Web based database backup with JDBC

Tymoteusz Gedliczka, Marcin Grabda, Tomasz Gurgul, Wiktor Kołodziej, Jakub Suder

12 stycznia 2006

1 Zespół

Wiktor Kołodziej team leader, public relations, code

Tomasz Gurgul scribe, code

Marcin Grabda code, testing

Jakub Suder code, testing

Tymoteusz Gedliczka code, testing

2 Określenie tematu

Archiwizacja danych poprzez mechanizmy JDBC

3 Opis problemu

Wyobraźmy sobie system informatyczny, który korzysta z bazy danych (być może wykorzystuje różne typy baz, np MySQL, Postgresql, MSSQL). Problem pojawia się, gdy zachodzi potrzeba archiwizacji danych z bazy, a możemy użyć tylko technologii JDBC (gdyż np. chcielibyśmy tą funkcjonalność zintegrować z większym, istniejącym systemem). Istnieją różne aplikacje służące do archiwizacji, jednak żadna z nich nie spełnia wszystkich naszych założeń (obsługa wielu różnych typów baz, użycie JDBC).

4 Proponowane rozwiązanie

Postanowiliśmy napisać system, który będzie umożliwiał administratorowi danego systemu:

- zalogowanie się do systemu poprzez interfejs WWW

  Panel administracyjny umożliwiający tworzenie kopii zapasowej bazy danych będzie udostępniany po uprzedniej autentykacji. Użytkownik w celu uzyskania dostępu do panelu będzie musiał podać login i hasło. System będzie przechowywał zakodowane hasło w pliku typu „properites". Po stwierdzeniu przez mechanizm autoryzacji poprawności tych danych będzie dopiero możliwe dokonanie czynności administracyjnych związanych z tworzeniem kopii bezpieczeństwa. Aby wprowadzane dane były bezpieczne, zostanie wykorzystane tunelowanie po SSL.

- stworzenie pliku konfiguracyjnego zawierającego listę baz danych z informacjami potrzebnymi do archiwizacji (adres, login, hasło itd.)

  Aby archiwizacja baz danych była możliwa system musi posiadać niezbędne do nawiązania połączenia dane:

  - nazwa bazy
  - typ bazy
  - adres i port serwera
  - login
  - hasło

  Po zalogowaniu do systemu, użytkownik zobowiązany jest wypełnić powyższe dane (alternatywnie, zamiast części z tych danych użytkownik może podać specyficzny dla danego silnika baz danych URL połączenia). Dane te następnie zapisywane są w postaci stworzonego przez system pliku konfiguracyjnego, który w dalszej fazie zostanie wykorzystany do nawiązania połączenia. Wykorzystanie pliku pliku konfiguracyjnego pozwala na zapamiętanie informacji o wszystkich bazach, na których przeprowadzamy proces archiwizacji. Dzięki temu ponowne stworzenie backupu tej samej bazy wymaga jedynie wybrania jej z listy (bez konieczności ponownego wpisywania informacji). System uwzględnia również możliwość edycji danych zapisanych do pliku.

- połączenie się z bazami danych i stworzenie pliku z backupem, który można zapisać na lokalnej maszynie

  Aby rozpocząć proces archiwizacji, użytkownik zaznacza pożądaną bazę i wybiera opcję „Backup". System zczytując dane z pliku konfiguracyjnego próbuje nawiązać połączenie z wyszczególnioną bazą. Jeśli połączenie nie zostanie nawiązane, wygenerowany będzie odpowiedni komunikat o błędzie informujący o prawdopodobnej jego przyczynie (np. niepoprawne dane,

podanie nieobsługiwanego typu bazy danych, niedostępność serwera).
Po pomyślnym połączeniu z bazą, rozpoczyna się właściwy proces archiwizacji polegający na zczytaniu wszystkich tabel. Dane standardowo będą zapisywane do wybranego w konfiguracji lokalnego katalogu oraz własnego formatu binarnego.
Będzie również możliwość wyboru innych formatów, co zostanie opisane w dalszych punktach.
Użytkownik ma również możliwość zaznaczenia, czy ma zostać poinformowany o fakcie stworzenia backupu (może to być proces czasochłonny, np. przy sporych rozmiarów bazie danych). Jeżeli zaznaczy tę opcję, to na uprzednio zdefiniowany w ustawieniach adres mailowy zostanie wysłana wiadomość. Gotowy plik z backupem będzie miał w nazwie nazwę bazy danych oraz datę; pełna nazwa pliku będzie postaci: <numer_id_bazy>_<nazwa_bazy>_<data>_<format>.<rozszerzenie> gdzie:

