Db4o – obiektowa baza danych – wersja .NET

Wstęp

Db4o (database for objects) to obiektowa baza danych na platformę Java i .NET.

Pełna wersja bazy db4o jest dostępna na dwóch licencjach:
- open source: pozwala na ściągnięcie bazy oraz jej użycie w niekomercyjnych projektach
- komercyjna: do wykorzystania w projektach komercyjnych

Instalacja db4o

Instalacja bazy jest bardzo prosta, wystarczy rozpakować archiwum. Można ją ściągnąć ze strony:

```
http://developer.db4o.com/files/folders/db4o_72/default.aspx
```

Dodatkowo należy ściągnąć program ObjectManager (wykorzystywany w drugiej części opracowania) ze strony:

```
http://db4o-om.googlecode.com/files/objectmanager-7.2-java.zip
```

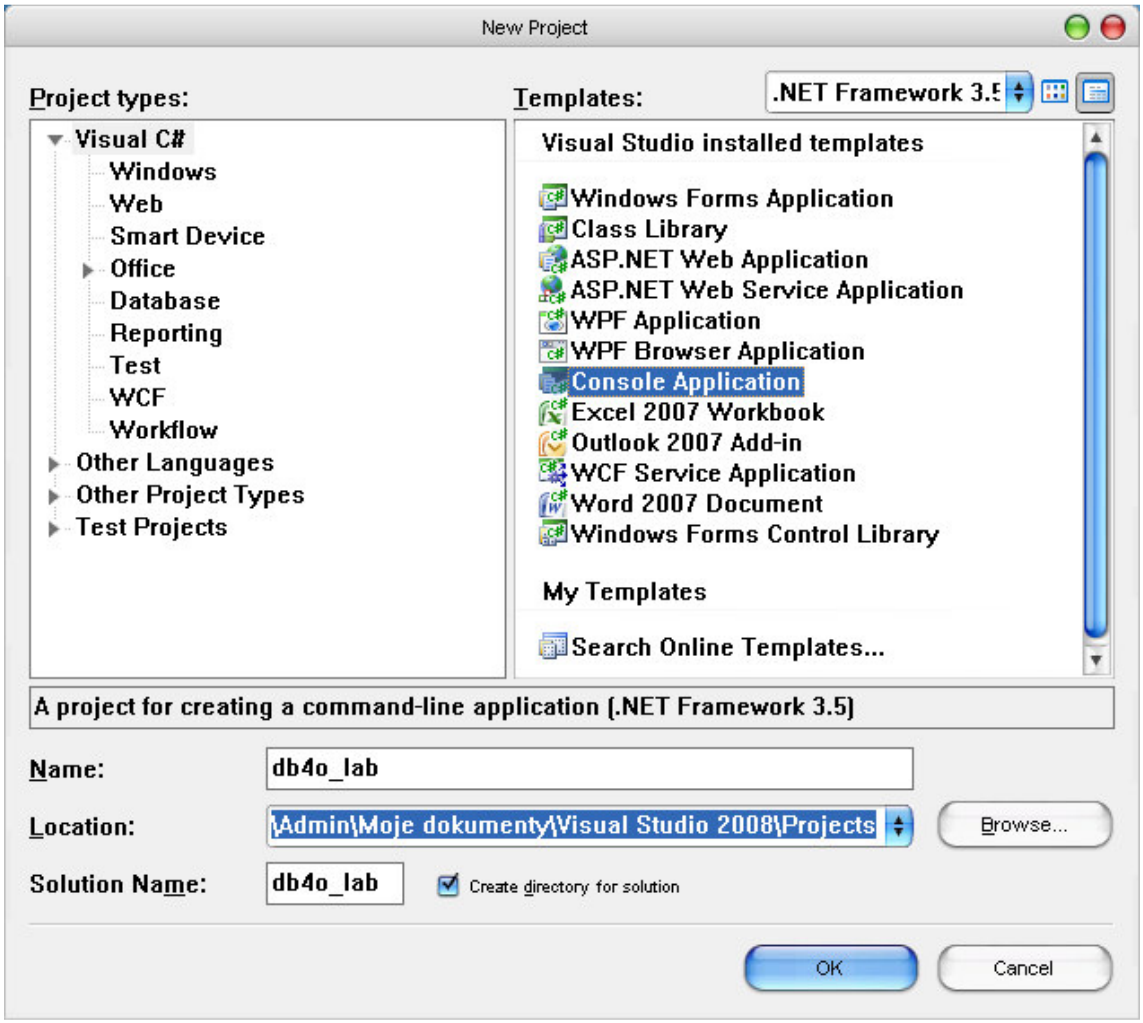
Strona wymaga zalogowania (login i hasło: politechnika).

