom tabulka *D_C_PRERUSENI*. Jeden cizí klíč ale může být navázán pouze na jednu tabulku.

Obrázek 7 ukazuje výsledné schéma.



Obrázek 7

Výhody tohoto řešení:

- je možné dynamicky přidat sloupec některé databázové tabulky, který se dříve nepřekládal a nyní je potřeba ho přeložit
- do databáze se nepřidává mnoho tabulek
- požadovaná data je možné získat dvěma SQL příkazy

Nevýhody tohoto řešení:

- mezi tabulkami *D_C_PRERUSENI* a *S_LOKALIZACE* není možné vytvořit vztah, záznamy v tabulkách se tedy musí mazat, upravovat a přidávat programově, nikoli na úrovni databáze

Z hlediska použitých technologií je možné takovéto řešení implementovat. Je možné vytvořit servisní vrstvu mezi databází obsluhovanou EJB (a Hibernate) a aplikační logikou.

Pro implementaci bylo vybráno právě poslední, dynamické, řešení.

3.4 Možnost přepínání mezi jazyky

Zadání přesně nespecifikuje, jakým způsobem bude moci uživatel mezi jazyky přepínat. Možnosti jsou následující:

- uživatel nebude moci přepínat mezi jazyky, uživatelův jazyk se zjistí z uživatelského nastavení na jeho lokálním počítači
- uživatel bude moci dynamicky přepínat mezi jazyky. Musí tedy existovat mechanismus, který by si pamatoval, který jazyk patří ke kterému uživateli.

Druhé řešení je logičtější. OKbase může běžet v některé firmě s pracovníky, kteří mluví různými jazyky. Více pracovníků může ale přistupovat k jednomu stroji na jeden uživatelský účet operačního systému. Proto uživatelská nastavení lokálního uživatele nemusí být vždy správným vodítkem pro to, jaký jazyk vybrat. Nehledě na to, že při práci s webovým klientem k informacím o lokálním uživateli vůbec nemusí být přístup. Přepínání by se tedy mělo uskutečňovat na úrovni OKbase, ne pouze podle toho, jaký má lokální uživatelský profil výchozí jazyk.

Z toho vyplývá nutnost zavést v OKbase uživatelské profily. Ty v OKbase už jsou implementovány. Stačí je pouze využívat a přidat k nim položku, která udává, který jazyk má uživatel vybrán.

Také bude samozřejmě potřeba vytvořit položku (v menu, v konfiguraci OKbase...), pomocí které bude uživatel jazyk přepínat. Při přepnutí se uživatelské nastavení uloží do uživatelského profilu.

4 Implementace navrženého řešení

V této kapitole se budu věnovat nejprve obecným krokům, které jsem musel podniknout při implementaci vícejazyčnosti. Dále se budu opět ve zvláštních kapitolách věnovat vícejazyčnosti frontendu a backendu aplikace.

4.1 Implementace vícejazyčnosti obecně

Nejprve musíme udělat základní kroky k tomu, aby program podporoval více jazyků. To zahrnuje:

1. uložení jazyka do uživatelského profilu
2. načítání profilu při přihlašování, aplikování uživatelských nastavení
3. uložení informace o aktuálním jazyku uživatele jak na klientovi, tak na serveru, zajištění komunikace mezi klientem a serverem tak, aby se promítly změny v lokalizovaných textech při výběru jiného jazyka

4.1.1 Uložení jazyka do uživatelského profilu

V OKbase jsou uživatelské profily (alespoň na databázové úrovni) navrženy a implementovány. K ukládání dat nám poslouží následující tabulky:

S_UZIV_PROFIL – obsahuje parametry uživatelského profilu. Má sloupce:

- primární klíč – složený z následujících sloupců *PARAM_KOD* a *S_UZIV_FXID*
- *PARAM_KOD* – cizí klíč do tabulky *S_C_PROFIL_PARAM*
- *S_UZIV_FXID* – cizí klíč do tabulky *S_UZIVATEL*
- *PARAM_HODNOTA* – hodnota parametru profilu (řetězec)

S_C_PROFIL_PARAM – abychom mohli uložit do tabulky *S_UZIV_PROFIL* nějaký parametr, musíme ho nejdříve definovat v této tabulce. Tato tabulka má sloupce *KOD* (primární klíč, id záznamu), *S_PRGMODUL_FXID* (id modulu, kterého se parametr uživatelského profilu týká) a *PORADI* (pořadí, v jakém se parametr případně bude zobrazovat v editoru systémových tabulek). *S_PRGMODUL_FXID* je tedy cizím klíčem do tabulky *S_PROG_MODUL*.