- numer_id_bazy - numer bazy z pliku zawierającego listę baz, która jest wyświetlana na podstronie „Database list”
- nazwa_bazy - nazwa („Title”) bazy przetworzona tak, aby można było jej użyć w nazwie pliku (np. z podkreśleniami zamiast spacji)
- data - data utworzenia backupu w formacie ddmmrrrrhhmm (dni+miesiące+rok+godziny+minuty)
- format - np. „Binary”
- rozszerzenie - „zip” lub „gz”

Ścieżka do katalogu zapisu znajduje się w pliku konfiguracyjnym i można ją ustawić na stronie „Settings”. W razie potrzeby plik z backupem może zostać skompresowany. Po utworzeniu będzie można go ściągnąć przez przeglądarkę WWW na lokalny dysk (na stronie „Backup list” będzie widoczna lista wszystkich plików backupu, które są dostępne na serwerze). Lista ta będzie przechowywana w javowym pliku „properies”.

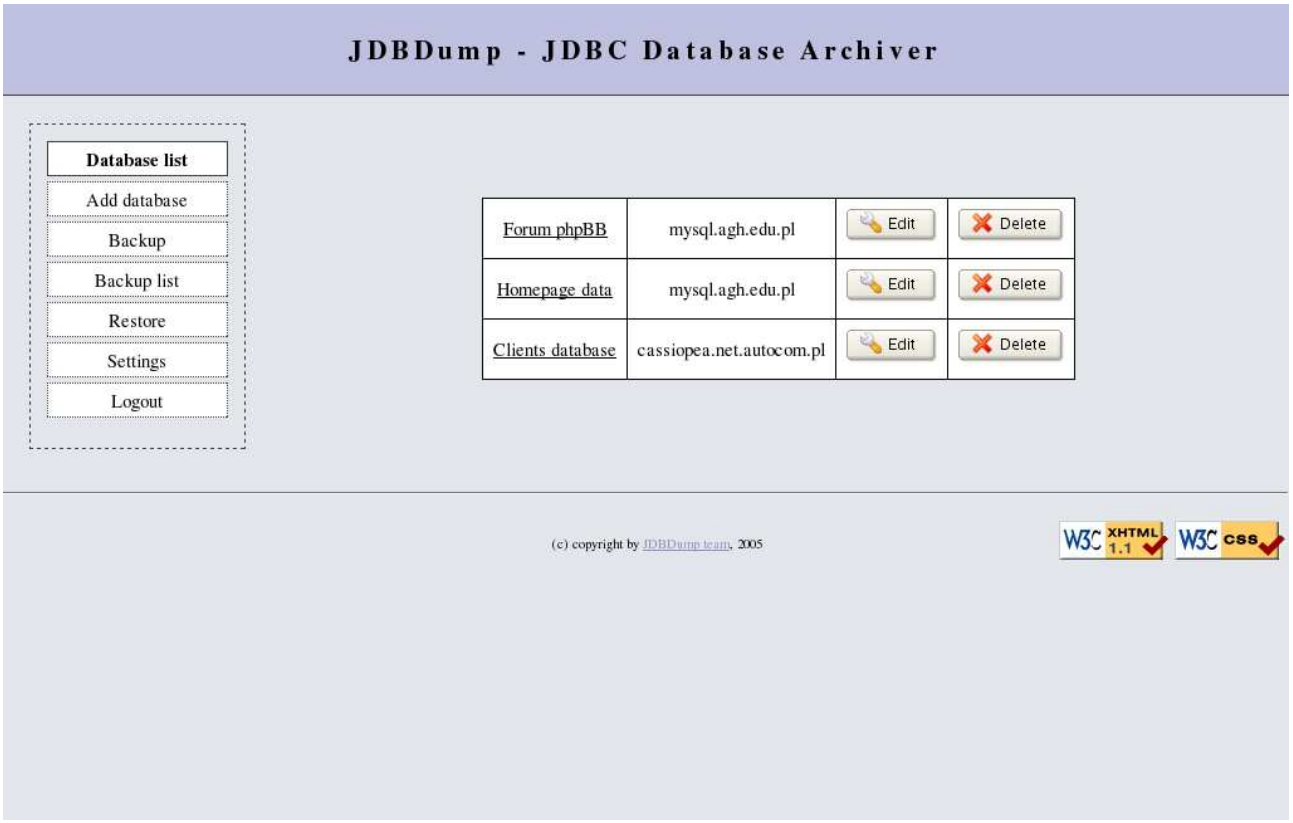
- w razie potrzeby odtworzenie wszystkich baz przy użyciu uprzednio stworzonego pliku z backupem
  Podstawową funkcjonalnością obok tworzenia pliku kopii bezpieczeństwa, jaką będzie udostępniał moduł, będzie odzyskiwanie bazy z takiegoż pliku. Użytkownik będzie mógł wybrać bazę (stan bazy), którą będzie chciał odzyskać.
  Zakładając, że plik z backupem jest spójny, odtworzenie bazy sprowadza się do usunięcia wszystkich tabel i przywrócenia danych.
  W tym celu administrator na swoim panelu administracyjnym wybiera opcję „Restore”. W oknie wyboru pliku należy podać plik z uprzednio stworzonym archiwum bazy. Plik ten można wybrać z lokalnego katalogu jak również z katalogu na serwerze.