Zakładam, że mamy ściągnięte pliki **db4o-7.2-net35.msi** lub **db4o-7.2-net2.msi** (w zależności od tego, z jakiej wersji .NET Framework chcemy korzystać) oraz **objectmanager-7.2-java.zip**. Aby można było korzystać z bazy należy mieć zainstalowane środowisko programistyczne Visual Studio 2005 lub Visual Studio 2005. Potrzebne też nam będzie środowisko uruchomieniowe (JRE) Javy do uruchomienia programu ObjectManager, które można ściągnąć z następującej strony:

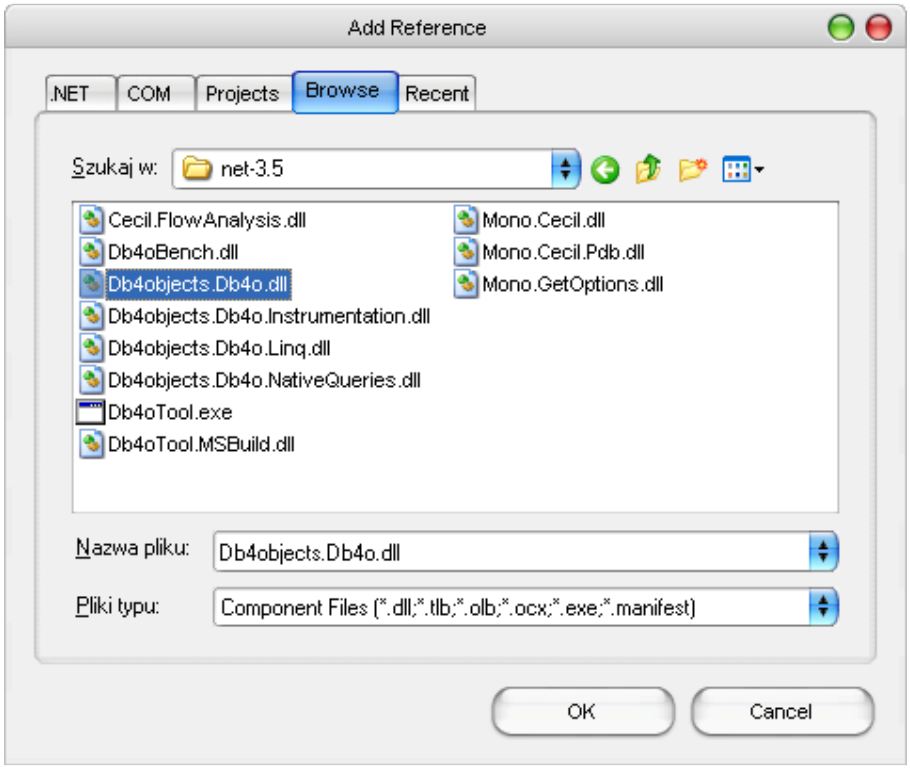
```
http://java.sun.com/javase/downloads/index.jsp
```

Następnie instalujemy bazę danych do wybranego katalogu np. `C:\Program Files\Db4objects\db4o-7.2.`

Uruchamiamy teraz Visual Studio i tworzymy nowy projekt klikając: `File->New->Project...` z listy dostępnych szablonów (ang. templates) wybieramy Console `Application`, Jako nazwę projektu podajemy db4o_lab. Efekt tych kroków widzimy na poniższym zrzucie ekranu. Potwierdzamy utworzenie projektu przyciskiem OK.



Teraz dodamy do projektu bibliotekę Db4objects.Db4o.dll, aby to wykonać należy wybrać `Project->Add reference…` w wyświetlonym oknie dialogowym wybieramy zakładkę `Browse` i odszukujemy bibliotekę w miejscu instalacji systemu bazy danych, co pokazuje poniższy zrzut ekranu:



Stworzenie bazy danych

Baza danych db4o składa się z jednego pliku. Gdy odwołujemy się do nieistniejącej bazy, jest ona automatycznie tworzona. Praca z bazą odbywa się przez objekt `ObjectContainer`. Na początku stworzymy prostą klasę o nazwie `Person`. Klikamy prawym myszy na nasz projekt i następnie na `Add->New item` z listy szablonów wybieramy `Class` i nazywamy go `Person`. Wypełniamy ciało klasy w następujący sposób:

```
public class Person
{
    private String _name;
    private int _age;

    public String Name
    {
        get { return _name; }
        set { _name = value; }
    }
```

```
        public int Age
        {
            get { return _age; }
            set { _age = value; }
        }

        public Person() { }

        public Person(String name, int age)
        {
            _name = name;
            _age = age;
        }

        public override String ToString()
        {
            return "[" + _name + ";" + _age + "]";
        }
    }
```

Należy zwrócić uwagę, że w klasie nie ma kodu powiązanego z bazą db4o.

