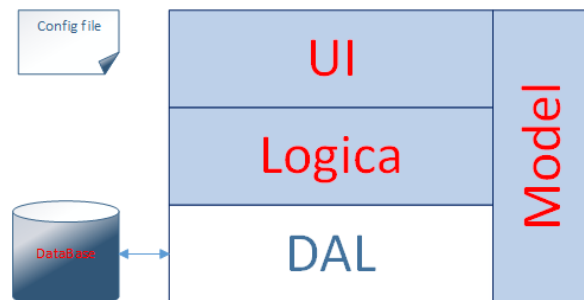


Boekreserveringsysteem

Als oefening voor het Database Project (periode 1.3) en het Applicatiebouw project (periode 1.4) gaan we een reserveringsysteem voor boeken maken. Met dit systeem kunnen (boek) reserveringen van klanten bijgehouden worden. Het systeem wordt opgebouwd volgens dezelfde lagen-architectuur als de 1.3 en 1.4 projecten.



Lagen-Architectuur

Elke laag in de architectuur heeft bepaalde verantwoordelijkheden en bevat een eigen soort klassen. Deze verantwoordelijkheden en klassen worden hieronder beschreven.

Het Model

Deze laag bevat de Model-klassen. Dit zijn de klassen die de dingen in ons systeem representeren. In het boek-reserveringsysteem zijn dat Customer, Book en Reservation. Instanties van deze klassen (objecten) gebruiken we in alle lagen van het systeem. Let op dat een Customer-object altijd een klant representeert.

Let op: Het is een slechte gewoonte om een object slechts gedeeltelijk te vullen met gegevens.

De Data Access Laag (DAL)

De Data Access Layer (DAL) bestaat uit Data Access Objecten (DAO's). De DAL bevat voor elke klasse uit het model een DAO-klasse. Deze klasse is ervoor verantwoordelijk om de data uit de database om te zetten in objecten, en om de objecten op te slaan in de database.

De CustomerDAO is verantwoordelijk voor het omzetten van klant-gegevens uit de database naar Customer-objecten. Dat is het enige wat een DAO doet. De DAO's zijn de enige klassen in het project die SQL en andere database-gerelateerde code bevatten. Een DAO bevat verder geen intelligentie/logica.

Naast de CustomerDAO hebben we in het reserveringsproject ook een BookDAO en een ReservationDAO.

De User Interface Laag (UI)

De UI laag is verantwoordelijk voor het (gedeeltelijk) zichtbaar maken van objecten in de user interface en voor het verwerken van gebruikersinvoer.

De UI laag bevat klassen die een onderdeel van de userinterface representeren, zoals bijvoorbeeld InlogForm, ReserveringenForm en ZoekForm. Ook kan de UI laag hulp-klassen voor UI-elementen bevatten.

Net als de DAO's bevatten de Forms geen intelligentie. Er wordt hooguit gebruikgemaakt van simpele invoer controle.

De Logica Laag

De logica laag bevat het eigenlijke systeem. Het is de plaats waar de business logica wordt uitgevoerd.

De logica laag bevat Service-klassen. Bij eenvoudige systemen zijn de services georganiseerd per model-klasse. We hebben in het reserveringsproject dus een BookService, een CustomerService en een ReservationService.

De ReservationService bevat bijvoorbeeld een methode CreateReservation(Customer customer, Book book). Deze methode controleert via de CustomerDAO en de BookDAO of de klant en het boek bestaan. Wanneer de klant en het boek bestaan, maakt de ReservationService een Reservation-object aan. De ReservationService laat dit Reservation-object door de ReservationDAO wegschrijven.

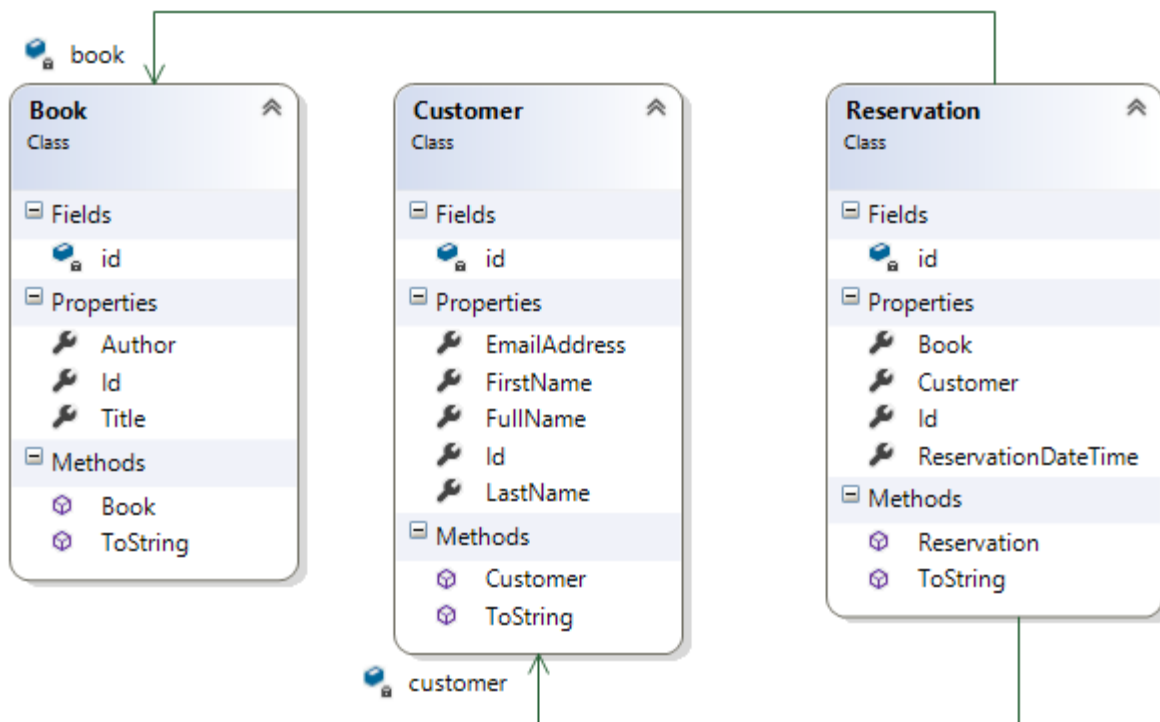
Inleiding Boekreserveringsysteem

We bouwen het boekreserveringsysteem op in een aantal stappen:

1. **Model:** Eerst maken we de modelklassen, die we testen in een console applicatie.
2. **DAL:** Daarna maken we de DAL laag met de DAO-klassen die objecten uit onze database halen. Ook deze laag testen we in onze console applicatie.
3. **Form + Logica:** Voor elke scherm in het UI-Design voegen we een Form toe aan de UI en plaatsen we de benodigde functionaliteiten in de Logica laag. Per functionaliteit testen we de werking van onze WinForm applicatie.

Opdracht 1 – Model klassen

Hieronder staat het classdiagram van de model-klassen van het Boekreserveringsysteem. We gaan deze klassen maken en testen m.b.v. een console application.



Maak een class-library met de naam Model voor de model-klassen. Maak de klassen volgens het klasse-diagram.

Let op de symbolen in het classdiagram: betekent een (public) property, betekent een public methode, betekent een private methode, betekent een private field.

Gebruik de volgende constructors voor de klassen:

```

public Customer(int id, string firstName, string lastName, string emailAddress) { ... }
public Book(int id, string title, string author) { ... }
public Reservation(int id, Customer customer, Book book) { ... }
  
```

Om je model-klassen te testen, maak je nu een Console application, die gebruik maakt van de class-library Model.

Creëer verschillende instanties van Customer, Book en Reservation en toon ze m.b.v. de ToString-methode in de console.

