Приложение Art Museum

ASP.NET MVC 4.0

2013

Цель

1. Разработать модель и базовую функциональность ASP.NET MVC приложения.

2. Ознакомиться с управлением генерацией базы данных.

Художественный музей - ArtMuseum

- Есть музей с залами и картинами в залах. Число залов, как и число картин, не фиксировано. О залах известно: название. О картинах – название, автор, год.

- Посетители посещают залы, осматривают экспозицию и пишут отзывы о картинах. Могут выполнять поиск картины по фрагменту названия, фамилии художника, диапазону лет и любой комбинации этих признаков. Чтобы писать отзывы, посетитель должен пройти процедуру аутентификации, а если надо, то и регистрации.

- Администраторы (зарегистрированные пользователи) манипулируют залами и картинами, модерируют отзывы посетителей.

Другие темы для разработки

Фотоальбом

- Каждый зарегистрированный пользователь может размещать фотографии в своем альбоме, создавать подальбомы, приглашать друзей, давать к отдельным фотографиям и подальбомам различные уровни доступа: открытый – для всех, закрытый – для себя и защищенный - для друзей.

Аукцион

- Есть два типа пользователей – администратор и покупатели. Администратор выставляет вещи для продажи, назначает первоначальную цену и подтверждает покупку. Покупатель просматривает вещи, торгуется и получает извещение о том, что вещь продана ему или не ему.

Книжный магазин

- Есть два типа пользователей – администратор и покупатели. Администратор выкладывает книги, разделяя их по жанрам. Покупатель осуществляет поиск книги, кладет ее в корзину и подтверждает покупку. Покупатель может оставлять отзывы о ранее купленных книгах и читать отзывы других покупателей.

Сайт голосования

- Администратор выкладывает формы для голосования, и просматривает результаты. Форма представляет собой вопрос с несколькими предопределенными ответами. Голосующий просматривает открытые формы и выбирает ответ, за который он голосует. Он может видеть результаты тех голосований, в которых принял участие.

- **Простая социальная сеть.** Регистрация пользователей. Настройка персональной информации. «Друзья» пользователя. Обмен сообщений с другими пользователями. Поиск пользователей по определенным критериям. Управление пользователями. Модерирование сообщений.[1]
- **Интернет-фотоальбом.** Регистрация, загрузка фотографий. Возможность просматривать и оценивать фотографии других пользователей. Поиск фотографий. Управление пользователями.
- **Персональный блог.** Регистрация пользователей. Создание блога (блогов). Создание и редактирование статей блога. Тэги статей. Поиск по тэгам, тексту. Комментирование статей. Управление пользователями. Модерирование статей и комментариев.
- **Интернет-аукцион.** Просмотр, поиск, и «покупка» лотов. Регистрация и выставление лотов на аукцион. Управление пользователями и модерирование списка лотов.
- **Система тестирования знаний.** Регистрация, выбор (поиск) теста. Прохождение теста с контролем времени. Статистика тестирования. Управление пользователями. Редактирование тестов. Расширенная статистика тестирования.
- **Файловое хранилище.** Доступ к файлу по короткой ссылке. Общие файлы и файлы с ограниченным доступом. Для зарегистрированных пользователей – возможность размещения файлов и управление файлами. Управление пользователями и их файлами. Поиск файлов.
- **Картотека текстовых материалов (вариант: аудио или видео).** Поиск в картотеке по различным критериями. Оценка материалов пользователями.
- **Система учета знаний.** Зарегистрированные пользователи («программисты») указывают и оценивают свои знания в различных (сгруппированных) областях. Пользователь - «менеджер» осуществляет отбор программистов по заданным критериям. Возможность генерирования отчетов. Управление пользователями и ролями. Управления списками областей знаний.
- **Система отслеживания заданий.** Выдача задания менеджером. Статус задания, согласно рабочему процессу. Процент выполнения. Почтовые уведомления клиентам системы. Управление пользователями и их ролями.
- **Форум.** Стандартные операции, присущие любому форуму – добавление тем, сообщений. Модерирование записей. Работа с пользователями форума.

Порядок работы над проектом

1. Сформулировать требования к приложению.
2. Разработать пользовательский интерфейс и карту сайта. Сделать эскизы страниц.
3. Разработать модель данных (классы).
4. Наращивать функциональность приложения в таком порядке: модель, контроллер, представление.

