# Веб-программирование

## Часть 1

Юрий Литвинов
yurii.litvinov@gmail.com

26.10.2018г

# Веб-приложения

## Как оно вообще работает

- Пользователь заходит браузером на определённый URL
  - На самом деле, выполняя HTTP GET-запрос на порт 80 или 443 (обычно)
- ОС сервера перенаправляет запрос запущенному там *веб-серверу*
  - Например, Apache, IIS
- Веб-сервер — отдельный процесс, в рамках которого запущено несколько *веб-приложений*, веб-сервер по URL запроса определяет, какому веб-приложению он адресован, и передаёт запрос ему
- Веб-приложение формирует ответ и отправляет его обратно по HTTP в виде HTML-страницы
- Эта страница и показывается пользователю в браузере

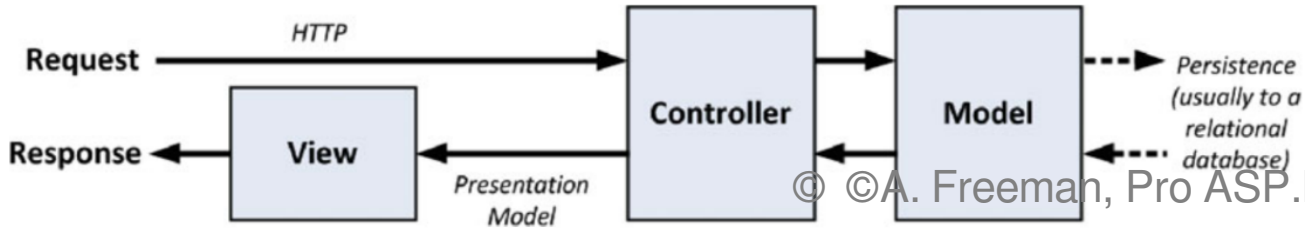
# Веб-сервисы

- *Веб-сервис* — это примерно то же самое, но не для пользователя, а для других приложений
- Нужны для создания распределённых приложений
- Общаются не с помощью HTML, а посредством специализированных протоколов
  - Например, SOAP
    - Использует синтаксис XML, может использовать HTTP как транспорт
- Как правило, содержат механизм публикации метаинформации
  - Например, WSDL
- Реализуются посредством технологий, например, Windows Communication Foundation

# Веб-приложения и .NET

- Веб-сервер — IIS (Internet Information Services), IIS Express, Kestrel
  - Есть “из коробки” в Windows, IIS Express поставляется с Visual Studio и используется для отладки
- Технология для разработки веб-приложений — ASP.NET MVC
  - ASP.NET MVC 5
  - ASP.NET MVC Core 2.0
- Технология для разработки веб-сервисов — WCF
- Работа с базами данных — MS SQL Server (SQL Server Express)
- ORM — Entity Framework (Entity Framework Core)
- Облачный хостинг — Azure

# ASP.NET MVC, основные понятия






- **Модель** содержит или представляет данные, с которыми работает приложение
  - **Domain model** содержит объекты предметной области вместе с бизнес-логикой, механизмами сериализации и т.д.
  - **View Model** содержит классы, удобные для отображения во View, без бизнес-логики
- **Представление** (View) отвечает за показ данных из модели пользователю
  - Работает в браузере, но генерится на сервере
- **Контроллер** отвечает за обработку входящих запросов, преобразование моделей и формирование данных для видов

# Работа с БД

- Entity Framework
  - Object-Relational Mapping-технология, представляет таблицы реляционной БД как объекты C#
  - Если вы пишете SQL руками в коде, вы делаете что-то не так
- MS SQL Server LocalDB
  - Реляционная СУБД, урезанная версия SQL Server, предназначенная прежде всего для разработчиков
  - Не требует конфигурации, после установки экземпляр создаётся автоматически и к нему можно просто подключаться
- На самом деле, подобных технологий только популярных десятки
  - Например, MongoDB или SQLite неплохи

# Entity Framework, подробности

- Работа с базой делается через классы *модели*
  - Code first
  - Database first
  - Model first
- *DbContext* — базовый класс, представляющий сессию при работе с базой

# Немного примеров

## Из https://docs.microsoft.com/en-us/ef/core/

Конфигурирование:

```
public class BloggingContext : DbContext
  {
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
      optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=MyDatabase;Trusted_Connection=True;");
    }
  }
```

Чтение:

```
using (var db = new BloggingContext())
{
  var blogs = db.Blogs
    .Where(b => b.Rating > 3)
    .OrderBy(b => b.Url)
    .ToList();
}
```

Запись:

```
using (var db = new BloggingContext())
{
  var blog = new Blog { Url = "http://sample.com" };
  db.Blogs.Add(blog);
  db.SaveChanges();
}
```