PLANTLORE

Vývojová dokumentace

Univerzita Karlova
Matematicko-fyzikální fakulta
Praha, Česká republika

Vedoucí: RNDr. Antonín Říha, CSc.

Lada Oberreiterová ladaob@seznam.cz

Jakub Kotowski fraktalek@gmx.net

Tomáš Kovařík tkovarik@gmail.com

Erik Kratochvíl krater@seznam.cz

Obsah

I. Projekt Plantlore

Řešitelé

Lada Oberreiterová

Erik Kratochvíl

Tomáš Kovařík

Jakub Kotowski

Rozdělení úkolů

Lada Oberreiterová

- Webová část aplikace využívající BioCASE Provider Software.
- Přizpůsobení aplikace pro naše potřeby, komunikace s vývojáři BioCASE.
- Testování a opravy aplikace pro využití s databázemi PostgreSQL a Firebird.
- Namapování Plantlore databáze na standardy ABCD Schema 1.2, ABCD Schema 2.06, Darwin Core.
- Klient Plantlore: Správa metadat, Správa uživatelů, Celková historie, Historie záznamu, Převody mezi souřadnými soustavami. Implementace různých formátů pro export. Uživatelská a programátorská dokumentace těchto částí.
- Návrh grafického vzhledu webové části.
- Příprava a úpravy databázového schematu, udržování databáze.

Erik Kratochvíl

- Implementace RMI mechanizmu pro komunikaci klienta a serveru.
- Plantlore klient: Kostra pro Import a Export nálezů, načítání a zpracování XML dokumentů.
- Serverová a komunikační část aplikace.
- Spolupráce na implementaci vrstvy DBLayer.
- Práce na společných částech Swing GUI – ProgressBary, inteligentní ComboBoxy.
- Proces přihlašování k databázi a odhlašování.

Tomáš Kovařík

- Vrstva DBLayer pro přístup k databázi, rozhraní pro zadávání dotazů, použití ORM systému Hibernate.
- Plantlore klient: Správa autorů, Správa publikací, Vytváření nové databáze, Systém pro zobrazování interaktivní nápovědy.
- Instalátor (lokalizace, instalace PostgreSQL, vytváření odkazů).
- Stránky projektu na plantlore.berlios.de resp. plantlore.sourceforge.net
- Příprava databázového schematu.

Jakub kotowski

- Plantlore klient: Hlavní okno aplikace (tzv. overview se seznam nálezů, zobrazování stromu nálezů), hlavní MVC komponenta, tzv. AppCore která vytváří jednotlivé dialogy, Dialog pro přidávání a editaci nálezů, vyhledávání nálezů, zobrazování a tisk schedy a seznamu nálezů. Nastavení, načítání konfiguračních souborů.
- Instalátor (základní kostra – výběr balíčků, jednotlivé informační panely).
- Práce na společných částech Swing GUI – ProgressBary.
- Stránky projektu na plantlore.berlios.de resp. plantlore.sourceforge.net
- Správa systému pro správu verzí (SVN).

Společné úkoly

- Komunikace se zadavateli – Botaniky Jihočeské pobočky ČBS
- Hledání informací o jiných podobných projektech, botanických programech používaných v České republice.

Důvody zpoždění projektu

Komunikace s budoucími uživateli byla složitá a neefektivní, zejména zpočátku. Požadavky byly špatně interpretovány jednou nebo druhou stranou. Přispěla k tomu i neznalost botanické problematiky a aplikací běžných v této oblasti – nebylo nám vždy jasné zamýšlené použití.
Zásadní problém byl i nedostatek zkušeností s používanými technologiemi – přetrvávající nebo dokonce opakující se problémy s RMI, Hibernate, databázovými systémy, Swingem.

II. Zadání projektu a popis řešeného problému

Oficální zadání projektu

Cílem projektu je vytvořit databázový systém pro zadávání a vyhledávání dat o výskytu rostlin. Databáze by měla odpovídat mezinárodním standardům taxonomických databází (za účelem importu dat z mezinárodních databází). Aplikace bude umožňovat uživatelsky jednoduché přidávání dat (na základě už existujících seznamů jmen rostlin) a vyhledávání podle různých kritérií (čas, místo, okolnosti nálezu atp.). Dále bude možné zadávat a vyhledávat data na základe souřadnic z GPS a dle mezinárodních souřadnicových systémů, mezi nimiž bude aplikace schopna dělat převody (S42 na WGS84, SJTSK na WGS84). Součástí systému bude také možnost exportu dat do různých formátů.

Aplikace bude postavena na architektuře klient-server a bude tak umožňovat práci s daty většímu množství botaniků. K datům bude možné přistupovat také prostřednictvím sítě Internet. Celý systém bude vytvořen a bude využívat jenom volně šiřitelné technologie a bude distribuován pod licencí GNU GPL. Program samotný bude implementován v jazyce Java s cílem maximální platformní nezávislosti.

Systém bude instalován v jihočeské pobočce ČBS příp. v Jihočeském muzeu a na katedře botaniky BF JČU. Naším cílem je však vytvořit systém, který by mohli používat botanici v celé ČR (resp. po celém světě), protože v současnosti většinou používají vlastní (často zastaralé) systémy, které neumožňují výměnu a sdílení nálezových dat a jsou nedostačující pro potřeby širší botanické obce. Právě proto chceme, aby aplikace splňovala mezinárodní standardy a vyhovovala požadavkům co největšího množství botaniků.

Identifikované požadavky

Obecné požadavky

- Architektura klient-server
  - Webový klient – umožní pouze prohlížení dat
  - GUI klient – umožní tyto operace:
    - Zadávání dat
    - Úprava dat
    - Možnost připojit se na server přes síť

- Systém práv
  - Registrovaní uživatelé mají přístup ke všem datům (přes web)
  - Veřejně je přístupná pouze určitá podmnožina dat (přes web)
  - Funguje to podobně jako na http://botanika.bf.jcu.cz/jpcbs
  - Součástí nálezového záznamu je autor a zdroj

- Podpora vícejazyčnosti
  - Čeština
  - Angličtina
  - Aplikace je připravena na jednoduché přidání dalších jazyků

Funkční požadavky

- Zadávání dat
  - Akceptovaná jména rostlin a jejich synonyma
  - Povinné položky - Taxon, Nálezce, Rok nálezu, Lokalita, Nejbližší větší sídlo
  - Minimální nutný počet kroků - automatické doplňování zadaných slov, omezení výběru na relevantní možnosti podle již zadaných údajů
  - Možnost zadání dat z GPS

- Export dat
  - XML – různé formáty
  - Text (CSV)

- Import
  - Transakční zpracování (import velkého množství dat)
  - Import XML dat

- Vyhledávání
  - Musí obsahovat vyhledávání
    - Podle názvu taxonu
    - Podle synonyma nebo akceptovaného jména
    - Podle nejbližšího většího sídla
  - Vyhledávání rozděleno na jednoduché a rozšířené

- Zobrazení dat
  - Jednoduché tabulkové (základní položky)
  - Podrobné tabulkové (všechny položky)
  - Herbářové položky

- Nápověda
  - Tooltipy – určitě k důležitým položkám GUI
  - Kontextová nápověda - GUI klient (jednotlivé části, dialogy)

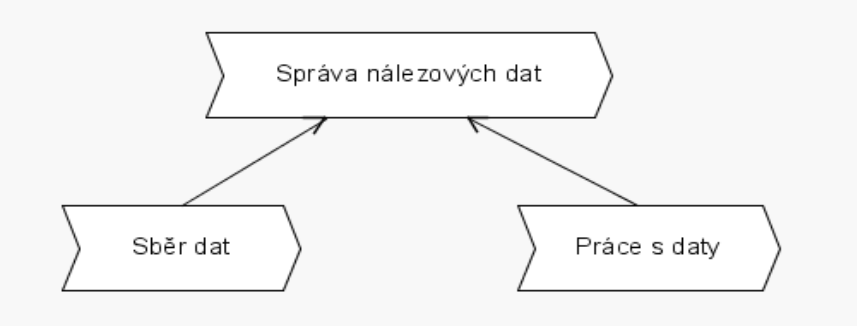
- Editace dat - viz zadávání dat

- Transformace souřadnicových systémů
  - S42 na WGS84
  - SJTSK na WGS84

- Tisk - výsledek vyhledávání je možné vytisknout v herbářové i tabelární podobě

Analýza procesů

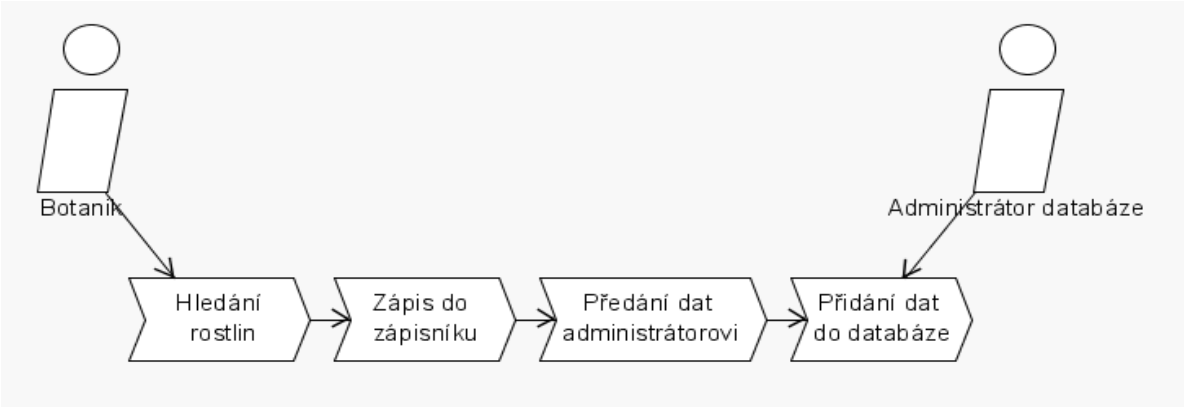
Následujíci část obsahuje popis základních procesů při sběru, zaznamenávání a zpracování nálezových dat. Jsou zobrazeny procesy probíhající dnes při použití existující databáze na platformě MS Access a to, jak to bude vypadat při použití systému Plantlore.



Správa nálezových dat obecně pozůstáva ze sběru dat a další práce s daty (zpracování dat).

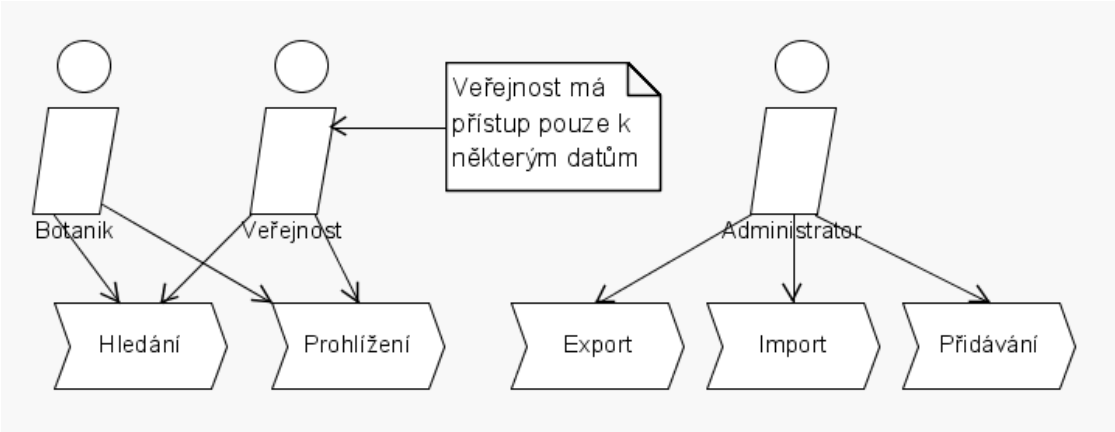
Procesy dnes

Sběr dat



Botanici hledají rostliny a poznamenávají si nálezy do zápisníků. Na konci vegetační sezony vezmou veškerá data, sjednotí je a předají administrátorovi databáze. Administrátor uloží data do databáze.