- wylogowanie się z systemu

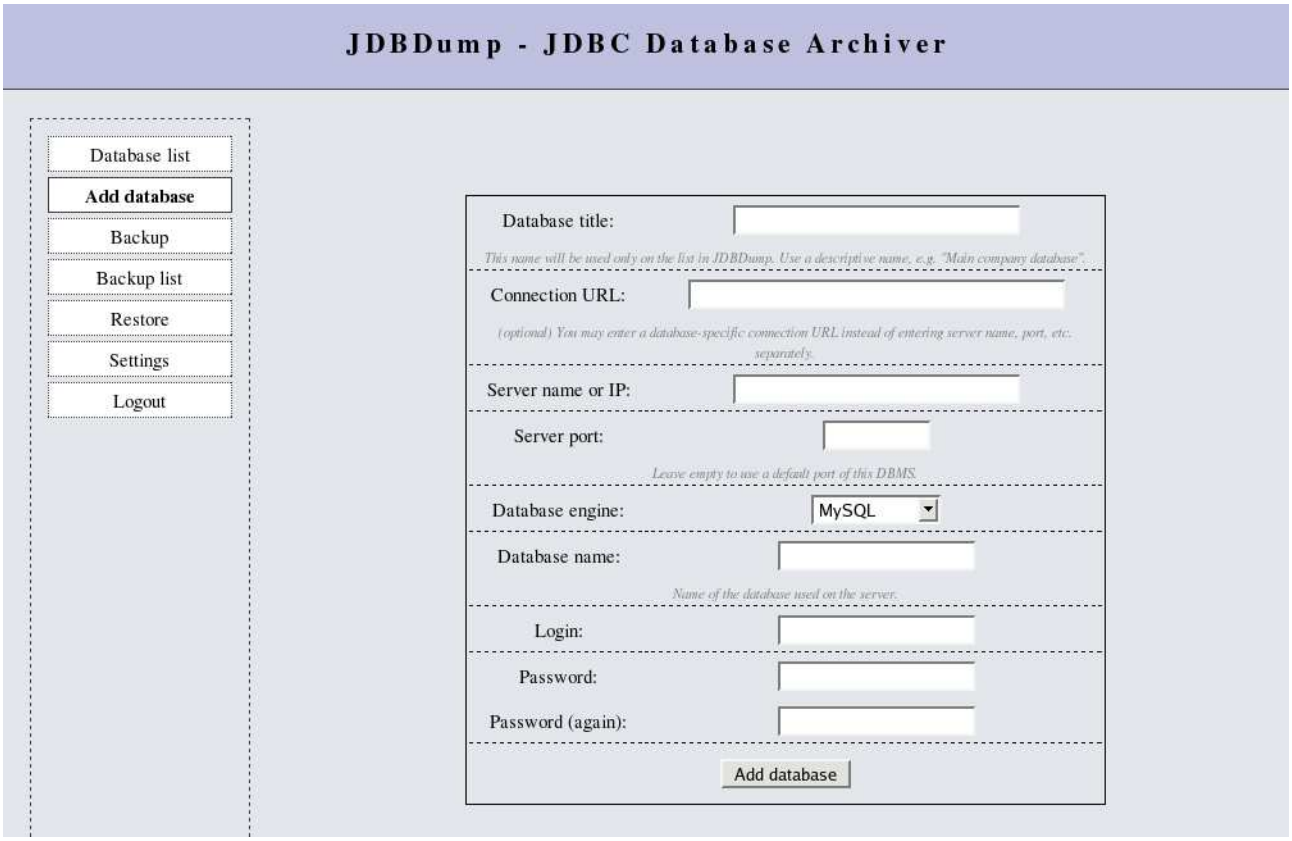
5 Web-interfejs użytkownika

Dostęp do systemu JDBDump odbywa się poprzez interfejs WWW. Mówi się, że jeden obraz wart jest tysiąca słów. Kierując się tym przelicznikiem prezentujemy interfejs na poniższych screenshotach.