Vícejazyčnost bude implementovat systémový modul OKbase, který má v tabulce *S_PROG_MODUL id*=1. Vložíme tedy nejdříve řádek do tabulky *S_C_PROFIL_PARAM*:

```
insert into S_C_PROFIL_PARAM values ('1', 1, 'jazyk', 1);
```

Výchozí nastavení uživatele bude vždy na český jazyk. Proto se do tabulky *S_UZIV_PROFIL* záznam o češtině nemusí ukládat. Abychom mohli vícejazyčnost demonstrovat, zvolíme pro jednoho uživatele jiný jazyk následujícím způsobem:

```
insert into S_UZIV_PROFIL values (100804, '1', 'en_US');
```

Z tabulky *S_UZIVATEL* jsme vybrali uživatele s id=100804, zadali jsme id parametru jazyka a tento parametr jsme nastavili na hodnotu ‘en_US’, což je angličtina v prostředí spojených států.

4.1.2 Načítání parametru uživatelského profilu z databáze

OKbase využívá technologii EJB, která (jak už název napovídá) používá ke čtení dat z DB tzv. beany. Jsou to třídy servisní vrstvy (v OKbase uložené zpravidla v projektech `obX_service`), které poskytují metody, jež načtou data z DB (případně je namapují na jiné objekty), uloží data do DB (z jiných objektů) a další funkce pracující s DB, které potřebujeme.

Tyto třídy mohou být buď „statefull“ nebo „stateless“, tedy stavové nebo bezestavové. Stavové beans kromě poskytování metod vyšším vrstvám umožňují také uchovat v sobě nějaký stav. Tyto

třídy se označují pomocí anotací „`@Statefull`“ nebo „`@Stateless`“. V OKbase používáme pouze bezestavové beans.

Pro získání jazyka z DB tedy vytvoříme bezetavovou bean. Pro tu musíme napsat nejdříve rozhraní (interface), které pojmenujeme `ObcLokalizaceManager`. Následně vytvoříme bean, která toto rozhraní implementuje – `ObcLokalizaceManagerBean`.

Tuto bean musíme umístit tak, abychom přidali co nejméně závislostí do projektů, z kterých je OKbase složena – do projektu `obs_service` (služby systémového modulu), balíčku `cz.oksystem.okbase.okbases.service.lokalizace`.

Rozhraní `ObcLokalizaceManager` obsahuje metodu `setLokalizace()`, která získá záznam o jazyku z databáze a zařídí nastavení a uchování tohoto údaje v paměti. Více tuto problematiku popíši v kapitole 4.1.4.

4.1.3 Načítání parametru profilu při přihlašování

Pro načtení parametru profilu při přihlašování už stačí jenom na správném místě zavolat metodu `setLokalizace()` rozhraní `ObcLokalizaceManager`. Tímto místem je metoda `prihlas()` ve třídě `AutentizaceManagerBean` (pro bohatého klienta) a metoda `doPost()` ve třídě `LoginServlet` (u webového klienta).

4.1.4 Uložení informace o aktuálním jazyku v paměti

Aktuální jazyk budeme nejspíše ukládat do paměti ve formě nějakého objektu.

Řešíme následující otázky:

- co bude obsahovat
- kam ho umístit, aby přibylo co nejméně závislostí mezi projekty
- jak ho uložit v paměti tak, aby byl odevšad dostupný, ale aby nebyl statický – na serveru přece nemůže být pouze jeden objekt informace o jazyku, když k němu může být zaráz připojeno více klientů.

Pojmenujme třídu tohoto objektu `ObcLokalizace`. V této kapitole nás především zajímá, že bude obsahovat informaci o nastaveném jazyku. Ta se v Javě ukládá pomocí třídy `Locale`. Stačí tedy jednoduše vzít řetězec získaný z databáze a předat ho jako parametr konstruktoru objektu třídy `Locale`. Co bude `ObcLokalizace` dále obsahovat blíže popíši v kapitole 4.2 o implementaci vícejazyčnosti frontendu.

Nejlepší umístění pro takový objekt bude takové, odkud bude „všude vidět“. Takové místo bude nepochybně projekt `ob_lib`, což je knihovna napsaná speciálně pro OKbase, která se využívá ve většině projektů OKbase, v kterých budeme vícejazyčnost potřebovat.