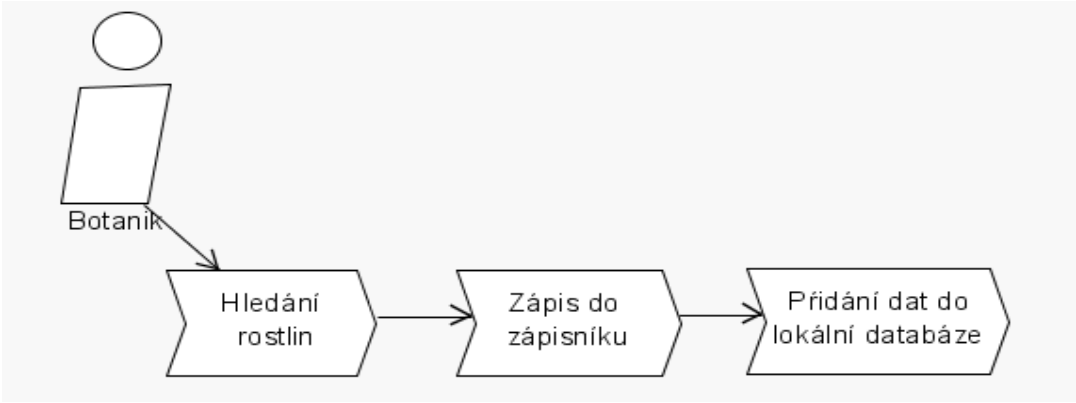
Práce s daty



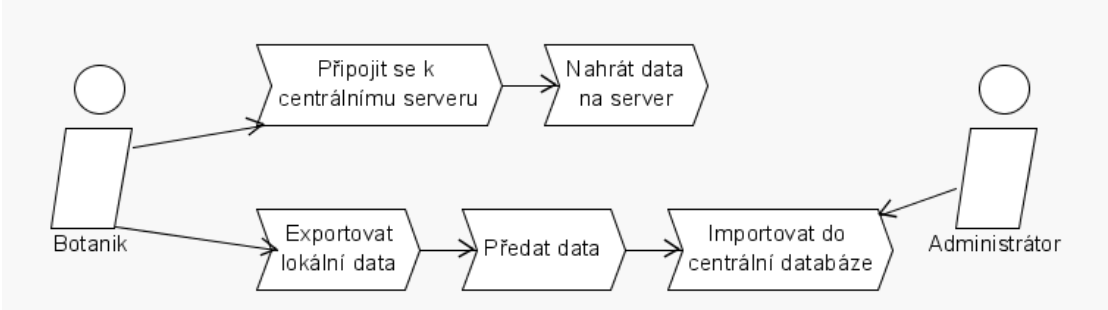
Obvyklé akce při práci s databází jsou vyhledávání, procházení a přidávání dat. Občas administrátor importuje do databáze nové data nebo exportuje data z databáze. Botanici obvykle jenom prohlížejí data a vyhledávají v nich. Veřejnost (uživatelé bez privilegovaného přístupu) mají možnost prohlížet jenom některé data.

Procesy s využitím Plantlore

Sběr dat

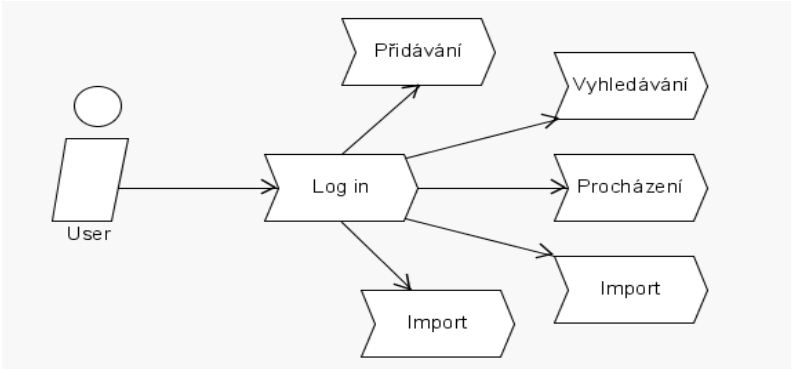


Botanici hledají rostliny a poznamenávají si nálezy do zápisníků. Doma tyto data uloží do lokální instalace Plantlore.



Po určitém čase se buď připojí k Internetu a nahrají své data na centrální server nebo je exportují a doručí administrátorovi centrálního serveru, který je naimportuje do databáze.

Práce s daty



Uživatel se připojí k Plantlore na centrálním serveru. Pro prohlížení dat může použít buď Plantlore klienta nebo webové rozhraní. Pokud chce přidávat nebo upravovat data, připojí se pomocí Plantlore klienta a může měnit data.

III. Použité technologie

Instalátor

Pro Plantlore je použit instalátor IzPack 3.8. Jedná se o multiplatformní instalátor napsaný v Javě, což vyhovuje zejména proto, že se zjednodušila příprava instalátoru pro různé platformy. Instalátor byl otestován na platformě Win32, Linux a Apple Macintosh.

Součástí instalace pro platformu Windows je instalace databázového systému PostgreSQL, pro který jsme využili nativní instalátor dodaný s PostgreSQL pro Windows, který je možné spustit v tichém režimu a zadat mu parametry na příkazové řádce. Nevýhodou je nutnost používat instalační program distribuovaný pro Miscrosoft Windows msiexec pro spouštění msi instalátorů.

Instalátor je lokalizovaný do dvou jazyků, češtiny a angličtiny

Databázový systém

Náš původní cíl byl, aby Plantlore fungovala nad různými databázovými systémy. Nakonec jsme se ale počas vývoje rozhodli, že použijeme systém PostgreSQL díky jeho kompatibilitě s jinými použitými technologiemi, dobré dokumentaci a jednoduché administraci. Důvody, kvůli kterým jsme se rozhodli nepoužít jiné databáze jsou většinou způsobeny nedostatky jiných databázových systémů, příp. jejich nevhodností pro náš projekt. O kterých systémech jsme uvažovali:

1. **FirebirdSQL** – Jak je poznamenáno v části o systému Hibernate, je Firebird v současné stabilní verzi nekompatibilní s tím, jak používáme v Plantlore systém Hibernate. Mezi problémy, na které jsme narazili patří:
    1. Nemožnost vykonat velké SQL SELECT dotazy, jejichž plán vykonání je větší než určitá mez. Problém vyniká v Hibernate, který SQL dotazy generuje sám a často používá velké dotazy s mnoha tabulkami. Toto chování je možné do určité míry upravit, ale dochází při tom k dalším problémům. Firebird 2.0 toto již řeší.
    2. Podle našeho názoru diskutabilní interpretace SQL standardu, kdy není možné použít přejmenování (SELECT sloupek AS popis_sloupku...) ve vnořených SELECT dotazech. Jiné databázovém systémy toto implementují správně. Bohužel Hibernate toto chování využívá a není možné ho vypnout nebo změnit.
2. **MySQL** – Tento systém byl testován v úvodních fázích vývoje, ale narazili jsme na několik tradičních problémů – od vnořených dotazů až po správu rolí a uživatelů. Systém se vyvíjí a v posledních major verzích implementuje mnoho požadovaných vlastností, rozhodli jsme se ale, že existují lepší možnosti
3. **Oracle** – databázový systém Oracle je sice dostupný pro osobní použití, jedná se ale stále o proprietární produkt který je sice vyspělý, ale pro naše potřeby asi málo použitelný, neboť typický botanik si ho nebude instalovat na svůj domácí počítač.
4. **PostgreSQL** – Tento systém nás zaujal jednoduchostí administrace, dostupností dokumentace, kompatibilitou, podporovanými platformami a vcelku dobrou podporou pro kódování UTF8 s kterým jsme měli také problémy u jiných systémů.

Hibernate ORM

Hibernate je systém pro objektově-relační mapování. Umožňuje při práci s relační databází použít objektově orientovaný přístup a pracovat se záznamy v databázi jako s objekty. Jeho nasazení je ideální v kombinaci s jazykem Java, který je postavený na myšlence objektově orientovaného vývoje. Další výhodou, kterou Hibernate přináší, je přenositelnost mezi databázovými systémy díky tomu, že úplně odděluje SQL příkazy (které se v různých databázových systémech více či méně liší) a vlastní kód aplikace. Při jeho použití nemusí programátor používat přímo jazyk SQL, ale může využít pestrou nabídku API, které Hibernate poskytuje včetně vlastního dotazovacího jazyka podobného SQL (který se nazývá HQL – Hibernate Query Language).

Tyto výhody se sebou samozřejmě nesou i řadu nevýhod. Programátor je omezený na to, co poskytuje Hibernate, a nemůže naplno využít potenciál použitého Databázového systému (bez ztráty přenositelnosti).

Naše zkušenosti s Hibernate

Před prací na projektu jsme neměli žádné zkušenosti se systémem Hibernate. Jeho použití nám v některých ohledech pomohlo, ale také sme narazili na závažné problémy v kombinaci Hibernate a jiných technologií. Nastíníme některé z těchto problémů.

1. **Hibernate a RMI**

   Použití Hibernate a Remote Method Invocation se ukázalo problematické, neboť jsme byli nuceni psát vrstvu, která byla mezi Hibernate a samotnou aplikací, aby bylo možné zabezpečit posílání objektů mezi klientem a serverem (pomocí RMI nebylo rozumně možné posílat objekty Hibernatu). Další problém se ukázal při přenosu tzv. holder objektů, které zapouzdřují data z databáze do objektů jazyku Java. Hibernate tyto objekty používá pro ukládání dat do perzistentního úložiště a vadí mu, pokud jsou serializovány a posílány pomocí mechanizmu RMI ze serveru na klienta a zpátky.

2. **Hibernate a DB systém FirebirdSQL**

   Databázový systém Firebird (původní systém Interbase společnosti Borland který byl uvolněn jako open source) v poslední stabilní verzi 1.5.x neimplementuje některé funkce které bylo potřeba pro úspěšné použití v projektu v kombinací s Hibernate. V současné době se dokončuje verze 2.0, která by měla takovéto problémy odstranit, ale rozhodli jsme se ji nakonec nepoužít (i když Plantlore by měla s verzí 2.0 fungovat), neboť se jednalo ještě stále o betaverzi. Problémy, na které jsme narazili, jsou popsané v této dokumentaci v části Databázový systém.

Komunikační vrstva a Remote Method Invocation

Protože jedním z cílů projektu bylo umožnit uživatelům práci se vzdálenými databázemi, bylo nutné navrhnout způsob síťové komunikace. Nabízely se dvě možnosti: **vlastní komunikační vrstva** a využití některé z existujících technologií, konkrétně **RMI**.

Vlastní komunikační vrstva

Původně jsme zvažovali implementaci vlastní komunikační vrstvy pro práci se vzdálenými databázemi. To by ovšem vyžadovalo návrh vlastního rozhraní a jeho implementaci, která by

zahrnovala způsob rozkládání objektů tak, aby je bylo možné posílat po síti a opět rekonstruovat. V aplikaci pracujeme s holder objekty Hibernate, které jsou složitě provázány, museli bychom si umět poradit jednak s nimi a jednak s vracením kolekcí (polí) těchto objektů. Krom toho by bylo zapotřebí vyřešit uspokojivě rozdíly v lokální komunikaci a v komunikaci vzdálené - pokud bychom nechtěli používat dvojí systém práce, museli bychom navrhnout speciální adaptéry, které by pro vyšší vrstvy, tj. těmi, které komunikační vrstvu používají, sjednocovaly síťové a lokální použití vrstvy bezprostředně pod vrstvou komunikační. Postupně se ukázalo, že k tomuto účelu lze stejně dobře použít již existující technologii - technologii vzdáleného volání metod, RMI, která je v každém virtuální stroji (JVM) standardně obsažena.

Další nevýhodou by se pravděpodobně ukázal i návrh serveru a komunikace s klienty pomocí dalších vláken nebo synovských procesů. Výhoda takového řešení by spočívala v ověřené architektuře (dva vývojáři Plantlore měli zkušenosti s psaním serverů - programovali vlastní FTP server).

RMI

Důvody pro použití RMI vyplývaly přímo z kladených požadavků na řešení síťové komunikace, zejména na to, aby

1. celá komunikační vrstva byla maximálně transparentní pro vyšší vrstvy,
2. bylo možné jednoduše sjednotit pohled na komunikaci síťovou i lokální, a aby nedocházelo ke zbytečné režii v případě lokální komunikace.

Hlavní výhodou, kterou RMI nabízí, je vlastní síťový protokol, tedy vlastní transport všech objektů po síti. Jako programátoři bychom se tedy nemuseli starat o to, jak mnohdy komplexní objekty poslat přes síť - RMI tento problém řeší z hlediska programátora velmi elegantně, plně postačuje použít Serializable interface (tento interface nemá žádné metody). Téměř všechny objekty třídy java.land i java.utils jsou již serializovány, mj. i námi používané kolekce. Nebylo by tedy zapotřebí vyvinout žádné větší úsilí.

Další výhodou, která se nám jako programátorům velmi hodila, byl jednotný způsob práce s objekty lokálními (tedy objekty "žijícími" ve stejné JVM, ve které je i volající) a objekty vzdálenými. Jediná penalizace spočívala v odchytávání výjimky RemoteException. Byli bychom s to velmi elegantně, vyřešit problém s komunikační vrstvou splňujíce oba kladené požadavky.

Nevýhodou byla praktická neznalost technologie a jejích úskalí. Zejména bylo potřeba dořešit způsob vytváření objektů ve vzdálené JVM a předávání zpátky na klientskou JVM. Dále bylo nutné zajistit korektní rušení těchto objektů a to i v případě, že vzdálený klient spadl nebo že došlo k dlouhodobému výpadku síťového spojení, neboť se jednalo o objekty databázové vrstvy, které uchovávaly otevřená DB spojení a mnohdy měly provedeny dotazy nad databází.

Závěr

Postupně se ukázalo, že použití RMI bylo dobré rozhodnutí - mohli jsme se soustředit na problémy spojené s celkovou komunikací, ne na detaily implementace přenosu objektů nebo řízení zpracování požadavků na serveru, neboť RMI mechanismus zaručuje vykonání vzdáleného volání v různých vláknech automaticky, uzná-li to za vhodné.

Zejména jsme však dosáhli splnění obou dvou kladených požadavků.

Vyskytly se potíže pouze technického rázu vycházející z různé implementace síťování v

různých operačních systémech, zmiňme např. zvláštní chování RMI při vyhození ConnectionException (výjimka při ztraceném připojení) v OS Windows XP, které způsobovalo viditelné zpomalení celé aplikace. Další problémy souvisely s problematickou identifikací serverů, nastavování cest k třídám pro stuby (v middleware terminologii proxy).

Grafické rozhraní Plantlore

Dlouho jsme se rozhodovali, jaým způsobem navrhneme a budeme programovat uživatelské rozhraní Plantlore.

Matisse

Nejprve jsme GUI začali vytvářet ručně jen ve Swingu většinou za pomoci GridBagLayoutu. To se ale ukázalo jako v podstatě téměř neúnosné pro dialogy té složitosti, které v Plantlore existují. Alternativou byl návrh GUI za pomoci editoru Matisse zabudovaném ve vývojovém prostředí Netbeans 5.0. Od toho nás odrazovala hlavně zanesená závislost na prostředí Netbeans. Nicméně výhody editoru Matisse brzy převážily a postupně jsme celé, už hotové, grafické rozhraní pomocí něho přepsali a nové dialogy už vytvářeli už pouze v něm. Tento krok se nám velmi vyplatil například u dialogů pro zadávání, editaci a hledání v nálezech, které byly často upravovány podle požadavků botaniků. Přechod na Matisse byl také výrazně usnadněný díky použití arichitektury MVC, pro kterou jsme se rozhodli hned na začátku vývoje Plantlore. Bylo tedy potřeba přetvořit pouze pohledy(*View.java) a znovu je navázat na příslušné modely a kontrolery(*Ctrl.java).

Swing

Různé potíže jsme museli překonat i při používání samotného Swingu. Plantlore často musí vykonávat různé databázové operace, které mohou trvat potenciálně velmi dlouho. Proto není možné provádět všechny výpočty pouze v jednom vlákně. Jinak by okna Plantlore zamrzala, nepřekreslovala se, což by pro uživatele mohlo být velmi matoucí.

Vlákna a modální progress bary

Tento problém jsme vyřešili použitím více vláken. Zprvu jsme vlákna vytvářeli nejednotně, každý ve své komponentě. To ale bylo nepřehledné, proto se zavedla třída Task, která usnadňuje a sjednocuje provádění dlouhých výpočtů v novém vlákně. Spolu s třídou Task byl s problémy naprogramován i modální, blokující progress bar. Ten dlouho fungoval poměrně dobře, ale jak se Plantlore stala ještě složitější projevily se dva nové problémy. Modální progress bar, protože je blokující, neumožňuje za svého chodu zobrazení dalšího modálního blokujícího progress baru. To následně může vést k deadlocku, který se začal objevovat. To souvisí s druhým problémem. Pokud by byl uživatel netrpělivý (nebo měl pomalý počítač) a rychle vyvolával různé příkazy používající Task, tak by se dané úlohy prováděly najednou v nových vláknech. To mohlo vést k dalším konfliktům. Z tohoto důvodu byla vymyšlna třída Dispatcher, která jednotlivé Tasky synchronizuje a neumožní souběžný běh dvou nebo více Tasků.

„Single thread“ podmínka

Pro úplné vyřešení problému s deadlocky bylo nutné také provést revizi kódu a zajistit, že je dodržena takzvana single thread podmínka Swingu. Ta říká, že k deadlocku Event Dispatch vlákna

nedojde, pokud je ke komponentám Swingu přistupováno vždy pouze z Event Dispatch vlákna (a ne z uživatelských vláken).

Jasper reports

V požadavcích na Plantlore je i požadavek na možnost tisku nálezových dat a na generování sched. V tomto případě jsme se díky předchozím zkušenostem rovnou rozhodli pro použití open-source knihovny JasperReports. Její použití bylo víceméně bezproblémové. Bylo nutné vyřešit problém s propagací výjimek (viz programátorská dokumentace). Nedořešený je problém informování uživatele o průběhu výpočtu. Momentálně se v Tasku pouští pouze metoda knihovny JasperReports, která pomocí zadaného, námi naprogramovaného zdroje vyplní zadaný formulář. Díky tomu není možné žádným snadným čistým způsobem informovat Observery Tasku o průběhu výpočtu. Proto se při generování sched a při tisku uživateli zobrazí pouze „neurčitý“ progress bar. Řešením by mohlo být datovému zdroji při konstrukci předávat referenci na Task, ve kterým je použitý. To by ale znamenalo, že pro každý Task by bylo nutné datový zdroj vytvořit znova. Jiným řešením by bylo v našem datovém zdroji implementovat i nějaký náš nový interface, který by umožňoval zadání a změnu Tasku, který se má použít pro informování o průběhu výpočtu. Ani to ale bohužel není ideální řešení.

Zpracování XML dat (DOM4J, SAX)

Plantlore hojně využívá technologie XML. V XML formátu jsou uloženy konfigurační soubory a do tohoto formátu musí umět Plantlore provádět export záznamů z databáze a stejně tak musí umět z XML načítat data pro následný import záznamů.

V případě práce s XML jsme nechtěli opakovat již provedenou práci a psát si vlastní parsery XML, použili jsme známé a prověřené knihovny SAX a DOM4J.

DOM4J

DOM4J umožňuje snadné vytvoření reprezentace XML stromu ze souboru. Mezi jeho slabiny patří zejména vysoké paměťové nároky, proto jej bylo lze použít pouze pro případy malých konfiguračních souborů nebo rozdílových seznamů pro import dat přímo do tabulek, které slouží k jejich opravě či doplnění.

Ukázalo se, že v obou případech (konfig. soubory i rozdílové seznamy) je použití DOM4J ospravedlnitelné - konfig. soubory jsou velmi malé a rozdílové seznamy mohou obsahovat řádově tisíce položek, aniž by bylo zapotřebí extremních paměťových nároků (nejobsáhlejší záznamy tabulek TPlants je možné zpracovávat v řádech tisíců, přičemž paměťové nároky jsou několik desítek MB, např. soubor 5tis rostlin potřebuje cca 20MB paměti, což při velikostech dnešních operačních pamětí nepředstavuje žádný problém). Pro představu uveďme, že v celé České republice je rozpoznáváno zhruba 6tis větších sídel a aplikace Plantlore pracuje se zhruba 5tis druhy rostlin. Je nutné si uvědomit, že zamýšlené použití pro import do tabulek je oprava některých záznamů nebo přidání nových záznamů - a v případě větších sídel nebo rostlin nehrozí nárůst více než o několik desítek záznamů ročně.

DOM4j byl dále použit pro konstrukci výstupu při exportu záznamů z databáze. Zde bylo jeho použití poněkud upraveno, jelikož by nebylo možné postavit DOM strom v paměti pro několik tisíc nálezových záznamů, natož pak pro avizované desetitisíce (zadavatelé nashromáždili přibližně 50tis nálezových dat). DOM4J byl použit pro konstrukci XML stromu vždy pouze pro jeden

záznam a pak okamžitě zapsán na výstup; zbývající XML tagy byly vygenerovány ručně. Tímto bylo efektivně využito výhod, které DOM4J nabízí (především snadné generování stromu a jeho zápis do souboru), aniž by to vedlo na zvýšené paměťové nároky. Export vyžaduje paměť v řádu desítek kB během celého procesu.

SAX

Pro import nálezových dat nebylo již DOM4J možno vůbec použít z důvodu velikosti vstupních dat. Proto byl použit (pro programátora trochu méně příjemný) způsob parsování souboru pomocí SAX, který pouze reportuje nastalé události (nalezení tagů, text mezi tagy), a je na programátorovi, aby z daných událostí sestavil adekvátní strom - v našem případě tedy rekonstruoval čtený záznam.

Užití SAXu nás zbavilo problémů spojených s parsováním souboru, ale znemožnilo přímou aplikaci návrhového vzoru Builder, protože SAX pracuje na principu zpětného volání (callback); architektura importu nálezových dat je postavena odlišně.

Obě technologie našly své uplatnění a zjednodušily zpracování XML souborů, třebaže u DOM4J bylo zapotřebí pečlivě monitorovat spotřebovanou paměť a u SAXu poupravit architekturu aplikace. Při jejich použití se žádné problémy nevyskytly.