Opdracht 2 – Data Access Laag (DAL)

De Data Access Laag bevat DataAccessObject-klassen (DAO's). Een DAO-klasse heeft als verantwoordelijkheid om de communicatie met de database te verzorgen.

Zo heeft de `public class CustomerDAO` de methode `public List<Customer> GetAll()`. Deze methode levert een lijst met Customer-objecten op. De implementatie van deze methode bepaalt hoe deze lijst wordt gemaakt (gefaket of m.b.v. een query op de database).

Methoden die je typisch in een DAO vindt zijn:

- `GetAll()`. Deze methode doet een "SELECT * FROM Tabel"-query en zet de resultaten van deze query om in een lijst met objecten.
- `GetById(id)`. Deze methode zoekt het record met het opgegeven id op en maakt daar een object van.
- `Create(object)`. Deze methode slaat het opgegeven object op in de database.
- `Update(object)`. Deze methode wijzigt het opgegeven object in de database.
- `Delete(object)`. Deze methode verwijdert het object uit de database.

Houdt deze naamgeving aan; dat maak samenwerken aan dezelfde code makkelijker.

De class-library DAL

- ☐ Voeg je database connectie informatie toe aan de App.config van jouw (Console) project. De gegevens hiervan heb je via e-mail ontvangen (je kunt ook de database van het Database project gebruiken).
- ☐ Maak een nieuwe class-library met de naam DAL. In deze class-library plaatsen we de DAO-klassen.

De database tabel Customers

Voor we de DAO's kunnen implementeren, is het handig om een database te maken en deze te koppelen aan je applicatie.

- ☐ Maak in jouw database een tabel 'Customers' aan met de volgende velden:

```
Id: int
FirstName: nvarchar(100)
LastName: nvarchar(100)
EmailAddress: nvarchar(100)
```

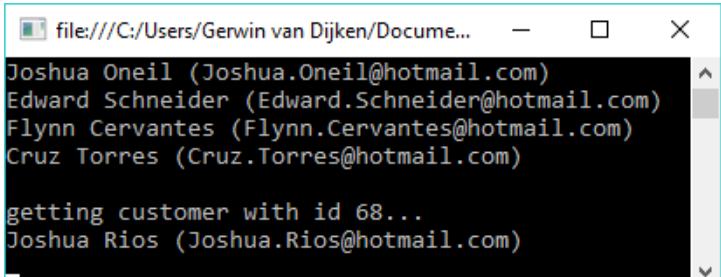
Maak het veld 'Id' de Primary Key en zorg dat deze waarde automatisch verhoogd wordt bij nieuwe records (zet property 'Identity' op True).

- ☐ Vul de tabel 'Customers' met een aantal records.

De klasse CustomerDAO

- ☐ Maak de klasse CustomerDAO in de DAL.
Maak methode `GetAll()` van de CustomerDAO en zorg dat deze een lijst met Customer-objecten teruggeeft. De signatuur van de `GetAll`-methode is: `public List<Customer> GetAll()`
- ☐ Maak ook de methode `public Customer GetById(int customerId)`, die het gevraagde Customer-object maakt op basis van gegevens uit de database.
- ☐ Test de methoden van de CustomerDAO in je Console-applicatie:

```
CustomerDAO customerDAO = new CustomerDAO();
foreach (Customer c in customerDAO.GetAll())
    Console.WriteLine(c);
```