Největším problémem bude uložení `ObcLokalizace` v paměti. Začneme tím, že uložíme objekt `ObcLokalizace` pro všechny vlákna (pro všechny připojené klienty) aplikačního serveru pouze jeden. Vytvoříme jakýsi statický objekt. Java vytvoření statické třídy nepodporuje (vytvoření třídy, která by byla celá statická), umožňuje ale ve třídě vytvářet statické metody a proměnné. Nicméně způsobem, jak vytvořit třídu, která by byla „celá statická“ je návrhový vzor Singleton. Použití singletonu demonstruji následujícím příkladem převzatým z Literatury [4]:

```
public class ClassicSingleton {
   private static ClassicSingleton instance = null;
   protected ClassicSingleton() {
      // Exists only to defeat instantiation.
   }
   public static ClassicSingleton getInstance() {
      if(instance == null) {
         instance = new ClassicSingleton();
      }
      return instance;
   }
}
```

Třída v sobě obsahuje statickou instanci sama sebe. Obsahuje také privátní (nebo protected) konstruktor a veřejnou metodu `getInstance()`. Ta vrátí instanci třídy, která je však statická, je tedy pro všechny volající (ať jsou kdekoli), stejná. Při prvním zavolání této metody se vytvoří instance, která existuje po celý běh programu.

Vytvoříme tedy Singleton třídu `ObcLokalizace`. Nyní ovšem stojíme před problémem, jak přetvořit třídu `ObcLokalizace` tak, aby byla statická, ale pouze pro jednoho připojeného klienta (mluvíme pořád o objektu `ObcLokalizace` umístěném na aplikačním serveru).

Při vytváření nové relace mezi klientem a serverem se používají vlákna – v Javě implementované třídou `Thread`. Zjednodušeně řečeno se při připojení dalšího klienta na serveru vytvoří další vlákno. Musíme tedy dotvořit třídu `ObcLokalizace` tak, aby byla „statická“, ale přitom lokální pro každé vlákno. K tomu nám poslouží Javovská třída **`ThreadLocal`**. Tato třída slouží právě k vytvoření proměnné, která je lokální pouze pro jedinné vlákno. Přikládám opět ukázku použití – předchozí příklad jsem doplnil na Singleton, který bude lokální pro každé vlákno:

```
public class ClassicSingleton {
   private static ThreadLocal<ClassicSingleton> instance =
                                       new ThreadLocal<ClassicSingleton>();
   protected ClassicSingleton() {
      // Exists only to defeat instantiation.
   }
   public static ClassicSingleton getInstance() {
      if(instance.get() == null) {
         instance.set(new ClassicSingleton());
      }
      return instance.get();
   }
}
```

Poměrně snadným způsobem jsme dospěli k požadovanému výsledku. Privátní instance bude místo typu `ClassicSingleton` typu `ThreadLocal<ClassicSingleton>` (od Javy 5 jsou

podporovány generické typy, při volání metod `get()` a `set()` se tedy nemusí přetypovávat a výsledný kód vypadá velmi přehledně a elegantně). Na objekt třídy `ThreadLocal` potom můžeme volat metody `get()` a `set()` pro získání a nastavení lokální proměnné pro naše vlákno. `ThreadLocal` rozpozná automaticky, o které vlákno se jedná.

Podobně tedy aplikujeme tento poznatek na třídu `ObcLokalizace`.

4.1.5 Komunikace mezi aplikačním serverem a klientem

Po úspěšném načtení uživatelského nastavení jazyka a jeho uložení pomocí `ObcLokalizace` na serveru je potřeba informaci poslat na klienta a uložit ji i tam.

Vzhledem k tomu, že třída `ObcLokalizace` je Singleton, není vhodné jí přes síť posílat. Ze serveru na klienta bychom měli posílat jenom opravdu nezbytné informace. Nezbytnou informací není nic jiného, než informace o nastavení jazyka. Po přihlášení tedy pošleme z klienta na server instanci třídy `Locale`, která je nestatickým prvkem třídy `ObcLokalizace`.