Rysunek 1: Logowanie do systemu

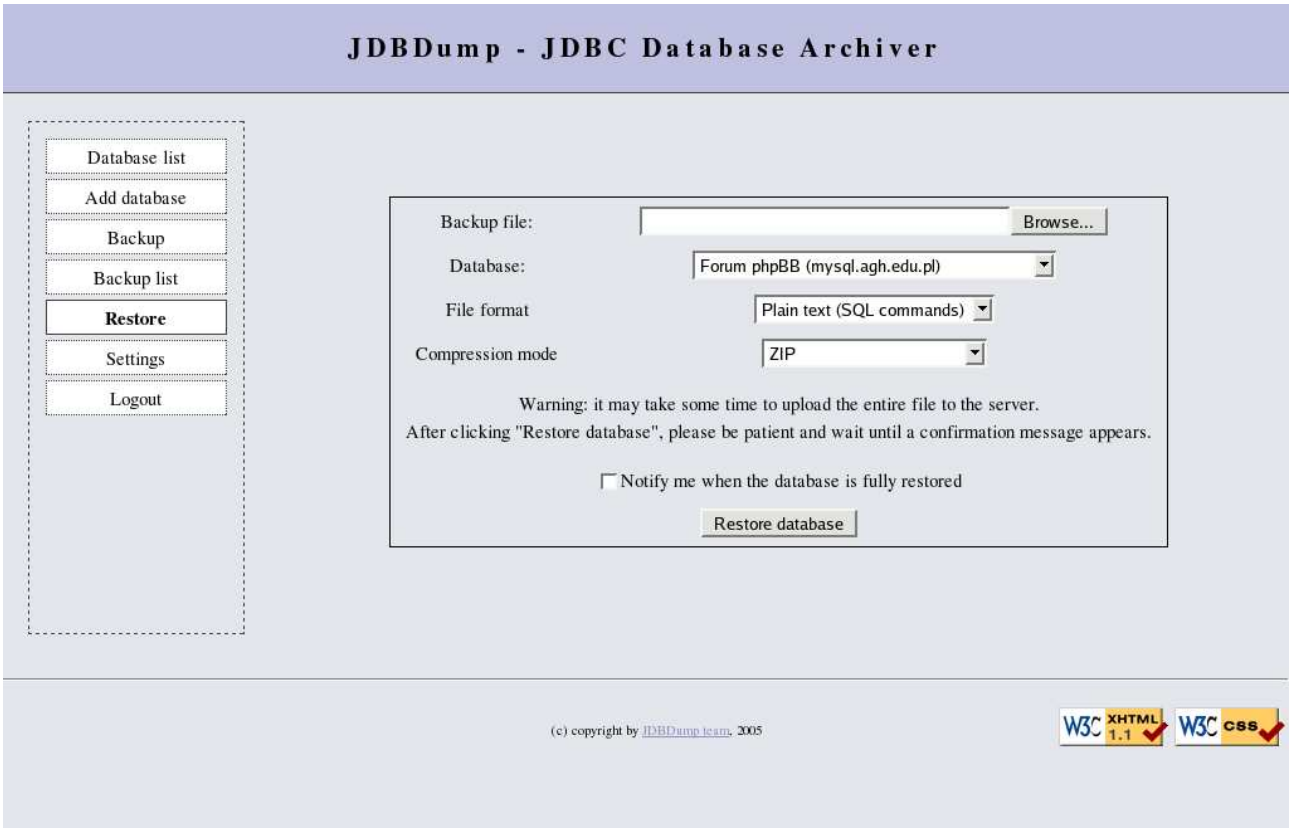


Rysunek 2: Lista baz danych



Rysunek 3: Dodawanie bazy

Rysunek 4: Archiwizacja bazy



Rysunek 5: Odtworzenie bazy

6 Możliwe problemy implementacyjne i decyzyjne

- format zapisu danych

  Aby się dało łatwo odtworzyć bazę, skorzystamy z naszego własnego formatu binarnego. Jednak nie będzie to jedyna możliwość. Postanowiliśmy stworzyć system na tyle elastyczny, by można było do niego przy pomocy wtyczek dodać obsługę różnych formatów zapisu danych. Gdy tylko zostanie załadowany odpowiedni plugin, będzie można obraz bazy zapisać w pliku tekstowym, np. w formacie mysqla czy postgresa. Dzięki temu aplikacja nie tylko doskonale nadaje się do ewentualnego późniejszego odtworzenia bazy, lecz również do ewentualnej konwersji pomiędzy różnymi bazami danych.

- kompresowanie pliku (opcja)

  Bazy danych przechowują zazwyczaj bardzo duże ilości danych (rzędu wielu megabajtów). W związku z tym, niezależnie od tego jaki format zapisu wybierzemy, plik wynikowy z backupem będzie sporych rozmiarów. Taki plik będzie użytkownikowi dość trudno będzie ściągnąć na dysk lokalny, zwłaszcza jeżeli będzie on dysponował wolnym łączem internetowym (co zwłaszcza w Polsce nie powinno nikogo dziwić...). Tymczasem taki plik prawdopodobnie będzie się bardzo dobrze nadawał do kompresji, gdyż większą jego część będą zapewne stanowiły powtarzające się ciągi znaków. Kompresja plików w Javie jest bardzo prosta, nie trzeba ściągać żadnych zewnętrznych bibliotek - funkcje te są dostępne w standardowej bibliotece Javy (pakiet java.util.zip). Dostępne są dwa algorytmy kompresji - do pliku formatu ZIP i GZIP. Aby dać użytkownikowi większy wybór, pozwolimy mu samemu zdecydować którego algorytmu chce użyć (czy też może woli w ogóle nie kompresować pliku, jeżeli nie jest on zbyt duży). Prawdopodobnie administrator pracujący na codzień w Windowsie będzie wolał, żeby system przysłał mu plik *.zip, a administrator linuxowy - plik *.gz, więc w ten sposób wszyscy powinni być zadowoleni.

- zabezpieczanie pliku backupowego oraz połączenia przed niepowołanym dostępem (szyfrowanie).

  Plik wynikowy będzie można poddać opcjonalnie zakodowaniu. Jednakże najprostszą metodą będzie korzystanie z naszego systemu poprzez kanał ssl i właśnie ten sposób będziemy zalecali użytkownikom.

- wybór SZBD, które nasza aplikacja będzie obsługiwała

  Serwery baz danych używają wielu różnych systemów zarządzania relacyjnych baz danych. Jak wiadomo, wszystkie one implementują wspólny standard SQL, ale każda z nich interpretuje niektóre szczegóły nieco inaczej, każda też ma nieco inne możliwości (np. najpopularniejszy chyba MySQL dopiero w obecnej wersji, wydanej pod koniec października, zaczął obsługiwać widoki, procedury składowane i triggery). Dlatego nie da się napisać jednego uniwersalnego modułu który ściągałby wszystkie dane ze wszystkich typów baz. Możemy albo ograniczyć się

tylko do pewnego zakresu elementów które są obsługiwane przez wszystkie bazy (ale wtedy z części z nich otrzymywalibyśmy niekompletne dane, i taki dump nie nadawałby się w zasadzie do niczego), albo wybrać kilka typów które chcemy obsługiwać, i do każdego z nich podejść indywidualnie. Postanowiliśmy napisać część systemu odpowiedzialną za ściąganie danych w taki sposób, aby z jednej strony zminimalizować czas pisania kodu, z drugiej - umożliwić obsługę jak największej liczby typów baz. Zostanie w tym celu wykorzystana specjalna grupa klas:

- jedna klasa główna, która będzie implementowała wszystkie możliwe funkcje w najbardziej ogólny sposób
- kilka klas dziedziczących po niej, które będą wykonywały niektóre zapytania w sposób specyficzny dla konkretnych typów (np. klasa GeneralConnection i podklasy MysqlConnection, PostgresqlConnection itd.)

W ten sposób te funkcje, które wszystkie bazy wykonują tak samo, wystarczy napisać tylko raz w klasie głównej, a specyficzne funkcje zostaną zdefiniowane w podklasach. Klasa ogólna będzie mogła być wykorzystana przez użytkownika w sytuacji, kiedy będzie chciał spróbować zrobić backup bazy jakiegoś mało znanego typu, którego obsługi nie przewidzieliśmy (niestety nie możemy zagwarantować, że mu się to uda).

Aby umożliwić użytkownikom łatwe rozszerzanie systemu o kolejne typy baz, spróbujemy napisać system w ten sposób, aby listę dostępnych klas obsługujących bazy tworzył dynamicznie, przeszukując odpowiedni pakiet w poszukiwaniu klas dziedziczących po ogólnej klasie obsługi połączenia. W ten sposób użytkownik nie będzie musiał nawet nic zmieniać w naszym końcowym pliku *.jar, a tylko podłączy do systemu inny plik ze swoimi klasami obsługującymi, a system powinien je wykryć i umożliwić korzystanie z nich.

- wybór wersji javy: 1.5
- zastosowanie JNDI do łączenia z bazami danych. JNDI zajmie się łączeniem z bazami, aplikacja łączy się z bazą pobierając DataSource z JNDI
- kolejność wgrywania danych przy odtwarzaniu systemu (tak aby nie powodować konfliktów)

  Na początku zostanie stworzona struktura tabel, następnie zostaną one wypełnione danymi, by na końcu dodać constrainty.
- ograniczone możliwości JDBC

  Aby rozpoznać nazwy tabel w bazie danych skorzystamy z tzw. metadanych (klasa DatabaseMetaData).

7 Diagramy

Dokumentacja obejmuje następujące diagramy: diagram klas, diagram przypadków użycia i diagram sekwencji. Zostały one umieszczone w suplemencie do tego dokumentu, w celu uzyskania większej czytelności diagramów.

8 Wzorce projektowe wykorzystane w projekcie

8.1 Singleton

Singleton oznacza, iż w ramach programu pojawia się pojedyncza instancja danego obiektu. Wzorzec ten jest wykorzystywany, gdy tworzenie więcej niż jednej instancji danej klasy nie jest wskazanie.

Lista klas, w których użyty został ten wzorzec:

- Configuration - klasa trzymająca konfigurację, potrzebny tylko jeden obiekt takiej klasy
- DatabaseConnectionFactory - klasa produkująca na żądanie połączenie z danym typem bazy danych, wystarczy jedna instancja.

8.2 MVC - Model-View-Controller

Jeden z pierwszych wzorców jakie opisano [Krasner i Poper, 1988], pozwala na oddzielenie interfejsu użytkownika od modelu danych i zarządzania nimi. W naszym projekcie rolę tę pełni Struts.

8.3 Fabryka - Factory

Prosta klasa, która na podstawie jakiejś zmiennej podejmuje decyzję o utworzeniu jednej z możliwych podklas. Lista klas, w których użyty został ten wzorzec:

- DatabaseConnectionFactory - klasa produkująca na żądanie połączenie z danym typem bazy danych

8.4 Façade

Façade polega na tym, by ukryć złożony proces, uprościć go. W naszym przypadku będzie to ukrycie kolejnych kroków tworzenia i odzyskiwania dumpa bazy danych, odpowiednio pod przyciskami „Dump” i „Restore” w interfejsie użytkownika.

9 Wykorzystanie sourceforge.net

Projekt postanowiliśmy uruchomić na sourceforge.net. Serwis ten zapewnia takie udogodnienia jak konto www na stronę projketu, listę mailingową, konto cvs, czy system zgłaszania bugów. Na sourceforge.net powstało tysiące znakomitych projektów tworzonych przez ludzi z całego świata, mamy

nadzieję, że i nasz trafi do grona tych, które zakończyły się powodzeniem. Nazwa naszego projektu na sourceforge.net to: jdbdump.

10 Wykorzystanie Maven'a

Wygenerowanie struktury katalogów źródeł, testami oraz dokumentacji, a także wygenerowania strony głównej naszego projektu użyliśmy narzędzia Maven. Strona projektu znajduje się pod adresem

`http://jdbdump.sourceforge.net`

11 Wykorzystanie CVS'a

Skorzystaliśmy z CVS'a dostępnego na sourceforge.net, można przeglądać aktualny stan repozytorium pod tym adresem:

`http://cvs.sourceforge.net/viewcvs.py/jdbdump/`

12 Narzędzia do modelowania UML

Do sworzenia diagramów wykorzystaliśmy dwa narzędzia:

- Poseidon - jego zaletą jest wieloplatformowość (napisany w javie), wadą natomiast zużycie pamięci (dobre 150 MB), posłużył do stworzenia diagramu klas oraz przypadków użycia, niestety nie udało się stworzyć diagramów sekwencji, a ściślej nie udało ich się zapisać, dlatego do tych ostatnich użyliśmy drugiego z wymienionych narzędzi.

  `http://gentleware.com/`

- Enterprise Architekt - bardzo szybkie narzędzie, ogromne możliwości, działa pod Windowsem a także pod emulatorem Wine pod linuxem. Posłużył do stworzenia diagramów sekwencji, których nie udało się zapisać przy pomocy Poseidona.

  `http://www.sparxsystems.com.au/`

13 Changelog

05.11.2005 Rozwinięcie opisu problemu

06.11.2005 Podjęcie części decyzji implementacyjnych i opisanie ich

07.11.2005 Opis interfejsu użytkownika

08.11.2005 Założenie projektu jdbdump na sourceforgu : jdbdump

08.11.2005 Wygenerowanie dokumentacji w mavenie : maven

14.11.2005 Powstanie diagramu klas

14.11.2005 Powstanie listy dyskusyjnej projektu na sourceforge

15.11.2005 Powstanie diagramu przypadków użycia

15.11.2005 Aktualizacja dokumentacji wynikła z powstania diagramu klas

18.11.2005 Powstanie dokumentacji w wersji pdf

19.11.2005 Powstanie makiety interfejsu WWW

20.11.2005 Dodano w dokumentacji: opis użytych wzorców projektowych, oraz wykorzystywanych technologii

22.11.2005 Przejście tylko na angielską wersję językową, być może w przyszłości angielski projekt zostanie przetłumaczony.