Lokalizace a internacionalizace

Jedním z interních požadavků na funkčnost Plantlore byla i lokalizace a internacionalize. Lokalizování rozhraní Plantlore sleduje cíl případné použitelnosti aplikace i mimo Českou republiku. Lokalizace je řešena od počátku vývoje Plantlore. Třída L10n poskytuje podle aktuálního nebo nastaveného locale přistup k lokalizovaným řetězcům podle zadaného klíče. Poskytuje také podporu pro internacionalizaci formou poskytování aktuálního používaného locale.

Při vytváření klíčů pro řetězce jsme z počátku nebyli jednotní, což vytvářelo zmatek v Plantlore.properties souborech uchovávajících překlady. Proto jsme strukturu klíčů určili konvencí, kterou je nutné dodržovat. Její dodržování je nutné pro přehlednost hlavně v tomto případě, kdy existuje jeden properties sobor pro celou aplikaci. Zvažovali jsme i použití samostatných properties souborů pro každou třídu, případně pro každý package. Zvolili jsme ale cestu jednoho společného souboru, protože naše konvence struktury klíčů je v prostředí Javy přirozená a navíc tento systém umožňuje jednoduché sdílení jednou definovaného klíče různými částmi aplikace. Hojně se tak děje například u klíčů s předponou „Common.“.

Nápověda

Plantlore poskytuje uživateli systém interaktivní nápovědy. Pro implementaci nápovědy jsme použili knihovnu JavaHelp 2.0 která poskytuje platformně nezávislý systém interaktivní nápovědy. Zdrojové soubory nápovědy jsou ve formátu HTML a JavaHelp obsahuje HTML prohlížeč pomocí kterého je možné nápovědu prohlížet.

Výhodou je kromě přenositelnosti také jednoduchá lokalizace nápovědy, možnost nastavit klávesové skratky (typicky klávesa F1) pro automatické otevření nápovědy a další.

Jediný problém, na který jsme při použití systému JavaHelp narazili byla detekce kódování zdrojových HTML souborů (systém se ne vždy řídil deklarací kódování v <meta> tagu). Řešili jsme také možnost maximalizace okna s nápovědou v případě, že je nápověda otevřená z modálního

dialogu (okno není možné v tomto případě maximalizovat). Toto je ale vlastnost systému a ne chyba, protože vlastnosti okna nápovědy se dědí po oknu, ze kterého byla nápověda otevřena a teda u modálních dialogů není maximalizace možná.

Transformace souřadnic

Při zadávání nálezových dat v aplikaci Plantlore může uživatel upřestnit místo zadáním souřadnic GPS. GPS je implicitně udávána v souřadnicovém systému WGS-84, ale umožňuje i přepočet do některých jiných souřadnicových systémů.

Na území ČR jsou platné tyto souřadnicové systémy :

1. **Světový geodetický referenční systém 1984 (WGS84)** - je definován souborem pozemních stanic a polohami družic systému GPS. Je navržený tak, aby odpovídal celé zeměkouli.
2. **Souřadnicový systém Jednotné trigonometrické sítě katastrální (S-JTSK)** – je to rovinný systém. Zeměpisná šířka a délka v S-JTSK je charakterizována na Besselově elipsoidu. Při převodu sférických souřadnic na rovinné se používá Gaussova koule.
3. **Souřadnicový systém 1942 (S-42)** - je to rovinný systém. Základem tohoto systému je Gaussovo konformní válcové zobrazení, založené na Krasovského elipsoidu, rozděleného na šestistupňové (někdy třístupňové) poledníkové pásy. Každý pás je zvlášť promítnut do roviny a má svou vlastní pravoúhlou souřadnicovou soustavu. Rovník prohlásíme za osu y a k rovníku kolmý 15. poledník za osu x. Souřadnice x nějakého bodu je vzdálenost v metrech od osy y, tedy od rovníku. Souřadnice y by analogicky měla být vzdálenost od 15. poledníku. Aby se předešlo záporným hodnotám západně od osového poledníku, přičítá se k souřadnici 500 000 metrů a aby se předešlo záměně souřadnic v jednotlivých pásech, předřazuje se ještě na začátek pořadové číslo pásu.

Zabývali jsme se převody mezi souřadnicovými systémy WGS-84, S-JTSK a S-42. Algoritmy a informace k převodům jsme dohledávali na internetu.

- http://astro.mff.cuni.cz/mira/sh/sh.php
- http://gpsweb.cz/JTSK-WGS.htm
- http://sas2.elte.hu/tg/majster.htm

Převod ze systému WGS-84 do S-JTSK a zpět

WGS-84 používá elipsoid WGS-84, kdežto S-JTSK používá elipsoid Besselův .

Transformace z WGS84 do S-JTSK:

1. Transformace elipsoidu WGS84 na Besselův elipsoid (Hrdina, Heltmertova transformace)
2. Transformace z Besselova elipsoidu do kartézského souř. systému S-JTSK (známá jako Křovákovo zobrazení)
    - zobrazení z Besselova elipsoidu na Gaussovu kouli
    - vypocet kartografickych souradnic na Gaussove kouli s posunutym polem Q o souradnicich 48°15`, 42°30`
    - zobrazení z Gaussovy koule na plochu koule

Transformace z SJTSK do WGS84:

1. Křovákova transformace
2. Výpočet pravouhlých souřadnic z geodetických souřadnic pro Besselův elipsoid
3. Transformace z Besselova elipsoidu na elipsoid WGS84 (Hrdina, Heltmertova transformace)

Převod ze systému WGS-84 do S-42 a zpět

WGS-84 používá elipsoid WGS-84, kdežto S-42 používá elipsoid Krasovského .

Transformace z WGS84 do S-42:

1. Výpočet pravouhlých souřadnic z geodetických souřadnic pro elipsoid WGS84
2. Transformace elipsoidu WGS84 na elipsoid Krasovského (Heltmertova transformace)
3. Výpočet geodetických souřadnic z pravouhlých souřadnic pro elipsoid Krasovsky
4. Mercatorova projekce

Transformace z S-42 do WGS-84:

1. Mercatorova projekce
2. Výpočet pravouhlých souřadnic z geodetických souřadnic pro elipsoid Kraovsky
3. Transformace pravouhlých souřadnic z S-42 do WGS84 (Heltmertova transformace)
4. Výpočet geodetických souřadnic z pravouhlých souřadnic pro elipsoid WGS84

Převod ze systému S-JTSK do S-42 a zpět

S-JTSK používá elipsoid Basseův a S-42 používá elipsoid Krasovského.

Vzájemný převod mezi těmito souřadnicovými systémy je řešen přes systmém WGS-84.

Při zadávání korektních hodnot dojde jen k malé chybě při převodu souřadnic ze zvoleného souřadnicového systému na jiný a zpět.

BioCASE

Webový klient v Plantlore je řešen pomocí BioCASE Provider Sofware (BPS). Toto řešení umožní zapojit se do evropského výzkumného projektu BioCASE. BPS je možné nalézt na adrese http://www.biocase.org/products/provider_software/

The Biological Collection Access Service for Europe, BioCASE, je mezinárodní síť všech možných druhů biologických sbírek. BioCASE umožňuje rozšíření jednotného přístupu k distribuované a různorodé evropské databázi sbírek a pozorování a to vše s využitím systémově nezávislého open-source software a otevřených datových standardů a protokolů.

Zkušenosti s BioCASE

Ve chvíli, kdy jsme začali pracovat s BioCASE, tak byla k dispozici verze 2.2.2. V této verzi nebyl funkční test software a komunikace s databází Firebird. Kontaktovali jsme vývojáře, od kterých jsme (po delší komunikaci) získali opravenou verzi programu. V další verzi byla tato chyba

již opravena.

Během většiny vývoje jsme pracovali s verzí BioCASE provider software 2.3.1., která byla v té době aktuální. V téte verzi jsme se setkali s několika problémy, kde některé jsme řešili přímo s vývojáři tohoto software.

Problémy s konfigurací:

- **Změna hesla** – nebylo možné změnit heslo administrátora BPS pře webové rozhranní
- **Definování databázové struktury** – zjistili jsme, že nelze nastavit dva cizí klíče z jedné tabulky do druhé
- **Mapování** – tato fáze byla časově náročná a vyžadovala nastudování standardů ABCD 1.20, ABCD 2.06 a Darwin Core 2. Bylo potřeba seznámit se podrobně s jednotlivými schématy, s biologickými termíny a daty. V této fázi jsme komunikovali s botaniky z Jihočeské botanické společnosti, využívali dostupných popisů stanardů (http://darwincore.calacademy.org/Documentation/DarwinCore2Draft_v1-4_HTML , http://www.bgbm.org/biodivinf/Schema/) a tréninkového příkladu pro BioCASE. Každé schéma obsahuje povinné a nepovinné položky. Po namapování povinných položek stále nebyly testy software (mapovaní) úspěšné. Bylo zjištěno, že je nutné rozšířit základní mapování ABCD schémat o další položky (country, recordBasis)

Vyhledávání v nálezových datech:

Při práci s databázovým systémem Firebird 1.5 jsme řešili problém s kódováním. Použité kódování v databázi bylo UNICODE_FSS a kódování nastavené pro BPS bylo UTF-8. Vyhledávání nálezových dat vždy proběhlo nesprávně pokud se ve výsledku vyhledávání vyskytoval speciální znak (č, ř, …). Tuto chybu se nám povedlo odstranit zásahem do aplikace (skripty v jazyce Python), kde bylo nutné ošetřit kódování při vytvoření XML souboru. Toto řešení pomohlo zobrazit výsledek vyhledávání, ale neumožnilo další práci s nálezem, který obsahoval speciální znak. Problémy s kódováním jsme řešili emailovou komunikací s vývojáři BPS, od kterých jsme získali nové skripty. Do současnosti ale neznáme uspokojivé řešení tohto problému.

Přotože se ještě před odevzdáním projektu objevila novější verze BPS 2.4.1., přešli jsme na tuto aktuální verzi. Nová verze s sebou mimojiné přinesla

- rozšíření vyhledávacích formulářů pro více standardů
- opravenou funkci pro změnu hesla administrátora

Použité knihovny

Seznam použitých knihoven a technologií, které obsahují.

Technologie	*Soubory*	*Popis*
Log4j	log4j-1.2.13.jar	Nástroj pro zaznamenávání logovacích výpisů.
jCalendar	jcalendar-1.3.1.jar	Knihovna pro výběr data z kalendáře.
JGoodies look & feel	looks-2.0.2.jar	Knihovna obsahující vzhled uživatelského

Technologie	***Soubory***	***Popis***
		rozhraní v Javě.
Dom4j	dom4j-1.6.jar	Knihovna pro parsování a práci s XML daty.
Matisse	swing-layout-1.0.jar	Knihovna použitá pro návrh uživatelského prostředí.
Jasper reports	jasperreports-1.2.1.jar	Nástroj pro tvorbu, export a tisk reportů.
Apache Ant	antlr-2.7.5H3.jar	Nástroj pro sestavování a spouštění build-skriptů.
Hibernate	hibernate3.jar	Systém pro objektově relační mapování.
Java Help	jh.jar	Systém pro tvorbu interaktivní nápovědy.
JDBC Drivers	mysql-connector-java-3.1.12-bin.jar postgresql-8.0-311.jdbc2.jar jaybird-full-2.1.0.jar	JDBC ovladače pro MySQL, PostgreSQL a Firebird.
Spin	spin.jar	Knihovna pro usnadňující vývoj "nezamrzajících" uživatelských rozhraní ve Swingu.