Posílání budeme realizovat přes VVA framework, na kterém je OKbase postavena. Při stisknutí tlačítka OK při přihlášení se vyvolá akce. Ta zavolá mimojiné metodu `prihlas()` z `AutentizaceManagerBean`, o které již byla řeč v kapitole 4.1.3. Hned po ní zavoláme metodu z VVA knihovny:

```
this.posliAkci(new TransportAction(ObsLoginController
  .getDefaultRootComponent().getFullName(), NameProperty.VALUE,
  ObcLokalizace.getInstance().getLocale()));
```

Toto volání se odehrává ve třídě `ObsLoginController`, což je třída na serveru, která tvoří obraz třídy `ObsLoginPanel` na klientovi. Do `ObsLoginPanel` musíme přidat listener, který bude registrovat poslaný objekt ze serveru. Provedeme to následujícím způsobem:

```
this.addReceiveProperty(NameProperty.VALUE,
      new AbstractReceiveProperty() {
          public void receive(TransportAction transportAction) {
            Object o = transportAction.getValue();
            if(o instanceof Locale) {
                  Locale locale = (Locale) o;
                  ObcLokalizace.setLocale(locale);
                }
          }
        });
```

Použijeme metodu `addReceiveProperty()`, kterou má každý panel na klientovi (je totiž přímým potomkem třídy `ObcDataPanel`. Jako parametr dáme této metodě:

- typ hodnoty, na který čekáme

- `AbstractReceiveProperty`, což je třída, která implemetuje reakci na přijatou událost

V reakci na událost jednoduše použijeme metodu objektu `ObcLokalizace` a i na klientovi (opět pro jedno vlákno klienta, může být tedy spuštěných více klientů na jednom stroji) nastavíme klientskou instanci `ObcLokalizace` na správnou hodnotu.

4.2 Implementace vícejazyčnosti frontendu

Jak bylo popsáno v návrhu, používáme standardní knihovny jazyka Java. Podporu vícejazyčnosti zabudujeme přímo do `ObcLokalizace`.

`ObcLokalizace` bude obsahovat také zdrojová data (resources) pro lokalizaci. Tato zdrojová data se načítají z .properties souborů, pro každý modul OKbase je zvláštní zdrojový soubor a zvláštní zdrojová data. Při změně `Locale` se načtou zdrojová data pro jiný jazyk. K takové změně dochází jednak při přihlášení uživatele (instance `ObcLokalizace` se vytváří nejdříve implicitně s `Locale`='cs_CZ', tedy českým, až potom se nastaví Locale na hodnotu danou v uživatelském profilu), a jednak při změně nastavení uživatele za běhu programu.

ObcLokalizace potom obsahuje metodu `getObXString(klíč)`, která podle klíče vrátí požadovaný řetězec (písmeno X nahradí písmeno příslušného modulu OKbase).

Jako příklad použití uvádím fragment kódu z `ObdCPreruseniModifikacePanel`, což je kód pro dialogové okno, které přidává nové přerušení do číselníku přerušení (*D_C_PRERUSENI*):

```
public class ObdCPreruseniModifikacePanel extends ObcDataPanel {

  private static final long serialVersionUID = 1L;
  private ObcLokalizace obcLokalizace = ObcLokalizace.getInstance();

// ...

  private void jbInit() throws Exception {
  // ...
    vvaCheckBox_autoUkonceni
      .setText(obcLokalizace.getObdString("ObdCPreruseniModifikacePanel.5"));
    vvaCheckBox_autoOpakovani
      .setText(obcLokalizace.getObdString("ObdCPreruseniModifikacePanel.6"));

vvaCheckBox_nabizet.setText(obcLokalizace.getObdString("ObdCPreruseniModifikacePanel.7"));
    vvaCheckBox_oriznout
      .setText(obcLokalizace.getObdString("ObdCPreruseniModifikacePanel.8"));
  // ...
  }
  // ...
}
```

Podobnou úpravu musíme provést se všemi třídami, které obsahují řetězce podléhající překladu. To je poměrně „mravenčí“ práce. Vývojové prostředí Eclipse naštěstí umožňuje přepisování javovských tříd a generování .properties souborů částečně zautomatizovat. Nevýhoda této automatizace ale spočívá v tom, že klíče k řetězcům jsou poměrně nesrozumitelné (např. `ObdCPreruseniModifikacePanel.5` nám toho moc o hodnotě, která se pod ním skrývá, neřekne). Další nevýhodou je, že generátor neumí rozpoznávat shodné řetězce. V properties souboru totiž musí být unikátní pouze klíč, nikoli hodnota (řetězec). V properties souboru se tedy může vyskytovat neomezené množství shodných řetězců s různými klíči (největší počet si lze např. představit u řetězce „OK“, „Zrušit“ apod.).

Tyto nevýhody ale nejsou tolik kritické, jak by se mohly zdát. Za prvé, klíč nám určitou informaci o řetězci poskytuje (alespoň z jaké je třídy). A za druhé, textové properties soubory zaujímají velmi malý prostor na to, abychom museli řešit nějaké optimalizace, nehledě na to, že řetězce jako „OK“ se sice v properties souboru mouhou vyskytnout dvacetkrát, ale je třeba přihlédnout také k tomu, že „OK“ je dvacetkrát kratší než mnohé řetězce, které se vyskytují v properties souboru pouze jednou.

4.3 Implementace vícejazyčnosti backendu

4.3.1 Změna databázového schématu

Nejdříve musíme změnit databázové schéma, jak bylo popsáno v kapitole 3.3.3. Vytvoříme nejdříve tabulku *S_C_JAZYK* pro uložení jazyků:

```
create table S_C_JAZYK (
  kod     CHAR(5) DEFAULT 'cs_CZ' NOT NULL,
  jazyk   VARCHAR2(50) NOT NULL
);
alter table S_C_JAZYK
  add ( constraint XPKS_C_JAZYK primary key (kod) );
```

Dále vytvoříme tabulku *S_C_TABULKA_SLOUPEC*, která bude uchovávat informaci o tom, které sloupce z kterých tabulek budou podléhat překladu, a tabulku *S_LOKALIZACE*, která bude uchovávat překládané výrazy:

```
create table S_C_TABULKA_SLOUPEC (
  s_c_tabulka_sloupec_pxid  NUMBER(11) default 0 not null,
  nazev_tabulky             VARCHAR(64) not null,
  nazev_sloupce             VARCHAR(64) not null
);
```

```
alter table S_C_TABULKA_SLOUPEC
  add ( constraint XPKS_C_TABULKA_SLOUPEC primary key
                                          (s_c_tabulka_sloupec_pxid) );
create table S_LOKALIZACE (
  jazyk_kod                 CHAR(5) not null,
  s_c_tabulka_sloupec_fxid  NUMBER(11) not null,
  zaznam_kod                NUMBER(11) not null,
  text                      VARCHAR2(1024)
);
alter table S_LOKALIZACE
  add (constraint XPKS_LOKALIZACE primary key
                 (jazyk_kod, s_c_tabulka_sloupec_fxid, zaznam_kod) );
```

Nyní je potřeba vytvořit vazby mezi tabulkami pomoci cizích klíčů:

```
alter table S_LOKALIZACE
  add ( constraint s_c_jazyk foreign key (jazyk_kod) references S_C_JAZYK,
        constraint s_c_tabulka_sloupec foreign key
              (s_c_tabulka_sloupec_fxid) references S_C_TABULKA_SLOUPEC );
```

4.3.2 Změna načítání číselníků z databáze

Po té, co jsme si nachystali databázové tabulky, musíme do nich vložit inicializační data. Například pro tabulku *D_C_PRERUSENI* to bude:

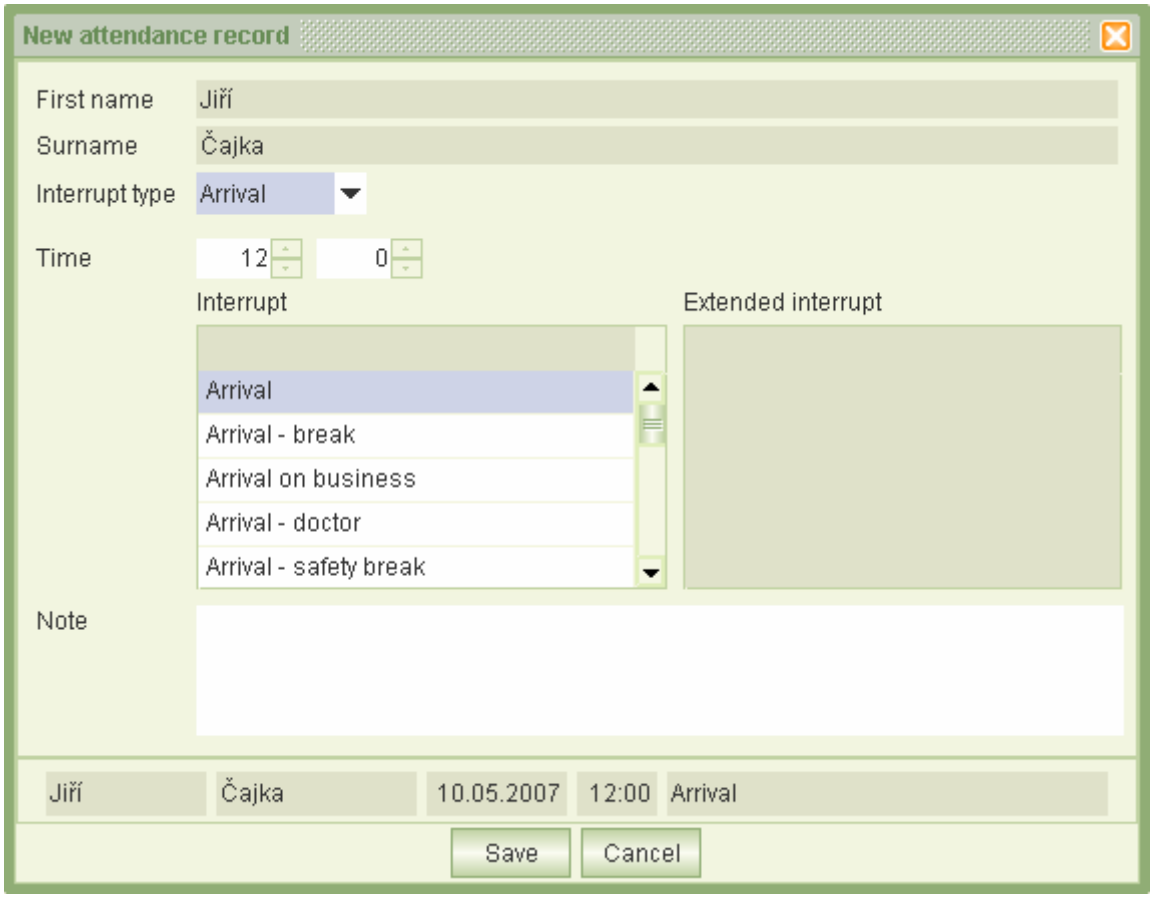
```
insert into S_C_TABULKA_SLOUPEC values ('1', 'D_C_PRERUSENI', 'TEXT');
insert into S_C_TABULKA_SLOUPEC values ('2', 'D_C_PRERUSENI', 'ALT_TEXT');
insert into S_C_JAZYK values ('en_US', 'English (United states)');

insert into S_LOKALIZACE values ('en_US', '1', '1', 'Arrival');
insert into S_LOKALIZACE values ('en_US', '2', '1', 'Work');
insert into S_LOKALIZACE values ('en_US', '1', '2', 'Arrival - doctor');
insert into S_LOKALIZACE values ('en_US', '2', '2', `Doctor');
insert into S_LOKALIZACE values ('en_US', '1', '7', 'Arrival - break');
insert into S_LOKALIZACE values ('en_US', '2', '7', 'Break');
insert into S_LOKALIZACE values ('en_US', '1', '51', 'Arrival - safety
break');
insert into S_LOKALIZACE values ('en_US', '2', '51', 'Safety break');
insert into S_LOKALIZACE values ('en_US', '1', '3', 'Arrival on
business');
insert into S_LOKALIZACE values ('en_US', '2', '3', 'Business journey');
-- ...
```

Dále je potřeba zajistit, aby aplikace, zaregistruje-li dotaz na tabulku, která obsahuje vícejazyčná data, nahradila vícejazyčné sloupce správnou jazykovou mutací. Tyto úkoly zajistí následující algoritmus, který popisuje činnost metody `getLocalizedQuerry()` z rozhraní `ObcLokalizaceManager`, která má jako vstupní parametry seznam objektů (jako výsledek dotazu do DB) a identifikátor typu objektu (třída `Class`):

- Zjisti nastavení jazyka přihlášeného uživatele.
- Pokud se jedná o češtinu, vrať výsledek. Jinak pokračuj.
- Načti z tabulky *D_C_TABULKA_SLOUPEC* záznamy pro tabulku. Její jméno získej z identifikátoru třídy (`Class`).
- Iteruj přes záznamy, iteruj přes atributy třídy a testuj jejich identifikátory na vzájemnou rovnost. Pokud jsou si rovny, ulož jejich identifikátor do seznamu.
- Načti z tabulky *S_LOKALIZACE* data potřebná k lokalizaci.
- Iteruj přes vstupní parametr výsledek a na základě výše získaných informací přepisuj odpovídající sloupce.
- vrať výsledek.

Metoda `getLocalizedQuery()` se tedy zavolá vždy, když bude potřeba zkontrolovat, zda není potřeba data výsledku dotazu upravit. Typicky to bude při čtení tabulek číselníků.

Jako ukázku řešení vícejazyčnosti jsem si vybral formulář nového záznamu docházky (Obrázek 8). Jsou v něm vidět jak statické, tak dynamické texty.



Obrázek 8

V ComboBoxu a v seznamu vidíme vícejazyčná data backendu z DB, ostatní vícejazyčná data (nadpis okna, popisky, tlačítka) jsou statická – frontend.

5 Závěr

Systém OKbase má nyní připravenou vícejazyčnou podporu. Tuto podporu nejvíce využijí manažeři, vedoucí pracovníci a zaměstnanci společností na našem trhu, kteří neovládají český jazyk. Poměrně elegantním způsobem se podařilo systém OKbase zobecnit; řetězce nejsou nyní uloženy přímo v kódu, ale ve zvláštních .properties souborech. Odpovídá to dnešním tendencím oddělovat kód (záležitost programátora) od zobrazení a dat (záležidost designéra a analytika).

Za zmínku také stojí, že při přidávání jazyka se nebude muset zasahovat do zdrojového kódu OKbase. Nebude se do něj dokonce muset zasahovat ani tehdy, chceme-li přidat sloupec tabulky podléhající překladu. Tento fakt bude hrát významnou roli při údržbě OKbase.

Podařilo se vytvořit základní podporu vícejazyčnosti. Vývoj bude samozřejmě pokračovat dále. Dalším vylepšením by mohlo být vytvoření vícejazyčného editoru systémových tabulek podléhajících překladu. Uživatelé by pak mohli při přidání záznamu do systémové tabulky přidávat také záznamy do tabulky *S_LOKALIZACE* a tím pomocí bohatého klienta doplňovat překlady nových položek číselníků.

Dále je také vhodné zkoumat možnost uložení podpory vícejazyčnosti backendu ještě níže. V OKbase existuje třída `OkbaseReverseEngineeringStrategy`, která říká, jakým způsobem se mají mapovat databázové tabulky na entity (Java třídy) a řádky tabulek na objekty. Ukládá, jak se mají atributy tříd jmenovat a jak se budou jmenovat jejich nastavovací metody (getters, setters). Ve třídě implementující rozhraní `ReverseEngineeringStrategy` (což je v OKbase právě `OkbaseReverseEngineeringStrategy`) je také možné navázat spojení s DB a pracovat s ní. Z toho plyne určitá možnost ovlivnit načítání číselníků z DB.

Cílem tohoto projektu nebylo OKbase přeložit, ale vytvořit podporu pro cizí jazyky. Je proto potřeba nyní zadat práci profesionálním překladatelům, aby naplnili .properties soubory a inicializační data v DB. Pro překladatele je možné vytvořit jednoduchou aplikaci, která jim bude plnění .properties souborů a vytváření inicializačních dat usnadňovat.

Jak bylo zmíněno v kapitole 3.2.5, vývojové prostředí Eclipse umožňuje částečně utomatizované generování .properties souborů. Je také možné vyvinout vlastní plug-in do vývojového prostředí Eclipse, který by toto generování plně zautomatizoval, a který by vytvářel identifikátory (klíče) v .properties souborech podle pravidel uvedených v kapitole 3.2.5.

Literatura

[1] Programová dokumentace sytému OKbase, OKsystem s.r.o.

[2] Spell, B.: Java Programujeme profesionálně. Computer Press, 2002, ISBN 80-7226-667-5

[3] Zelenka, P.: Vícejazyčné webové aplikace a relační databáze, 2005, článek: http://interval.cz/clanky/vicejazycne-webove-aplikace-a-relacni-databaze/

[4] Geary, D.: Simply Singleton, 2003, článek: http://www.javaworld.com/javaworld/jw-04-2003/jw-0425-designpatterns.html

Seznam příloh

Příloha 1. CD – zdrojový text tříd obsluhujících vícejazyčnost

Příloha 2. CD – okbase.ear – spustitelná verze

Příloha 3. CD – zkušební databáze, na kterou se lze napojit