ArtMuseum: Требования

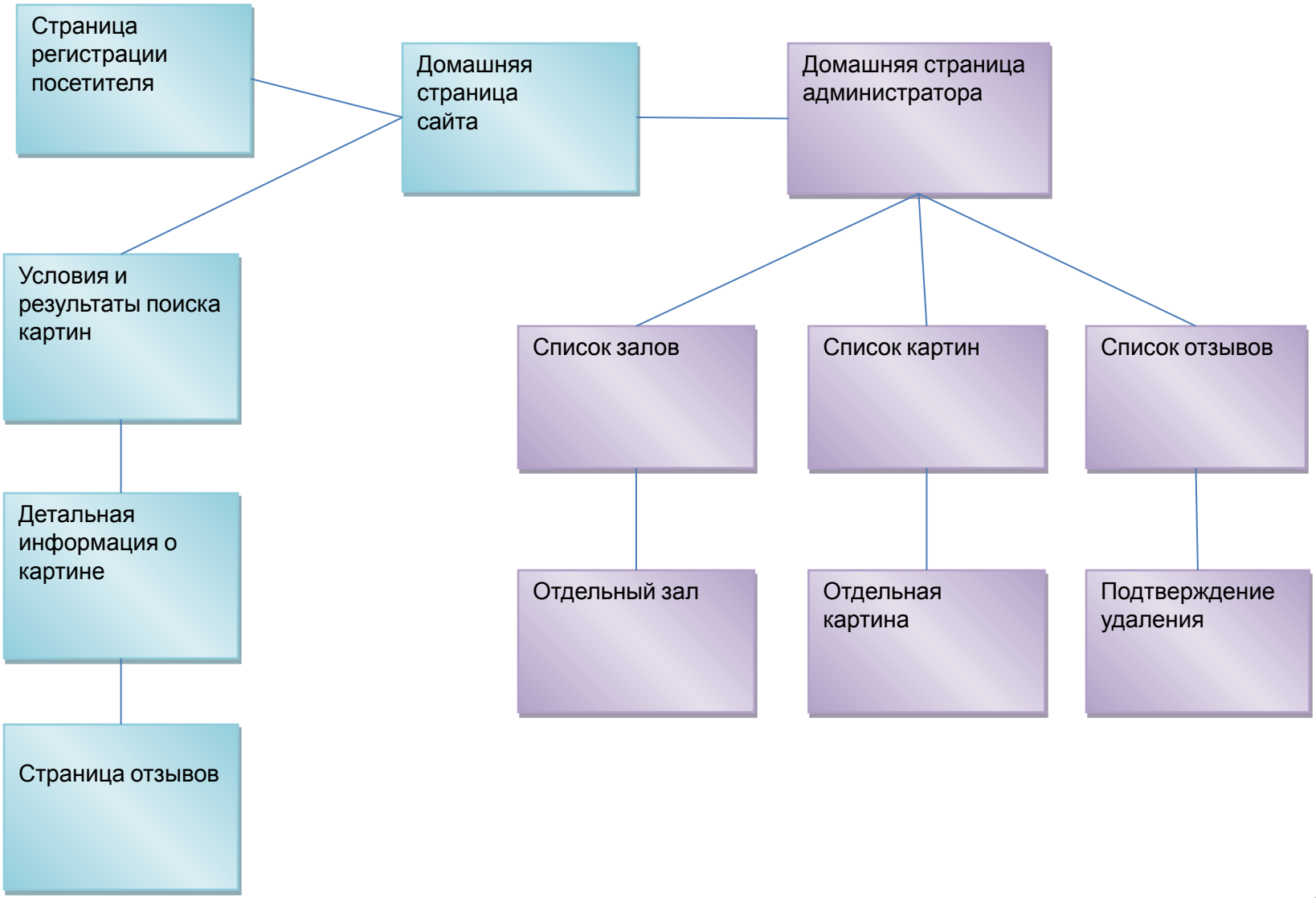
Требования для посетителя

- Посетитель вводит признаки картины и получает список картин, удовлетворяющих введенным признакам (зал, название, автор, год).
- Посетитель выбирает картину из списка и получает полную информацию о картине (изображение, описание, отзывы).
- Зарегистрированный посетитель пишет свой отзыв на выбранную картину.
- Посетитель может уничтожить свой собственный отзыв на картину.

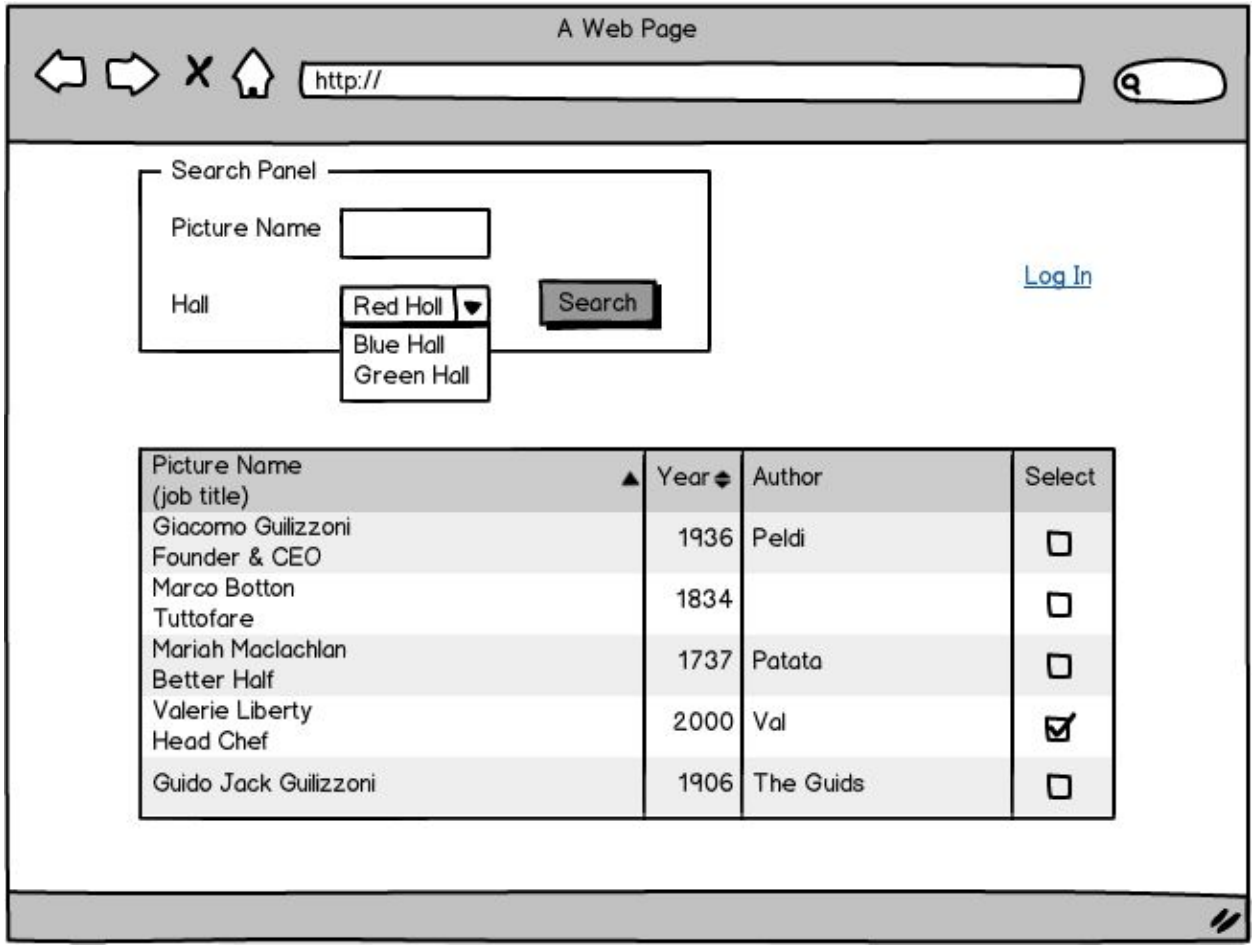
Требования для администратора

- Администратор может добавлять изменять, удалять залы. Удалять можно только пустой зал.
- Администратор может добавлять, изменять, удалять картины.
- Администратор может удалять отзывы посетителей.

ArtMuseum: Карта сайта

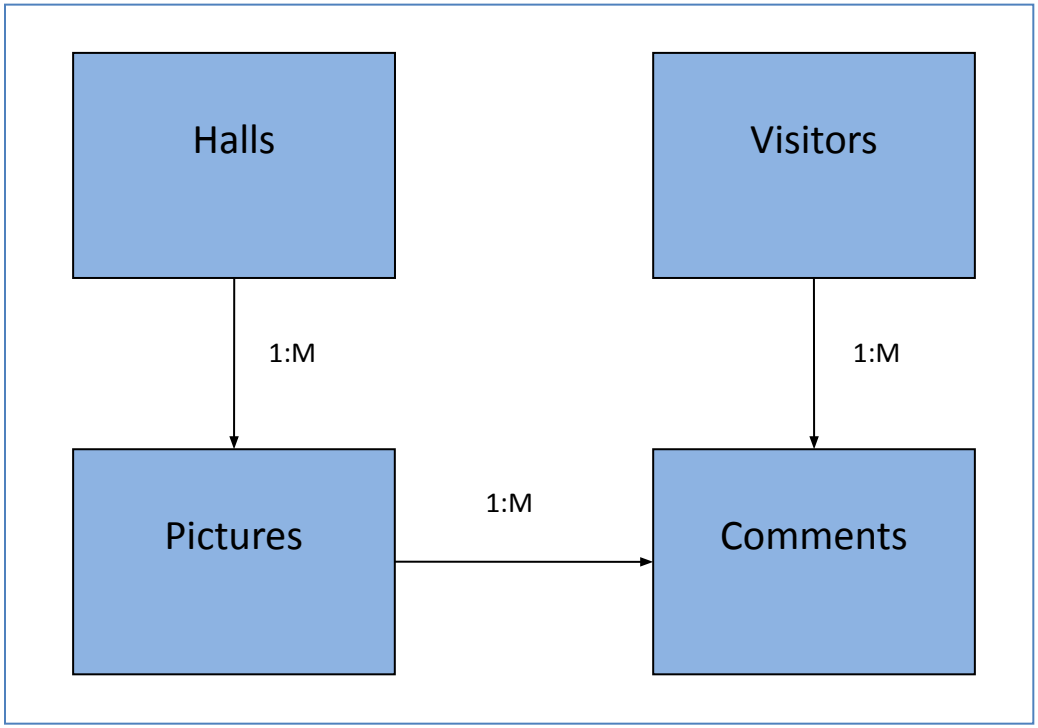


Art Museum: Условия и результаты поиска

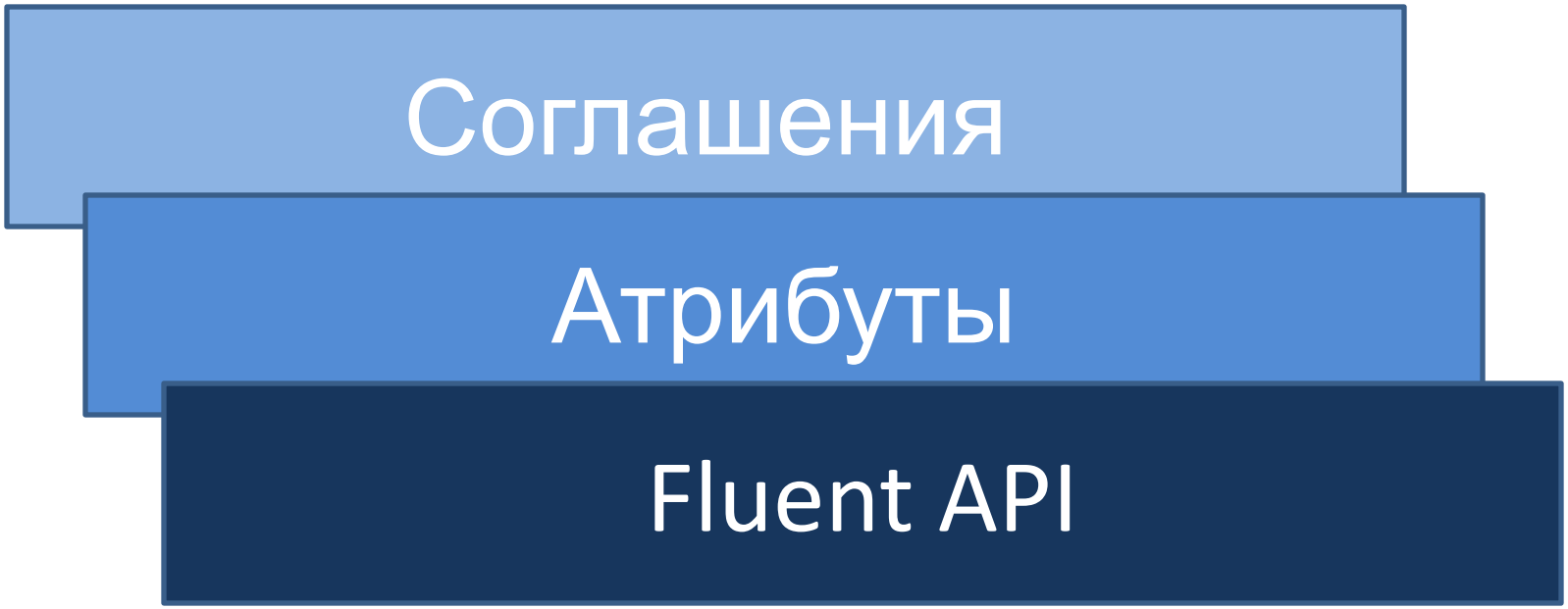


ArtMuseum: Модель данных

Сущности: картины, залы, посетители, комментарии.



Управление генерацией БД



Соглашение: Первичный ключ

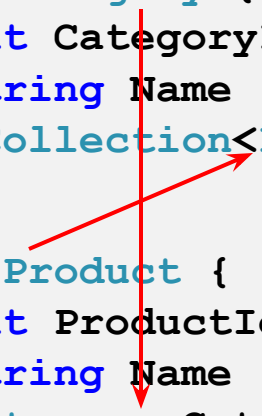
- Свойство является первичным ключом, если свойство называется Id или <class_name>Id.

- Если первичный ключ имеет тип int, long или short, он делается автоинкрементным.

Соглашение: Бинарные отношения

- Отношение определяется как бинарное, если у двух классов обнаруживаются свойства или свойства-коллекции с типом противоположного класса.

```
public class Category {
    public int CategoryId { get; set; }
    public string Name { get; set; }
    public ICollection<Product> Products { get; set; }
}

public class Product {
    public int ProductId { get; set; }
    public string Name { get; set; }
    public Category Category { get; set; }
}
```



Соглашение: Внешний ключ

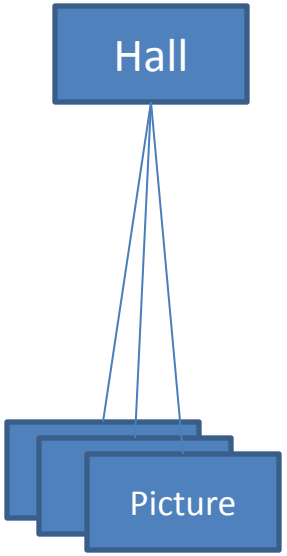
Для определения имени свойства, которое поставляет значения внешнего ключа

- <navigation property name><primary key property name> (например, PlaceId),
- <principal class name><primary key property name> (например, HallId)
- <primary key property name> (например, Id, но в данном примере неприменимо)

Навигац. свойство

Навигац. свойство

```
public class Hall
{
    [Key]
    public string Id { get; set; }
    public string Name { get; set; }
    public ICollection<Picture> Pictures { get; set; }
}
public class Picture
{
    public int Id { get; set; }
    public Hall Place { get; set; }
    public string PlaceId { get; set; }
}
```



Атрибуты аннотации данных

[Key]
Свойство входит с состав первичного ключа сущности

[Column]
Атрибут свойства для указания имени столбца, ординарного типа и типа данных

[ForeignKey]
Ставится на навигационное свойство и задает имя скалярного свойства – источник значений внешнего ключа.

[NotMapped]
Атрибут свойства или класса для его исключения из базы данных

[Table]
Атрибут класса для указания имени таблицы и схемы

[ConcurrencyCheck]
Свойство участвует в контроле оптимистической блокировки

[DatabaseGenerated]
Указывает, как база данных будет вычислять значение поля (Identity, Computed or None)

[InverseProperty]
Отмечает навигационное свойство, которое представляет другой конец отношения. Используется, например, при автоссылках.

Fluent API

При помощи класса DbModelBuilder можно управлять генерацией схемы.

```
public class ArtMuseumDb: DbContext
{
    public DbSet<Comment> Comments { get; set; }
    public DbSet<Hall> Halls { get; set; }
    public DbSet<Picture> Pictures { get; set; }
    public DbSet<Visitor> Visitors { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // То же, что атрибут [Key]
        modelBuilder.Entity<Hall>().HasKey(h => h.HallId);

        base.OnModelCreating(modelBuilder);
    }
}
```

Примеры использования Fluent API можно найти в

http://blog.vkuznetsov.ru/posts/2011/07/31/code-first-v-entity-framework-41-primery-ispolzovaniya-fluent-api

Классы Hall и Picture

```
public class Hall
{
    public int HallId { set; get; }
    public string Name { set; get; }

    public virtual List<Picture> Pictures { set; get; }
}
```

```
public class Picture
{
    public int PictureId { set; get; }
    public string Name { set; get; }
    public int Year { set; get; }
    public string ArtistName { set; get; }
    public byte[] Image { set; get; }
    public int HallId { set; get; }
    public string ImageMimeType { set; get; }

    public virtual Hall Hall { set; get; }
    public virtual List<Comment> Comments { set; get; }
}
```

Те же классы, но с атрибутами

```
public class Hall
{
    [Key]
    public int HallId { set; get; }
    [Column]
    public string Name { set; get; }

    public virtual List<Picture> Pictures { set; get; }
}
```

```
public class Picture
{
    [Key]
    public int PictureId { set; get; }
    [Column]
    public string Name { set; get; }
    [Column]
    public int Year { set; get; }
    [Column]
    public string ArtistName { set; get; }
    [Column]
    public byte[] Image { set; get; }
    [Column]
    public int HallId { set; get; }
    [Column]
    public string ImageMimeType { set; get; }
    [ForeignKey("HallId")]
    public virtual Hall Hall { set; get; }

    public virtual List<Comment> Comments { set; get; }
}
```

Классы Visitor и Comment

```
public class Visitor
{
    [Key]
    public int VisitorId { set; get; }
    [Column]
    public string Name { set; get; }
    public virtual List<Comment> Comments { set; get; }
}
```

```
public class Comment
{
    [Key]
    public int CommentId { set; get; }
    [Column]
    public int VisitorId { set; get; }
    [Column]
    public int PictureId { set; get; }
    [Column]
    public DateTime Date { set; get; }
    [Column]
    public string Text { set; get; }
    [ForeignKey("VisitorId")]
    public virtual Visitor Visitor { set; get; }  // 'virtual' for lazy load
    [ForeignKey("PictureId")]
    public virtual Picture Picture { set; get; }  // 'virtual' for lazy load
}
```

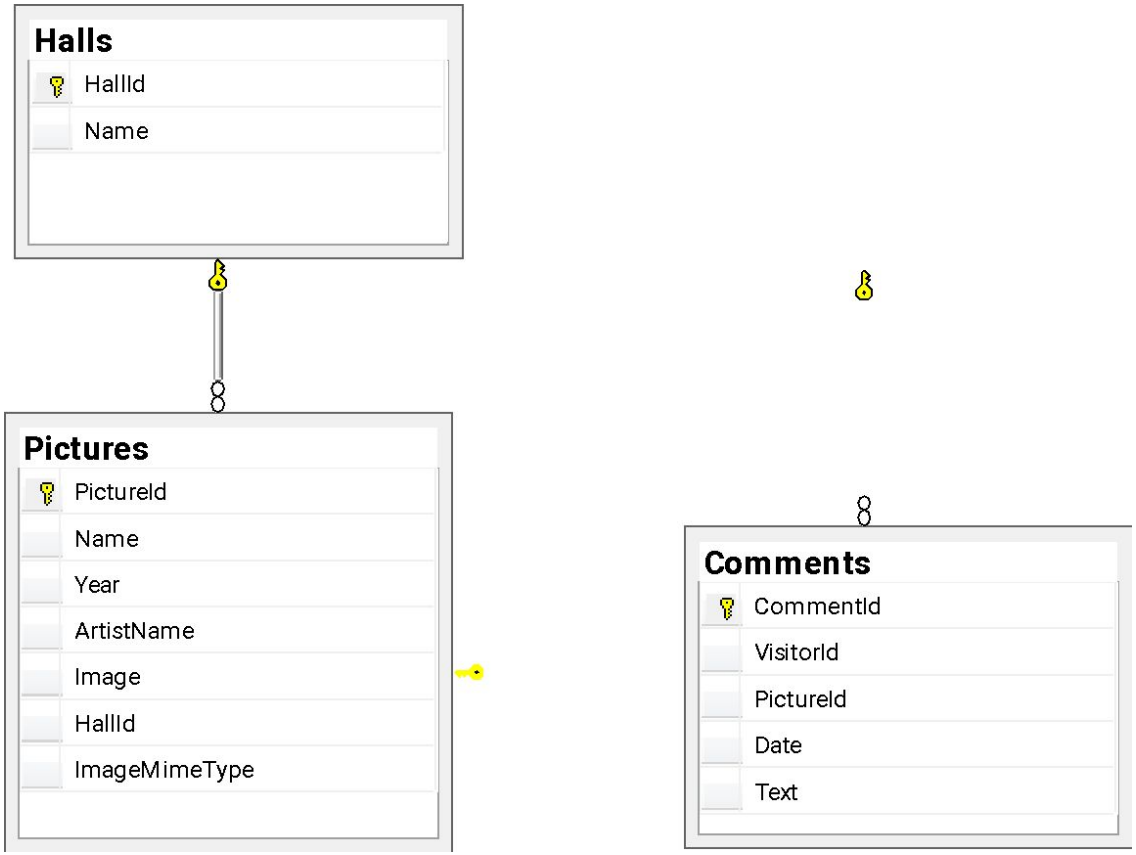
SqlServer Express

- Не может создавать базы данных объемом более 10G.
- Не может использовать более 1G оперативной памяти.
- Имеет механизм User Instance – создания отдельного экземпляра сервера для работы с отдельным пользователем. Это упрощает меры безопасности.

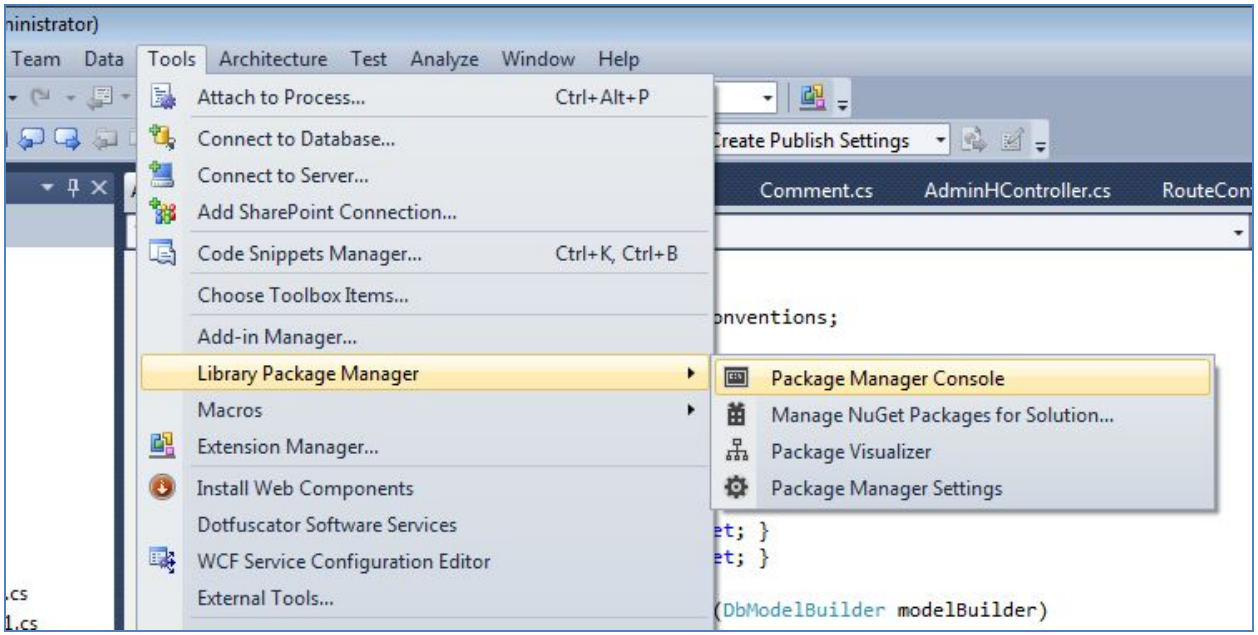
Строка соединения в web.config

```
<connectionStrings>
   <add name="ArtMuseumDb" providerName="System.Data.SqlClient"
        connectionString="Data Source=(LocalDb)\v11.0;
        Integrated Security=SSPI;
        AttachDBFilename=|DataDirectory|ArtMuseumDb.mdf" />
</connectionStrings>
```

Сгенерированная схема БД



Изменение модели данных



Для синхронизации модели со схемой базы данных имеется механизм миграций с тремя командами:

- **Enable-Migrations** – создается каталог Migrations, в котором будет накапливаться программный код миграций.
- **Add-Migration** имя_класса_контекста – добавляется код последней миграции
- **Update-Database** – вносятся изменения в базу данных

Слой доступа к данным

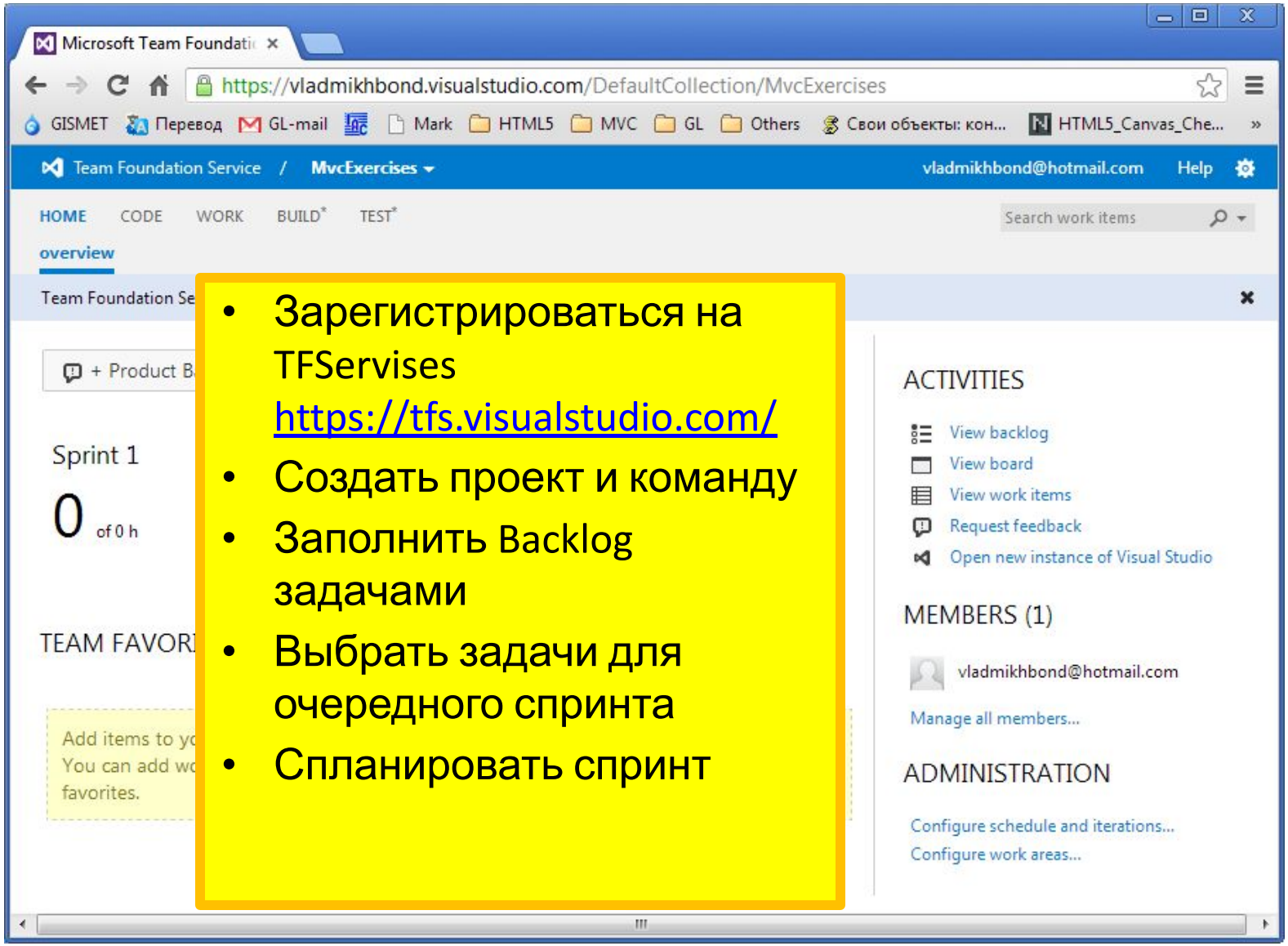
```
public class ArtMuseumDb : DbContext, IHallRepository
    {
      public DbSet<Comment> Comments { get; set; }
      public DbSet<Hall> Halls { get; set; }
      public DbSet<Picture> Pictures { get; set; }
      public DbSet<Visitor> Visitors { get; set; }

      public Hall HallById(int id)
      {
          return this.Halls.Find(id);
      }

      public void InsertHall(Hall hall)
      {
          this.Halls.Add(hall);
          this.SaveChanges();
      }

      public void UpdateHall(Hall hall)
      {
          this.Entry(hall).State = System.Data.EntityState.Modified;
          this.SaveChanges();
      }

      public void DeleteHall(int id)
      {
          Hall hall = this.Halls.Find(id);
          this.Halls.Remove(hall);
          this.SaveChanges();
      }
      public List<Hall> ReadHalls()
      {
          return this.Halls.ToList();
      }
   }
}
```

Чтобы увидеть сгенерированную команду SQL to Entity, достаточно вызвать метод ToString() для переменной запроса.

Самостоятельно

- Выбрать тему учебного проекта.
- Разработать модели данных.
- Сгенерировать схему в базе данных.
- Реализовать слой доступа к данным по шаблону "Репозиторий"

Team Foundation Services



- Зарегистрироваться на TFServises https://tfs.visualstudio.com/
- Создать проект и команду
- Заполнить Backlog задачами
- Выбрать задачи для очередного спринта
- Спланировать спринт