

СУ “Св. Климент Охридски”
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ПРОЕКТ ПО
ASP ПРОГРАМИРАНЕ

Online билетна каса за спортни прояви

Автор:
Емил Станчев,
ф.н.71100

Преподавател:
доц. д-р Павел Павлов

10 юни 2010 г.

Съдържание

1	Задание	2
2	Реализация	2
2.1	База от данни	2
2.2	Приложение	3
2.2.1	Помощни скриптове	4
2.2.2	Обща част	5
2.2.3	Администраторска част	5
2.3	Клиентска част	6

Таблицата **User** представлява абстрактен клас - инстанции на **User** не могат да се създават - само инстанции на „наследниците“ **Administrator** и **Client**. Първичен ключ на тази таблица е **email**. Проверката дали даден потребител е администратор се извършва като се провери колоната **is_admin**. Стойността на **is_admin** се променя чрез тригер, който се активира при прибавянето на нов администратор (ред в таблицата **Administrator**).

Client Клиент на системата. Тази таблица е в отношение 1 към 0..1 с **User**. Поради тази причина първичният ключ на **Client** е външният ключ към таблицата **User** - **User_email**. Всеки клиент има освен „наследените“ атрибути на **User** и телефонен номер (**phone_number**) и външен ключ към адрес **Address**.

Address Таблица с адреси. Тези адреси могат да са както на клиенти, така и на спортни съоръжения. Колоната **city** указва града, а **details** съхранява останалата част от адреса.

Place Таблица със спортни съоръжения. Всяко спортно съоръжение има име (**name**) и външен ключ към адрес (**Address_id**), както и кратко текстово описание. Първичен ключ е **name**.

Event Спортна проява. Всяка спортна проява се случва на дадено спортно съоръжение (външен ключ **Place_name**), и включва даден вид спорт (външен ключ **Sport_name**). Всяка проява има начален час (**start**) и продължителност (**duration**). Ако продължителността не е зададена, стойността по подразбиране е NULL. Ако стойността е NULL, за продължителност на събитието се счита типичната продължителност на асоциирания спорт (**typical_duration** колоната на таблицата **Sport**). Първичният ключ е сурогатен (**id**).

Sport Даден вид спорт. Ключ е името (**name**). **typical_duration** е типичната продължителност за даден вид спорт (например, 90 минути за футболен мач); тази стойност се използва за продължителност на спортната проява, ако такава не е зададена.

TicketCategory Категория билети. Всяка спортна проява има една или повече категории билети, всяка от които има име (**name**), дадена цена (**price**), и налична бройка билети от тази категория (**count**). Например, можем да имаме категория с име „Сектор А“ с цена на билета 12лв. и налични 5000 билета.

Cart Таблица, съхраняваща количката на клиент. Всяка количка е асоциирана с точно един клиент (**Client_User_email**).

CartItem Таблица, съдържаща информация за артикулите в количката на даден клиент. Всеки артикул представлява един или повече билети от дадена категория за дадена спортна проява (външен ключ **TicketCategory_name**). Всеки артикул има и съответна бройка билети (**count**). Например, „3 билета за сектор А на мача Левски - Славия“.

Order Представлява завършена поръчка. Състои се от артикули (**OrderItem**), и има асоцииран куриер (**Courier**).

OrderItem Представлява артикул от завършена поръчка. Аналогично с артикул в количката.

Courier Всеки куриер има име (**name**) и **calculate_url**, което е хипервръзка към сайта на куриера за изчисляване на цената. Тази връзка се появява при избора на куриер от клиента.

2.2 Приложение

Скриптовете на приложението са реализирани с *JScript*, който трябва да се конфигурира като скриптов език по подразбиране на сървъра, защото **не е указано**

```
<% @LANGUAGE=JScript %>
```

изрично във всеки ASP файл. Списъкът с файлове в проекта и тяхното предназначение е показан по-долу. Повечето от скриптовете са реализирани така, че при *GET* заявка те показват страница, която обикновено включва форма за попълване, а при *POST* заявка изпълняват съответното действие. Например, **admin/event.asp** показва нова форма на администратора при *GET* заявка, а при *POST* заявка от администратора извършва запис в базата данни след съответната валидация.

2.2.1 Помощни скриптове

db/db.inc Файл, съдържащ Javascript-класове за базата от данни. Създава се един метаклас, представляващ таблица в база от данни, след което от него се създават класове за всяка таблица. В резултат имаме **следните класове**:

- User
- Place
- Sport, и т.н.

Всеки от тези класове играе ролята на *модел* (в смисъла, определен от MVC архитектурата). Тоест всеки клас представлява интерфейс от високо ниво към базата от данни. Всяка колона на таблица от базата данни съответства на поле на обект, т.е. реализиран е прост ORM (Object-Relational Mapper). Някои от методите са изброени по-долу. Model представлява произволен клас, представляващ таблица от базата данни. model_obj представлява инстанция на Model.

model_obj.save() Запазва в базата от данни запис, съответстващ на user_obj. Ако user_obj представлява съществуващ запис, то се извършва UPDATE заявка, в противен случай се извършва INSERT. Тук се извършва и всичката валидация на данните, преди да се запази.

Model.primary_keys Статичен атрибут, съдържа списък от първичните ключове. Използва се от save метода, за да се определи дали обектът съществува като ред в таблицата, за да се направи съответно UPDATE или INSERT заявка.

model_obj.delete() Изпълнява DELETE заявка.

Model.all(filter, opts) Статичен метод, който връща потребители като масив от обекти на класа User. filter е асоциативен масив, който указва какво да се включи WHERE клаузата. opts е асоциативен масив, който съдържа други опции (например, ако opts['or'] е true, в WHERE клаузата се използва OR вместо AND. Например, ако Model=User

```
User.all({'email': 'ivan@mail.bg', 'email': 'georgi@mail.bg'},
{'or': true});
```

изпълнява заявката:

```
select * from "User"
where email='ivan@mail.bg' OR email='georgi@mail.bg';
```

Model.count(filter, opts) Статичен метод, връща брой записи.

Model.find(where_string) Статичен метод, който изпълнява следната заявка

```
select * from "Model" where <where_string>;
```

Model.exists(attributes) Проверява дали съществува ред с дадените атрибути.

Горните методи се поддържат от всички класове, съответстващи на таблици от базата данни, тъй като тези класове са създадени от един общ метаклас.

Освен това този файл съдържа клас DBConnection, който енкапсулира връзката с базата от данни. Например:

```
dbc = new DBConnection('PostgreSQL35W',
'SET search_path TO tickets;');
dbc.execute('select * from "Event"');
dbc.table_names();
```

В горния пример създаваме нов обект, представляващ връзка към базата данни. Първият аргумент на конструктора е dsn идентификатора на базата от данни, който е зададен в Data Sources настройките на сървъра. Вторият аргумент е заявка, която да се изпълни при стартирането на връзката, в този случай пътя за търсене става схемата tickets.

Методът execute просто делегира към съответния метод на ADOConnection, но връща масив от обекти вместо recordset.

Методът table_names връща имената на таблиците в базата от данни.

models.inc Създава инстанции на моделите за всяка от таблиците в базата данни и инициализира връзка с базата от данни:

Файл 1: models.inc

```
1 <!-- #include FILE="db.inc" -->
2 <%
3     db                = new DB.Connection("PostgreSQL35W",
4                                           "SET search_path TO tickets;");
5     Cart               = db.model("Cart");
6     CartItem           = db.model("CartItem");
7     Order              = db.model("Order");
8     Courier             = db.model("Courier");
9     User               = db.model("User");
10    Administrator      = db.model("Administrator");
11    Sport               = db.model("Sport");
12    Event               = db.model("Event");
13    Client              = db.model("Client");
14    TicketCategory      = db.model("TicketCategory");
15    Place               = db.model("Place");
16    Address             = db.model("Address");
17    OrderItem           = db.model("OrderItem");
18 %>
```

util.inc Помощни функции за работа с масиви, стрингове и други. Например:

function map(array, lambda) Връща нов масив, който е резултат от прилагането на функцията **lambda** върху всеки елемент на масива **array**.

function filter(array, lambda) Връща нов масив, който се състои само от елементите на **array**, за които **lambda** връща **true**-стойност.

md5.inc Файл, в който е дефинирана функция, която хешира с *MD5* алгоритъм. Използва се при работа с пароли.

2.2.2 Обща част

default.asp Началната страница на приложението. Показва се списък от предстоящи прояви, за които има налични билети.

login.asp Това е страницата за вписване.

logout.asp Изчиства се сесията и потребителят се отписва (ако не е вписан, се пренасочва към **login.asp**).

search.asp?from=<fd>&to=<td>&sport=<s>&city=<c> Търсене на проява по различни критерии.

2.2.3 Администраторска част

admin/admin_required.inc Скриптът в този файл проверява дали текущият потребител е администратор (използвайки атрибута **is_admin** на **User**). Ако не е администратор, то той се пренасочва към страницата за вписване **login.asp**, ако не е вписан; ако пък е вписан, се пренасочва към началната страница **default.asp**. Този файл се включва с **#include** директива във всички файлове от директорията **admin/**.

admin/default.asp Началната страница на администраторската част. Показва се списък от прояви, потребители и настройки. Администраторът получава възможност да редактира всички обекти, които се показват.

admin/event.asp?id=<id> Страница за редактиране на проява. Показва се форма за въвеждане на данни за проявата с **id=<id>** и категория билети и цени. Ако *HTTP* параметърът **id** не е зададен, се отваря форма за въвеждане на нова проява.

admin/place.asp?name=<name> Страница за редактиране на спортни съоръжения. Показва се форма за редактиране на мястото с **name=<name>**. Ако *HTTP* параметърът **name** не е зададен, се отваря форма за въвеждане на ново спортно съоръжение.

`admin/client.asp?email=<email>` Страница за редактиране на клиента, за който `email=<email>`.
Всичката информация за клиента е налична и разрешена за редактиране.

2.3 Клиентска част

`client_required.inc` Проверява дали потребителят е клиент. Ако не е и е вписан като администратор, се пренасочва към `admin/default.asp`. Ако пък въобще не е вписан, се пренасочва към `default.asp`.

`profile.asp` Редакция на потребителския профил.

`cart.asp` Количката на потребителя. При POST заявка се прибавя нов артикул.

`order.asp` Завършване на поръчката след избор на куриер и преглед на артикулите.

`event.asp?id=<id>` Подробности за дадено събитие.

`history.asp` История на поръчките на клиента.