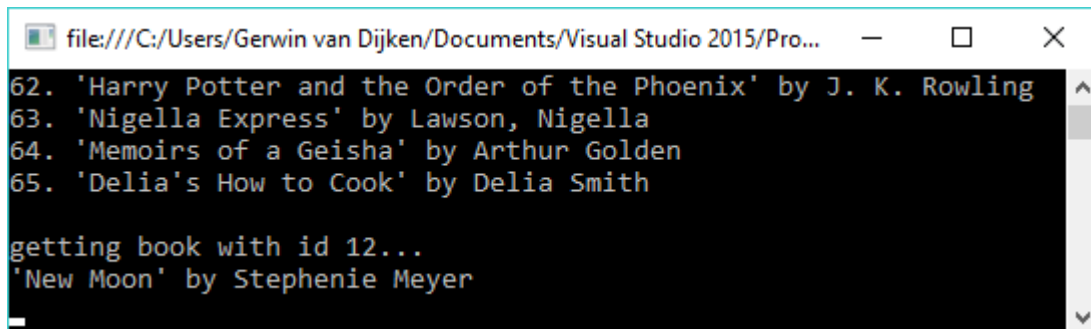
```
file:///C:/Users/Gerwin van Dijken/Docume...
Joshua Oneil (Joshua.Oneil@hotmail.com)
Edward Schneider (Edward.Schneider@hotmail.com)
Flynn Cervantes (Flynn.Cervantes@hotmail.com)
Cruz Torres (Cruz.Torres@hotmail.com)

getting customer with id 68...
Joshua Rios (Joshua.Rios@hotmail.com)
```

De tabel Books en de BookDAO

Net als de klanten moeten ook de (te reserveren) boeken uit de database gelezen worden. Dit is de verantwoordelijkheid van de klasse BookDAO.

- ☐ Maak in jouw database een tabel 'Books' aan met de volgende velden:
 - Id: int
 - Title: nvarchar(200)
 - Author: nvarchar(200)Maak het veld 'Id' de Primary Key en zorg dat deze waarde automatisch verhoogd wordt bij nieuwe records (zet property 'Identity' op True).
- ☐ Vul de tabel 'Books' met een aantal records.
- ☐ Maak een klasse BookDAO in de DAL.
- ☐ Geef de klasse BookDAO een methode GetAll(), die een lijst met Book-objecten uit de database terug geeft.
- ☐ Geef de klasse BookDAO ook een methode GetById(int bookId), die het gevraagde Book-object maakt op basis van gegevens uit de database.
- ☐ Test de methoden van de BookDAO in je Console-applicatie.



```
file:///C:/Users/Gerwin van Dijken/Documents/Visual Studio 2015/Pro...
62. 'Harry Potter and the Order of the Phoenix' by J. K. Rowling
63. 'Nigella Express' by Lawson, Nigella
64. 'Memoirs of a Geisha' by Arthur Golden
65. 'Delia's How to Cook' by Delia Smith

getting book with id 12...
'New Moon' by Stephenie Meyer
```

De tabel Reservations en de ReservationDAO

Net als de klanten en boeken moeten ook de (openstaande) reserveringen uit de database gelezen worden. Dit is de verantwoordelijkheid van de ReservationDAO.

- ☐ Maak in jouw database een tabel 'Reservations' aan met de volgende velden:
 - Id: int
 - CustomerId: int
 - BookId: intMaak het veld 'Id' de Primary Key en zorg dat deze waarde automatisch verhoogd wordt bij nieuwe records (zet property 'Identity' op True). Maak veld 'CustomerId' een Foreign Key naar tabel 'Customers', en veld 'BookId' een Foreign Key naar tabel 'Books'.
- ☐ Vul de tabel 'Reservations' met een aantal records.
- ☐ Maak een klasse ReservationDAO in de DAL.
- ☐ Geef de klasse ReservationDAO een methode GetAll(), die een lijst met alle Reservation-objecten uit de database teruggeeft. Je moet je query zo maken (m.b.v. JOIN) dat je meteen ook de juiste klant en boek gegevens per reservering krijgt.
- ☐ Geef de klasse ReservationDAO ook een methode GetAllForBook(Book book), die een lijst met reserveringen voor het opgegeven boek ophaalt uit de database. Signatuur:
`public List<Customer> getAllForBook(Book book)`
- ☐ Geef de klasse ReservationDAO ook een methode GetAllForCustomer(Customer customer), die een lijst met reserveringen van de opgegeven klant ophaalt uit de database. Signatuur:
`public List<Book> getAllForCustomer(Customer customer)`
- ☐ Test de methoden van de ReservationDAO in je Console-applicatie.