Klasę Program w analogiczny sposób wypełniamy jak poniżej. Przed uruchomieniem aplikacji należy stworzyć katalog `C:\myDb`.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Db4objects.Db4o;
using Db4objects.Db4o.Query;
using Db4objects.Db4o.Config;

namespace db4o_lab
{
    class Program
    {
        static void Main(string[] args)
        {
            System.IO.File.Delete("C:/myDb/myDb.yap");
            IConfiguration conf = Db4oFactory.Configure();
            conf.MessageLevel(0); // 0=silent, 3=loud
            IObjectContainer db = Db4oFactory.OpenFile(conf,
            "C:/myDb/myDb.yap");
            try
            {
                db.Set(new Person("Gandhi", 79));
                db.Set(new Person("Lincoln", 56));
                IObjectSet result = db.Get(new Person());
                // pobierz wszystko
                ListResult(result);
```

```
                Person p = new Person();
                p.Name = "Gandhi";
                IObjectSet result2 = db.Get(p);
                Person p2 = (Person) result2.Next();
                p2.Age = 90; // zwiększenie wieku
                db.Set(p2);
                result2.Reset();
                ListResult(result2);
                db.Delete(p2); // usuń obiekt Ghandi
                IObjectSet result3 =
                (IObjectSet) db.Get(new Person());
                ListResult(result3); // pobierz wszystko
            }
            finally
            {
                db.Close();
                Console.Read();
            }
        }

        private static void ListResult(IObjectSet result)
        {
            Console.WriteLine(result.Count);
            foreach (object item in result)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine("---------------");
        }
    }
}
```

W dalszej części mówię działanie tego programu.

Aby mieć dostęp do bazy danych należy stworzyć obiekt `ObjectContainer`:

```
IObjectContainer db = Db4oFactory.OpenFile(conf,
"C:/myDb/myDb.yap");
```

Jako parametr metody statycznej `OpenFile` podajemy ścieżkę do bazy, która chcemy stworzyć.
Plik ten może osiągnąć rozmiar do 256 GB. Bazę można skasować przy pomocy komendy:

```
System.IO.File.Delete("C:/myDb/myDb.yap");
```

Stworzenie i zapisanie obiektu:

```
Person p = new Person("Gandhi", 79);
db.Set(p);
```

Do obiektu automatycznie przypisywany jest unikalny ID. Aby wszystkie zmiany zostały zapisane do bazy należy wywołać komendę `Close()`, która jednocześnie zamyka bazę danych.
Podstawową funkcjonalnością każdej bazy danych jest wykonywanie zapytań. Jeden ze sposobów tworzenia zapytań w db4o to query-by-example (QBE). Do pobierania obiektów służy komenda `Get()`. W poniższym kodzie dodamy przykładowy obiekt do bazy, a następnie go pobierzemy.

```
Person p = new Person();
p.Name = "Gandhi";
IObjectSet result2 = db.Get(p);
Person p2 = (Person) result2.Next();
```

Wywołanie metody `Get()` zwraca referencje do obiektu o interfejsie `IObjectSet` z db4o API. W działaniu przypomina klasyczny iterator. Używając metod `HasNext()` i `Next()` można odwołać się do wszystkich obiektów w bazie pasujących do obiektu szablonu. Powyższy kod dał by następujący rezultat:

```
[Gandhi;79]
```

Dla większej czytelności kodu stworzyliśmy metodę `ListResult`, która wyświetla wszystkie obiekty znajdujące się w `IObjectSet`:

```
private static void ListResult(IObjectSet result)
{
    Console.WriteLine(result.Count);
    foreach (object item in result)
    {
        Console.WriteLine(item);
    }
    Console.WriteLine("---------------");
}
```

Dzięki czemu kod naszego zapytania wygląda tak:

```
Person p = new Person();
p.Name = "Gandhi";
IObjectSet result2 = db.Get(p);
ListResult(result2);
```

Aby zaktualizować obiekt trzeba po prostu go pobrać, dokonać zmian i ponownie zapisać do bazy przy pomocy polecenia `Set()` np.:

```
IObjectSet result=(IObjectSet)db.Get(new Person("Lincoln",0));
Person lincoln = (Person)result.Next();
lincoln.Age = 56;
db.Set(lincoln);
```

Przykład kasowania obiektów:

```
IObjectSet result = (IObjectSet)db.Get(new Person());
Person p = (Person)result.Next();
db.Delete(p);
```

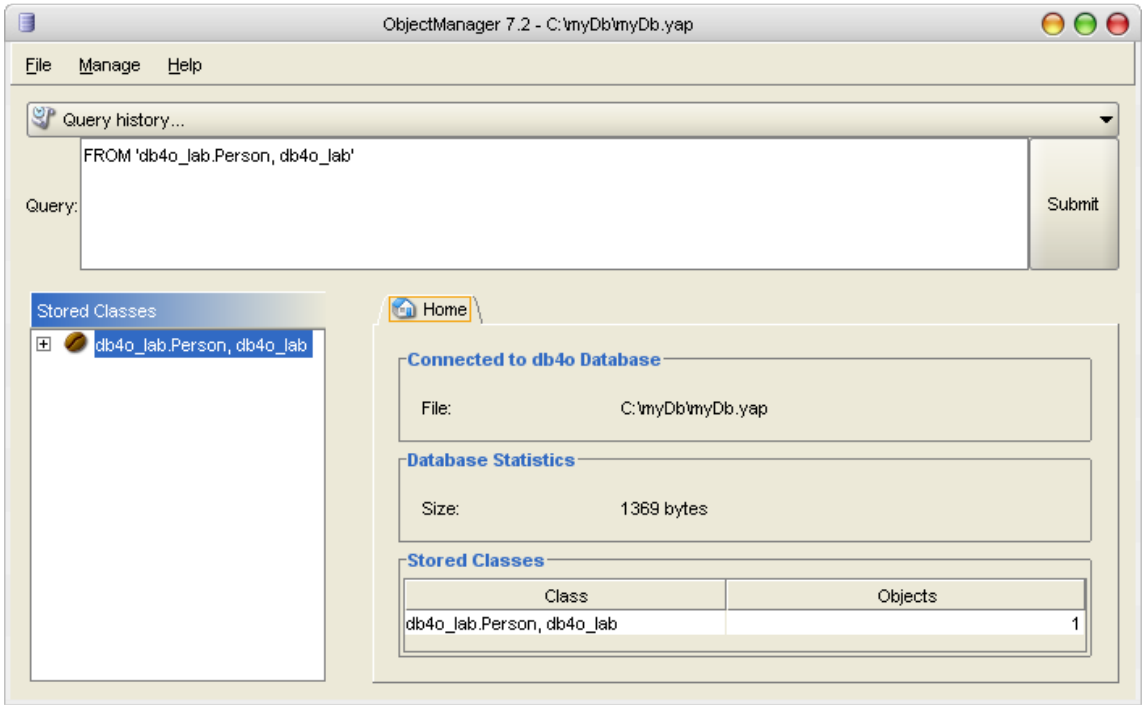
Wynikiem działania programu jest:

```
2
[Gandhi;79]
[Lincoln;56]
---------------
1
[Gandhi;90]
---------------
1
[Lincoln;56]
---------------
```

Praca z ObjectManager

Wielu użytkowników baz danych zamiast używać konsoli preferuje korzystać z graficznych narzędzi. Db4o dostarcza takie narzędzie do przeglądania i edytowania danych oraz tworzenia zapytań o nazwie ObjectManager. Różni się on trochę od podobnych narzędzi tego typu dla relacyjnych baz danych, gdyż zarządza się tutaj obiektami, a nie tabelami. Aby rozpocząć pracę z ObjectManager, rozpakowujemy archiwum **objectmanager-7.2-java.zip** do wybranego katalogu np.: `C:\objectmanager`, następnie wchodzimy do tego katalogu i uruchamiamy plik `objectmanager.bat`. Po kilku sekundach pojawia się następujące okno:

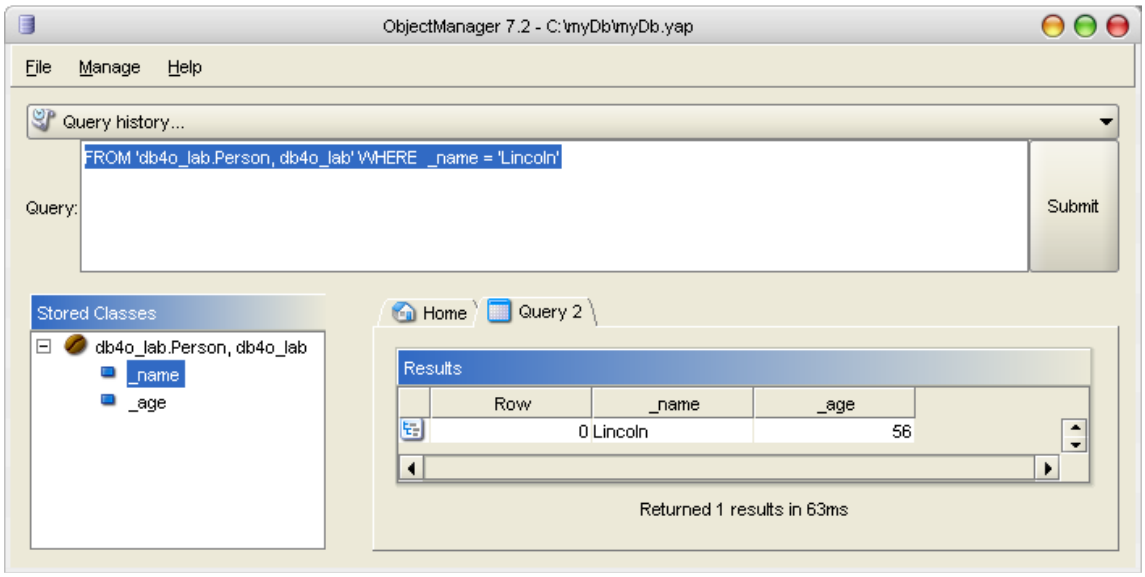
Klikamy na przycisk `Browse`, wskazujemy na ścieżkę `C:\myDb\myDb.yap` i klikamy na przycisk `Open`.



W górnej części znajduje się okno do tworzenia zapytań, natomiast po lewej stronie znajduje się lista zapisanych w bazie klas. Klikając na pola klas możemy w łatwy sposób tworzyć zapytania. Przykładowe zapytanie:

```
FROM 'db4o_lab.Person, db4o_lab' WHERE  _name = 'Lincoln'
```

Otrzymany rezultat:



Db4o – obiektowa baza danych – wersja Java

Wstęp

Db4o (database for objects) to obiektowa baza danych na platformę Java i .NET.

Pełna wersja bazy db4o jest dostępna na dwóch licencjach:

- open source: pozwala na ściągnięcie bazy oraz jej użycie w niekomercyjnych projektach
- komercyjna: do wykorzystania w projektach komercyjnych

Instalacja db4o

Instalacja bazy jest bardzo prosta, wystarczy rozpakować archiwum. Można ją ściągnąć ze strony strony:

```
http://developer.db4o.com/files/folders/db4o_64/entry41701.aspx
```

Dodatkowo należy ściągnąć program ObjectManager (wykorzystywany w drugiej części opracowania) ze strony:

```
http://developer.db4o.com/files/folders/objectmanager_64/default.aspx
```

Strona wymaga zalogowania (login i hasło: politechnika).

Zakładam, że mamy ściągnięte pliki db4o-6.4-java.zip oraz objectmanager-6.4.zip. Aby można było

```
http://javadl.sun.com/webapps/download/AutoDL?BundleId=18713
```

korzystać z bazy należy mieć zainstalowaną javę, którą można ściągnąć z nastepującej strony:

Aby sprawdzić czy mamy poprawnie zainstalowaną javę możemy wpisać w konsoli:

```
java –version
```

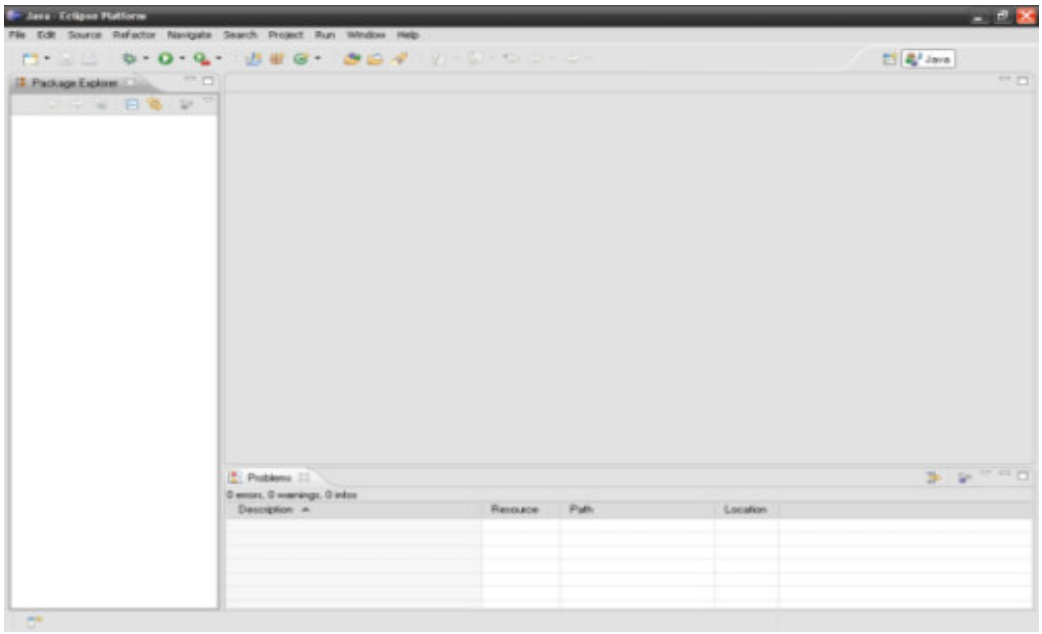
Następnie rozpakowujemy bazę danych do wybranego katalogu np.: `C:\db4o-6.4`.

Będziemy korzystać ze środowiska programistycznego Eclipse (można też użyć innego środowiska lub notatnika :)), które można ściągnąć stąd:

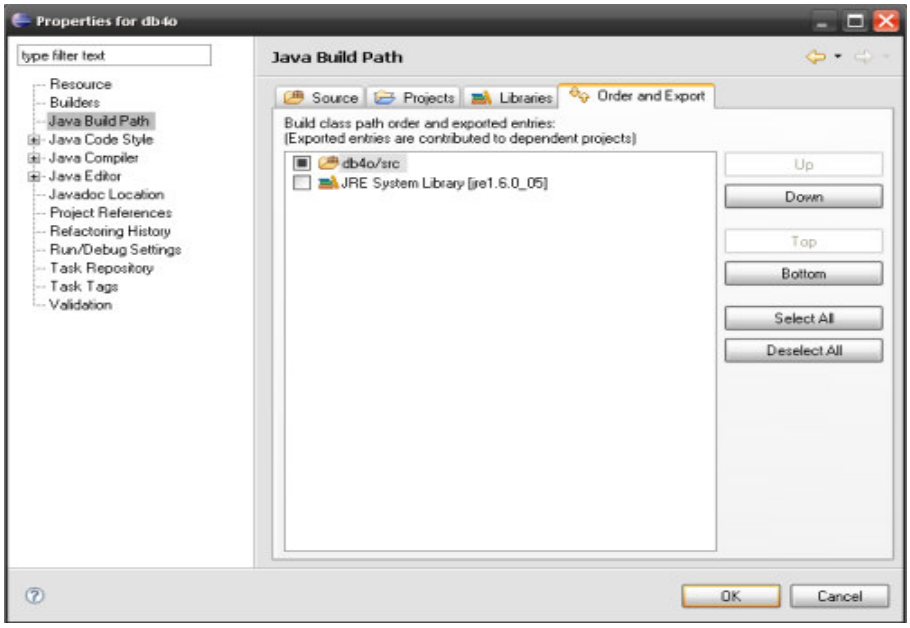
```
http://www.eclipse.org/downloads/
```

Następnie rozpakowujemy archiwum Eclipse i uruchamiamy klikając na eclipse.exe. Jako workspace wpisujemy `C:\workspace` i klikamy OK.

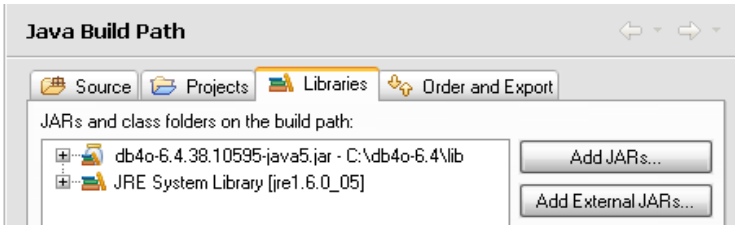
Gdy pozamykamy niepotrzebne zakładki nasze środowisko wygląda tak:



Tworzymy nowy projekt klikając na `File->New->Java Project`. Nazywamy go `db4o` i klikamy OK. Następnie klikamy prawym klawiszem myszy na nasz projekt i wybieramy properties. Klikamy na trzecią od góry zakładkę `Java Build Path`. Mamy następujące okno:



Klikamy następnie na zakładkę `Libraries` oraz na `Add External Jars`. Wybieramy bibliotekę `C:\db4o-6.4\lib\db4o-6.4.38.10595-java5.jar` klikamy OK i zamykamy okno `properties`.



Stworzenie bazy danych

Baza danych db4o składa się z jednego pliku. Gdy odwołujemy się do nieistniejącej bazy, jest ona automatycznie tworzona. Praca z bazą odbywa się przez objekt `ObjectContainer`.

Na początku stworzymy prostą klasę o nazwie `Person`. Klikamy prawym myszy na nasz projekt i następnie na `new->Class`. W oknie które się pojawi w pole Package wpisujemy `example`, a w pole Name wpisujemy `Person`. Wypełniamy ciało klasy w następujący sposób:

```
public class Person
{
    private String _name;
    private int _age;

    public Person(){}

    public Person(String name, int age)
    {
        _name = name;
        _age = age;
    }

    public int getAge()
    {
       return _age;
    }

    public void setAge(int value)
    {
        _age = value;
    }
```

Należy zwrócić uwagę, że w klasie nie ma kodu powiązanego z bazą db4o (dziedziczenia itp).

Tworzymy drugą klasę o nazwie `CompleteExample` w analogiczny sposób i wypełniamy jak

```
import java.io.File;
import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;

public class CompleteExample {
    public static void main(String[] args) {
        new File("C:/myDb/myDb.yap").delete();
        Db4o.configure().messageLevel(0); // 0=silent, 4=loud
        ObjectContainer db = Db4o.openFile("C:/myDb/myDb.yap");
        try {
            db.set(new Person("Gandhi", 79));
            db.set(new Person("Lincoln", 56));
            ObjectSet result = (ObjectSet) db.get(new Person());
            listResult(result); // pobierz wszystko

            Person p = new Person();
            p.setName("Gandhi");
            ObjectSet result2 = (ObjectSet) db.get(p);
            Person p2 = (Person) result2.next();
            p2.setAge(90); // zwiększenie wieku
            db.set(p2);
            result2.reset();
            listResult(result2);

            db.delete(p2); // usuń obiekt Ghandi
            ObjectSet result3 = (ObjectSet) db.get(new Person());
            listResult(result3); // pobierz wszystko
        } finally {
            db.close();
        }
    }

    public static void listResult(ObjectSet result) {
        while (result.hasNext()) {
            System.out.println(result.next());
        }
        System.out.println("---------------");
    }
}
```

poniżej. Przed uruchomieniem aplikacji należy stworzyć katalog `C:\myDb` .

W dalszej części mówię działanie tej klasy.

Aby mieć dostęp do bazy danych należy stworzyć obiekt ObjectContainer:

```
ObjectContainer db = Db4o.OpenFile("C:/myDb/myDb.yap");
```

Jako parametr metody statycznej `openFile` podajemy ścieżkę do bazy, która chcemy stworzyć. Plik

```
File.Delete("C:/myDb/myDb.yap")
```

ten może osiągnąć rozmiar do 256 GB. Bazę można skasować przy pomocy komendy:

Stworzenie i zapisanie obiektu:

```
Person p = new Person("Gandhi", 79);
db.set(p);
```

Do obiektu automatycznie przypisywany jest unikalny ID. Aby wszystkie zmiany zostały zapisane do bazy należy wywołać komendę `Close()`, która jednocześnie zamyka bazę danych.

Podstawową funkcjonalnością każdej bazy danych jest wykonywanie zapytań. Jeden ze sposobów tworzenia zapytań w db4o to query-by-example (QBE). Do pobierania obiektów służy komenda `get()`. W poniższym kodzie dodamy przykładowy obiekt do bazy, a następnie go pobierzemy.

```
Person p = new Person();
p.setName("Gandhi");
ObjectSet result = (ObjectSet) db.get(p);
while (result.hasNext())
System.out.println((Person) result.next());
```

Wywołanie metody `get()` zwraca referencje do klasy `ObjectSet` z db4o API. W działaniu przypomina klasyczny iterator. Używając metod `hasNext()` i `next()` można odwołać się do wszystkich obiektów w bazie pasujących do obiektu szablonu. Powyższy kod dał by następujący rezultat:

```
[Gandhi;79]
```

Dla większej czytelności kodu stworzyliśmy metodę `listResult`, która wyświetla wszystkie obiekty znajdujące się w `ObjectSet`:

```
public static void listResult(ObjectSet result)
{
   while (result.hasNext())
   System.out.println(result.next());
}
```

Dzięki czemu kod naszego zapytania wygląda tak:

```
Person p = new Person();
p.setName("Gandhi");
ObjectSet result = (ObjectSet) db.get(p);
listResult(res);
```

Aby zaktualizować obiekt trzeba po prostu go pobrać, dokonać zmian i ponownie zapisać do bazy przy pomocy polecenia `set()` np

```
ObjectSet result = (ObjectSet) db.get(new Person("Lincoln", 0));
Person lincoln = (Person) result.next();
lincoln.setAge(56);
db.set(lincoln);
```

Przykład kasowania obiektów:

```
ObjectSet result = (ObjectSet) db.get(new Person());
Person p = (Person) result.next();
```

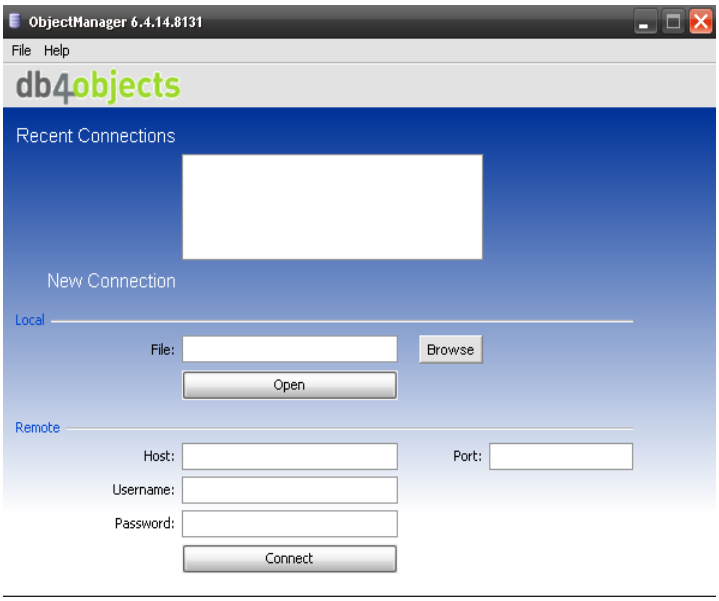
Kiedy wiemy już jak działa klasa, możemy sprawdzić jej działanie klikając na przycisk w menu eclipse. Powinniśmy zobaczyć następujący rezultat:

```
[Lincoln;56]

[Gandhi;79]

----------------

[Gandhi;90]
```

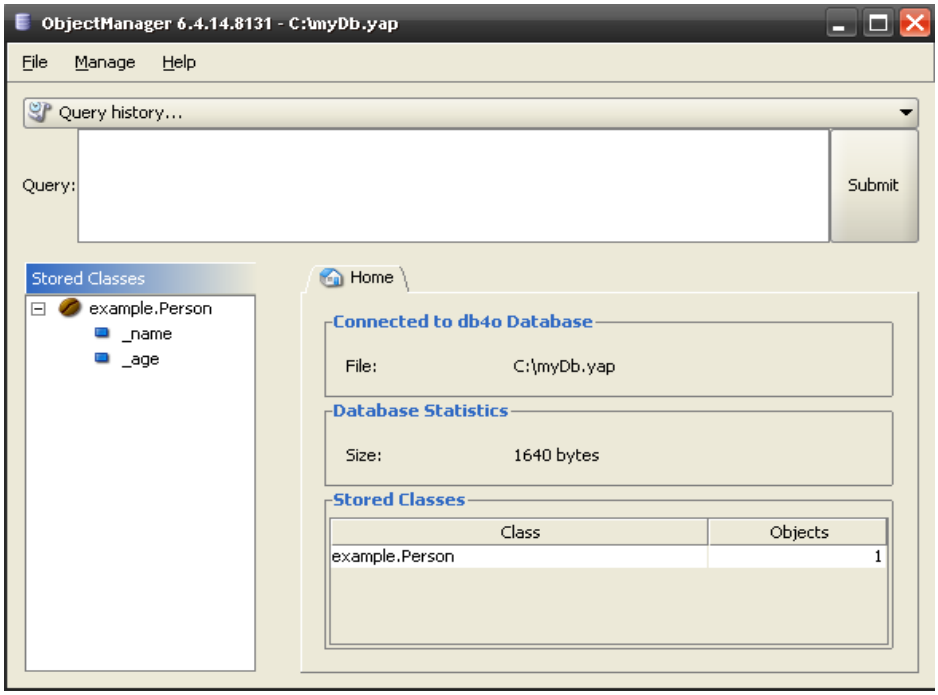
Praca z ObjectManager

Wielu użytkowników baz danych zamiast używać konsoli preferuje korzystać z graficznych narzędzi. Db4o dostarcza takie narzędzie do przeglądania i edytowania danych oraz tworzenia zapytań o nazwie ObjectManager. Różni się on trochę od podobnych narzędzi tego typu dla relacyjnych baz danych, gdyż zarządza się tutaj obiektami, a nie tabelami.

Aby rozpocząć pracę z ObjectManager, rozpakowujemy archiwum objectmanager-6.4.zip do wybranego katalogu np C:\objectmanager, następnie wchodzimy do tego katalogu i uruchamiamy plik objectmanager.bat. Po kilku sekundach pojawia się następujące okno:



Klikamy na przycisk Browse, wskazujemy na ścieżkę C:\myDb\myDb.yap i klikamy na przycisk Open.Maksymalizujemy okno które się wyświetliło.



W górnej części znajduje się okno do tworzenia zapytań, natomiast po lewej stronie znajduje się lista zapisanych w bazie klas. Klikając na pola klas możemy w łatwy sposób tworzyć zapytania.

Przykładowe zapytanie:

```
FROM 'example.Person' WHERE _name = 'Lincoln'
```

Otrzymany rezultat: