МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт Компьютерных технологий

и информационной безопасности

**Кафедра Математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

**О преддипломной практике**

в Лаборатории мобильной и веб-разработки SCINO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студента 3 курса  Очного отделения  Направления 02.03.03 «Программная инженерия»  ***Борисова Дмитрия Александровича*** | | | | | |
|  |  |  |  |  |  |
|  | *дата* |  | *подпись* |  | *Фамилия И.О.* |
| Руководитель от кафедры:  доцент кафедры МОП ЭВМ ***Фёдоров Фёдор Федорович*** | | | | | |
|  |  |  |  |  |  |
|  | *дата* |  | *подпись* |  | *Фамилия И.О.* |

Таганрог 2017 г.

**ОТЗЫВ-ХАРАКТЕРИСТИКА**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент ***3*** курса кафедры Математического обеспечения и применения ЭВМ Института компьютерных технологий и информационной безопасности Инженерно-технологической академии Южного федерального университета | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ***Борисов Дмитрий Александрович*** | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| с « | | ***6*** | | » | | ***февраля*** | 2017г. по « | | | ***19*** | » | ***февраля*** | | | | | | 2017г. | | | | | прошёл практику в | | | | |
| Лаборатории мобильной и веб разработки SCINO | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| В период практики выполнял обязанности про*граммиста* | | | | | | | | | | | | | | |  | | | | | | | | | | | | |
| За время прохождения практики ***Борисов Дмитрий Александрович*** показал; **высокий/средний/низкий** уровень теоретической подготовки;  **высокую/среднюю/низкую** степень умения и навыков применять и использовать знания, полученные в университете, для решения поставленных перед ним (ней) практических задач.  Студентом выполнены следующие виды работ: фреймворк ASP.NET MVC 4, LINQ to SQL, Twitter Bootstrap 3.0, JQuery; разработана система ведения блогов на ASP.NET MVC 4.  Студентом проявлены следующие личностные и профессиональные качества: дисциплинированность, усидчивость, умение работать в коллективе, вежливость, тактичность.  Проявленные профессиональные качества удовлетворяют потребностям предприятия **полностью/частично**.  Программа практики выполнена **полностью/частично**.  Сроки выполнения заданий соблюдались **полностью/частично**. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Руководитель практики в SCINO:**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Должность и ФИО | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Рекомендуемая оценка** | | | | | | | |  | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | |  |  | | | | | | |  |  | | | | | |
|  | | | | | | | | *дата* | | | | |  | *подпись* | | | | | | |  | *Фамилия И.О.* | | | | | |
| **Руководитель SCINO:**  Доцент кафедры МОП ЭВМ, Хашковский Валерий Валерьевич | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | |  | | | | | | |  |  | | | | | | |  |  | |
|  | | | | | | | | | *дата* | | | | | | |  | *подпись* | | | | | | |  | *Фамилия И.О.* | |
| **«Согласовано»**  **Руководитель от кафедры** | | | | | | | | |  | | | | | | | | | |  |  | | | | | |
|  | | | | | | | | | *подпись* | | | | | | | | | |  | *Фамилия И.О.* | | | | | |
| « |  | | » | |  | | | | 2017 г. | | | | | | | | | |  |  | | | | | |

**ДНЕВНИК ПРЕДДИПЛОМНОЙ ПРАКТИКИ**

**1. Общие сведения**

1. Фамилия: ***Борисов***
2. Имя: ***Дмитрий***
3. Отчество: ***Александрович***
4. Направление *02.03.03* «*Программная инженерия*»
5. Предприятие: *НОЦ* «*Математическое и программное обеспечение информационных систем реального времени*» *ЮФУ*
6. Руководитель практики: *доцент кафедры МОП ЭВМ* ***Фёдоров Ф.Ф.***
7. Срок практики по учебному плану: *с* «*6*» *февраля 2017. по* «*19*»*февраля 2017.*
8. Тема: «*Разработка системы блога на ASP.NET MVC 4*»

**2. Индивидуальное задание студента по преддипломной практике**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Содержание работы** | **Сроки выполнения** | **Форма отчетности** |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Студент** | | | |  |  |  |
|  | | | | *подпись* |  | *Фамилия И.О.* |
|  | | | | | | |
| **«Согласовано»**  **Руководитель от кафедры** | | | |  |  |  |
|  | | | | *подпись* |  | *Фамилия И.О.* |
| « |  | » |  | 2017 г. |  |  |

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

MVC – Model-View-Controller, название паттерна, лежащего в основе технологии ASP.NET MVC .

ASP.NET – Active Server Page on .NET, технология создания веб-приложений и веб-сервисов от Microsoft на платформе .NET.

Model – модель, описывает собой логику, и структуру данных.

Controller – класс, обеспечивающий инкапсуляцию логики приложения.

View – представление, пользовательский интерфейс, отвечает за показ и прием данных от пользователя.

ViewModel – разновидность моделей, которые представляют данные конкретного представления.

Hashtag – тип пометки, используемый в микроблогах или социальных сетях, обеспечивающий поиск сообщений по теме или содержанию.

DAL (Data Access Layer) – код, служащий для предоставления доступа к базе данных с помощью объектов.

md5 – алгоритм шифрования.

Unicode – способ кодировки текста, в котором один символ занимает два байта

Id – идентификационный номер.

XSS-атака - тип атаки на веб-системы, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы).

ВВЕДЕНИЕ

ASP.NET MVC – фреймворк для создания веб-приложений, который реализует шаблон Model-view-controller.

Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления. Контроллер принимает запросы, обрабатывает пользовательский ввод, взаимодействует с моделью и представлением и возвращает пользователю результат обработки запроса.

Модель представляет слой, описывающий логику организации данных в приложении. Представление получает данные из контроллера и генерирует элементы пользовательского интерфейса для отображения информации.

Для управления разметкой и вставками кода в представлении используется движок представлений. До версии MVC 5 использовались два движка:

Web Forms и Razor. Начиная с MVC 5 единственным движком, встроенным по умолчанию, является Razor. Движок WebForms использует файлы .aspx, а Razor — файлы .cshtml и .vbhtml для хранения кода представлений. Основой синтаксиса Razor является знак @, после которого осуществляется переход к коду на языках C#/VB.NET.

При обработке запросов фреймворк ASP.NET MVC опирается на систему маршрутизации, которая сопоставляет все входящие запросы с определенными в системе маршрутами, которые указывают какой контроллер и метод должен обработать данный запрос. Встроенный маршрут по умолчанию предполагает трехзвенную структуру: контроллер / действие / параметр.

ASP.NET MVC и ASP.NET Web Forms являются двумя родственными технологиями, в основании которых лежит одна платформа ASP.NET. И все же ASP.NET MVC имеет ряд преимуществ перед ASP.NET Web Forms:

а) разделение ответственности (отдельная разработка разных компонентов — контроллера, моделей, представлений);

б) улучшенная тестируемость;

с) соответствие протоколу HTTP (ASP.NET MVC не поддерживает объект ViewState);

г) повышенная гибкость и настраиваемость под собственные нужды;

В рамках учебного проекта передо мной стояла задача разработать блог., в котором нужно было реализовать: ленту, рейтинг статей, просмотр статьи, систему комментариев, возможность для автора редактирования своей статьи. В дополнение к этому было добавлено требование создания систем аутентификации и авторизации пользователей, а также возможности добавления к статьям хештегов, а также применение этой системы в виде выбора подобных статей в зависимости от тега.

Разработка велась в Microsoft Visual Studio 2015.

В качестве DAL была выбрана LINQ-to-SQL.

Front-end был написан на Twitter Bootstrap 3.0 + JQuery.

ОСНОВНАЯ ЧАСТЬ

Моей первостепенной задачей в данной работе являлась разработка базы данных, удовлетворяющей всем задачам проекта, так как я являюсь сторонником подхода Database-First.

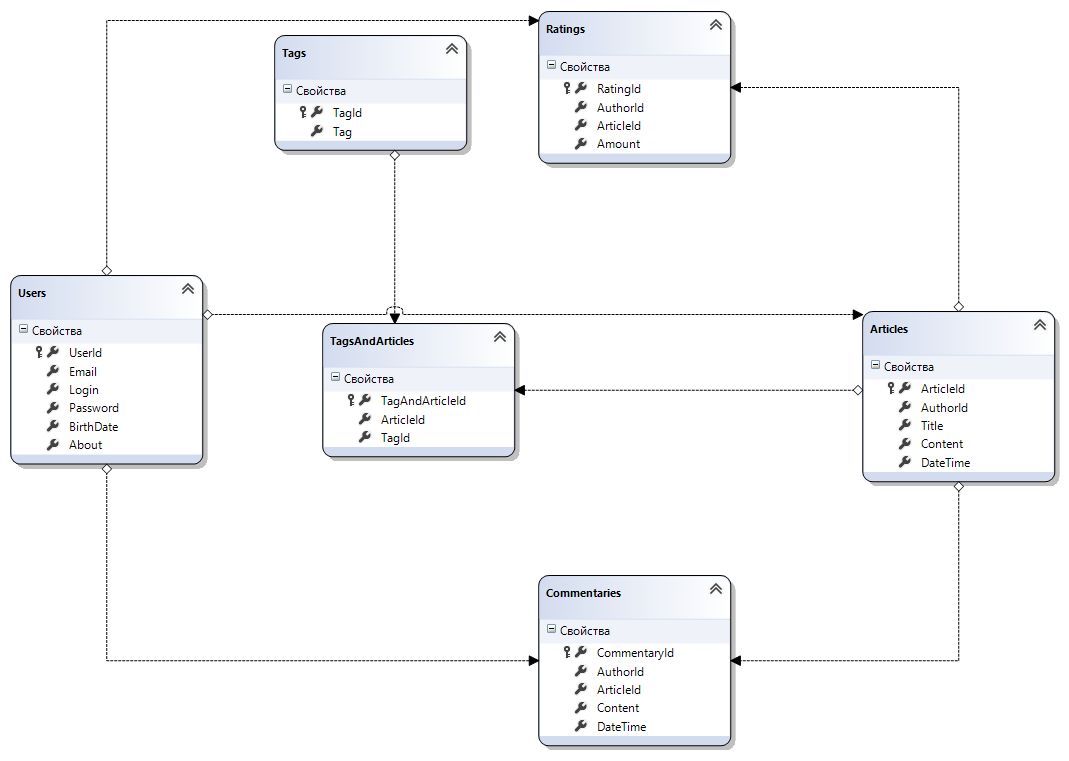
Результатом стала база данных под названием BlogDB, имеющая 6 таблиц. Она представлена на рисунке 1.

Рисунок 1. Общая модель базы данных

Поскольку я планировал сделать проект максимально модульным и наиболее простым для модификаций, пришлось создавать базу данных, удовлетворяющую этим требованиям.

Представление всех таблиц можно увидеть рисунках 2(Users), 3(Articles), 4(Commentaries), 5(Ratings), 6(Tags), 7(TagsAndArticles).

Подробнее о таблицах:

Users. Первичным ключем этой таблицы является поле UserId. Все первичные ключи таблицы имеют специализацию идентификатора, начиная отсчет с единицы, и используя такой же шаг нумерации. При разработке модели блога я столкнулся с проблемой: что использовать в качестве идентификации пользователя: адрес электронной почты, логин или и то, и другое? Конечным решением стал последний вариант: он показался наиболее предпочтительным из-за своей простоты, а также необходимостью обоих компонентов для работы как систем авторизации/аутентификации, так и межпользовательского взаимодействия. Таким образом, поле Login служит для идентификации пользователя и авторизации, а поле Email может использоваться в будущем для самых разнообразных целей: начиная от восстановления пароля путем отправки сообщения на e-mail и до рассылки новостей и уведомление о новых сообщениях. В рамках учебного проекта поле пароль не шифруется и хранится в исходном виде, несмотря на то, что такое решение – огромная угроза безопасности, и стоило бы хранить хотя бы в виде двойных md5-хешей. Поле BirthDate было введено в целях хранения информации о возрасте пользователя, никакой ценности с точки зрения программы оно не несет, но может быть использовано как основа для системы ограничения доступности контента по возрасту. Одной из причин добавления ViewModel’а с названием RegistrationViewModel было поле About. С ним возникло сразу несколько вопросов, например: а нужно ли оно вообще? А должно ли оно быть необходимым для заполнения? А сколько нужно выделить под него места? Должно ли оно поддерживать русские буквы и/или специальные символы, доступные только в Unicode? Отвечая на эти вопросы, я пришел к выводу что для формы регистрации пользователя оно не несет никакого смысла, т.к. не имеет критически важной информации, создавая лишь неудобства и занимая дополнительно место на форме. Количество места я выделил исходя из того, что это значение должно быть от 200 символов (некоторые люди любят много писать о себе), но в то же время быть меньше, чем четверть выделенного для блога места, т.е. верхней границей являлось число 400. Найдя среднее арифметическое, получил результат в 300 символов, который и использовал. Что касательно использования Unicode – он необходим, т.к. я ориентирую на русскоязычную аудиторию, и мне необходимы русские буквы.

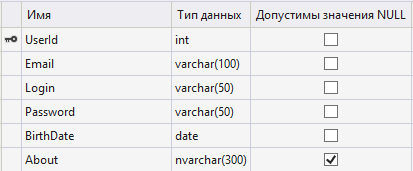


Рисунок 2. Таблица Users

Articles. Для этой таблицы был использован внешний ключ на таблицу пользователей, и полем с таким ограничением стало AuthorId. Из основных данных о записи в блоге стали поля Title, Content и DateTime. Поле Title используется для заголовка. Оно не допускает разметки (что такое разметка относительно данного проекта и остальная информация – ниже). Поле Context поддерживает использование допустимых(!) тегов и служит для хранения текста самой статьи. Поле DateTime хранит дату и время создания записи в блоге.



Рисунок 3. Таблица Articles

Commentaries. Эта таблица хранит комментарии пользователей. Она имеет два внешних ключа, которые указываю id статьи, к которой был создан комментарий, а также его автора. Такой подход (хранение ссылки на автора) исключает возможность анонимного комментирования. Такая проблема поначалу меня не устраивала, поскольку планировал создать сайт максимально удобный для модификаций, а потом я пришел к выводу, что отсутствие анонимности – не так уж и плохо, и это не недостаток, а особенность. К тому же, анонимные комментарии легко добавив, введя в таблицу поле “Тип” (возможно, лучше в другую вынести ссылку на пользователя, но это уже решится при разработке анонимных комментариев). Остальные два поля – Content и DateTime не содержат ничего необычного: первое – текст фразы пользователя, а второе – время создания, причем комментарий не содержит возможность использования тегов.

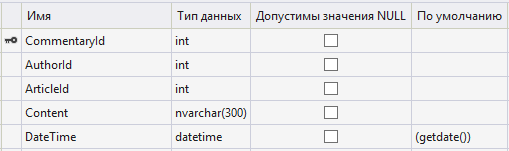


Рисунок 4. Таблица Commentaries

Ratings. Данная таблица содержит оценку (поле Amount) пользователя(AuthorId) статьи (ArticleId). При этом не стоит путать: в текущем контексте рассматривается автор не статьи, а комментария! Для поля Amount выбран тип int, несмотря на то, что оно может принимать только два значения: +1 и -1, т.к. при необходимости перехода на другую систему оценивания, например, десятибалльную, не нужно будет исправлять тип объекта в БД.

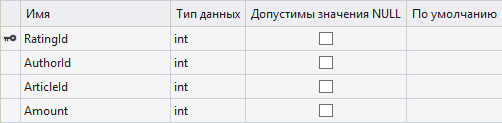


Рисунок 5. Таблица Ratings

Tags. Эта таблица хранит в себе текст хештега. В ней не хранится данных о том, сколько раз был использован тег, а также в каких статьях, так как первое будет неверное как с точки зрения разделения логики таблиц (тег и не должен хранить такую информацию, к тому же динамическую), а для второго была введена таблица TagsAndArticles. Она используется для избавления от связи типа “многие-ко-многим” и содержит два внешних ключа, на статью и на тег, соответственно.

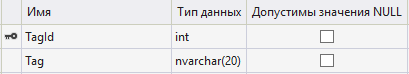


Рисунок 6. Таблица Tags

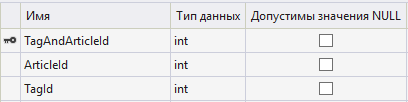


Рисунок 7. Таблица TagsAndArticles

После базы данных были сгенерированы классы LINQ to SQL, и модель данных была описана, но несмотря на это, были введены некоторые ViewModel’и, а именно:

BlogViewModel. Эта модель используется для создания и редактирования пользователем самой записи в блоге. Её особенностью является то, что её содержимое проверяется не фреймворком, а вручную. Такие меры были предприняты для того, чтобы пользователь мог использовать либо BB-коды, либо HTML-теги. Я остановился на втором, но тут возникла другая проблема, а именно возможность XSS-атак. Решением стала фильтрация вводимых пользователем данных и удаление потенциально небезопасных конструкций.

CommentaryViewModel. Содержит лишь строчку комментария, так как все остальные данные берутся либо из сессии, либо из БД.

LoginViewModel. Имеет строку логина и строку пароля для аутентификации пользователя.

RegistrationViewModel. Содержит все поля как для элемента таблицы Users, а также поле повторения пароля. На сходство введенные данные проверяются автоматически посредством атрибутов языка C# и их обработкой в фреймворке.

Теперь очередь контроллеров и представлений. Все в проекте контроллеров четыре:

1. HomeController. Такой контроллер обязан присутствовать по стандарту, поэтому он и добавлен. Отвечает за вывод представления Index, которое представляет собой главную страницу.
2. AccountController. Необходим для авторизации пользователей. Не стоит путать с UserController, так как последний отвечает работу с моделью пользователя с точки зрения логики сайта. AccountController содержит два представления: Login и Register.
3. UserController. Отвечает за взаимодействие с данными пользователей. Имеет три представления: Index (выводит список пользователей), Info (выводит информацию о пользователе) и UserNotFound (выводящую сообщение о том, что пользователь не найден в системе.
4. BlogController. Служит для всевозможных взаимодействий с блогами. Имеет наибольшее количество представлений, а именно десять: Feed (показывает все блоги в обратной хронологической последовательности), Create (создание записи в блог), Modify (изменение блога), Best (сортирует и показывает записи блогов по рейтингу), AddComment (отвечает за добавление комментария), Rate (выставление оценки пользователем блогу), Tag (вывод записей блогов по тегу), ViewArticle (страница вывода содержимого записи в блог), SuccessBlogCreate (сообщение об успешном создании/изменении блога) и WrongId (вызывается при вводе неправильного id’a).

Теперь об особенностях некоторых компонентов проекта и проблемах, с которыми я столкнулся во время реализации:

1. Запускать для тестирования и отладки проекты нужно в Debug’e, а не в Release’e. При компиляции в режим отладки компилятор создает дополнительную информацию, работа без которой иногда (как в моем случае) может повлечь абсолютно нелогичные ошибки по типу того, что объект ViewBag не создается. Это обеспечило десять часов отладки/изучения интернета и книг/изучения System.Web.Mvc Reflector’ом в бесплодных поисках причины ошибки
2. Фреймворк заботится о безопасности вводимых пользователем данных, и иногда даже излишне: пришлось поискать информацию о том, как отключить безопасность для одного компонента, а именно – контента статьи. Я решил разрешить пользователю использовать HTML, но лишь безопасный, без скриптовых вставок. Поэтому пришлось делать вручную проверку введенных данных, а также исключение нежелательных элементов.
3. Во время реализации системы авторизации пользователей познакомился с частичными представлениями (увидел это в стандартных шаблонах для MVC, идущих по умолчанию с установкой веб-инструментов для разработки). Удобно было выделить блок “Регистрация-Вход” и аналогичный, но для уже авторизованного пользователя.

ЗАКЛЮЧЕНИЕ

Во время практики я овладел основами веб-разработки с использованием ASP.NET MVC 4, а также LINQ to SQL, Twitter Bootstrap и т.д.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Герберт Шилдт “C# 4.0. Полное руководство”
2. Джеффри Рихтер “CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#”
3. Адам Фримен “ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов”