**Глава 32**

**Создание проекта для примера**

**Добавление модели данных**

Добавим в папку Models класс модели Person.cs, класс модели Department.cs и класс модели Location.cs.

Каждая из трех вышеуказанных моделей определяет ключевое свойство, значение которого будет выделяться базой данных при сохранении новых объектов, также определяется свойство foreign key, определяющее отношение между классами. Они добавляются навигационными свойствами, которые будут использоваться EF Core методом Include для включения связанных данных в запросы.

Создадим класс контекста DataContext.cs в папке Models.

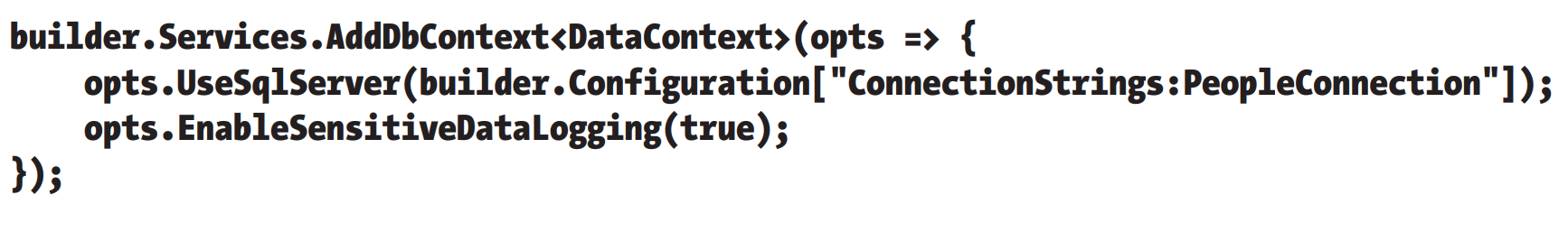
Класс контекста определяет свойства, которые будут использоваться при запросе данных для Person, Department и Location.

**Подготовка Seed Data**

Добавим файл SeedData.cs в папку Models.

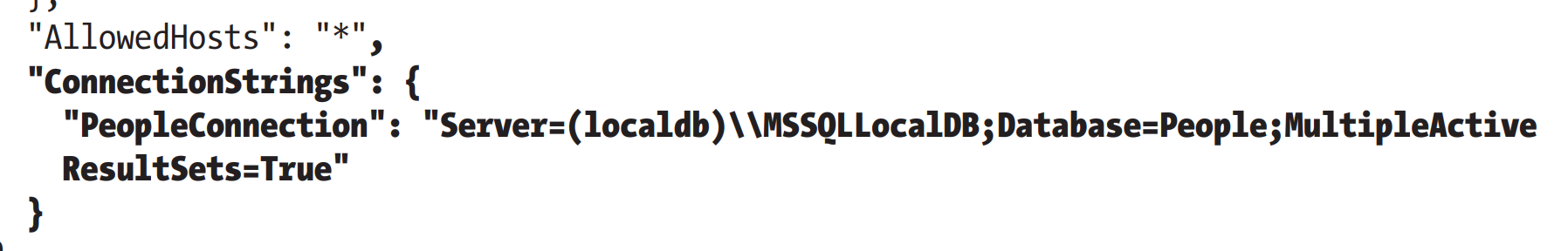
Статический метод SeedDatabase гарантирует, что все ожидающие миграции были применены к базе данных. Если БД пуста, данный метод заполняет базу начальными данными. EF Core заботится об отображении объектов на таблицы БД и значения ключевых свойств устанавливает автоматически, когда данные поступают в БД.

**Конфигурация EF Core сервисов и middleware**

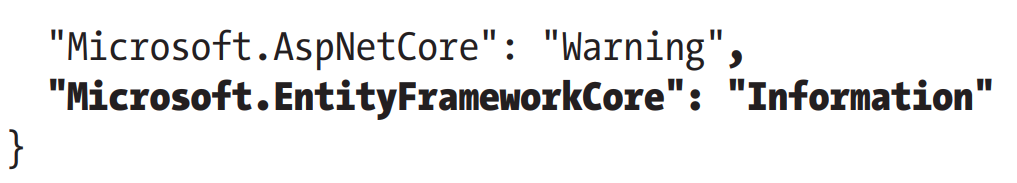




Для определения строки подключения, которая будет использоваться в приложении, добавим соответствующую конфигурацию в файл appsettings.json.

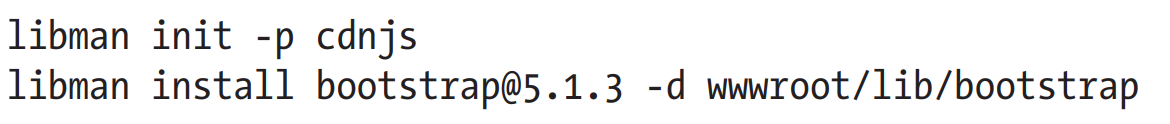


В дополнение к строке подключения повысим детализацию логирования для EF Core так, что SQL запросы к БД будут логироваться:



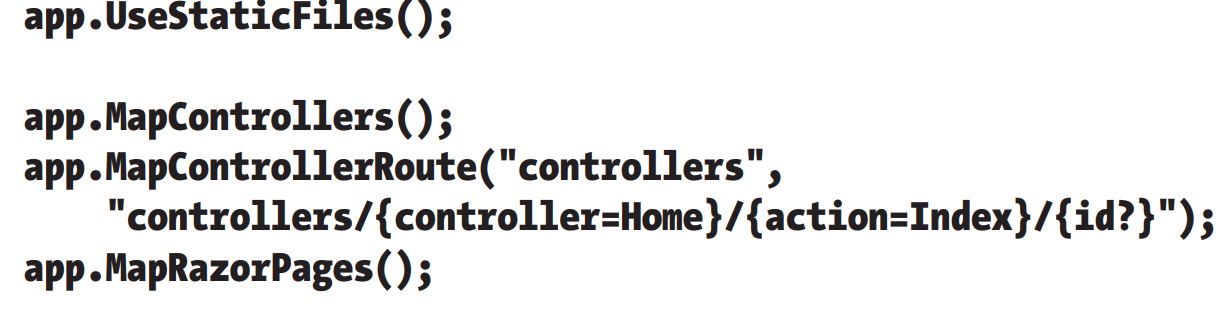
**Создание и применение миграции**

**Добавление Bootstrap CSS Framework**

****

**Конфигурирование сервисов и middleware**





В дополнение к маппингу роутинга контроллеров я добавил роутинг, который матчит URL путь, начинающийся со слова controllers, что облегчает следование примерам из следующих частей, потому как они будут переключаться между контроллерами и Razor Pages. Это та же конвенция, которую я заимствовал в прошлых главах и я буду использовать роут URL, начинающийся с pages для Razor Pages.

**Создание контроллера и вью**

Для вывода данных приложения используя контроллер, создадим папку Controllers в корне проекта и добавим в нее файл HomeController.cs.

Для предоставления контроллера со вью создадим папку Views/Home и добавим в нее Razor вью Index.cshtml.

Для доступности тег хелперов и добавления пространств имен, которые будут доступны в приложении по умолчанию, добавим Razor View Imports файл \_ViewImports.cshtml в папку Views.

Для установки дефолтного лейаута для вью контроллеров добавим Razor View Start файл \_ViewStart.сshtml в папку Views.

Создадим лейаут в папке Views/Shared и добавим в нее Razor Layout с наименованием \_Layout.cshtml.

**Создание Razor Page**

Для вывода данных приложения используя Razor Page создадим папку Pages и добавим в нее Razor Page с наименованием Index.cshtml.

Для доступности тег хелперов и добавления пространств имен, которые будут доступны во вью секции Razor Pages, добавим Razor View Imports файл \_ViewImports.cshtml в папку Pages.

Для установки дефолтного лейаута для Razor Pages добавим Razor View Start файл \_ViewStart.сshtml в папку Pages.

Создадим лейаут \_Layout.cshtml в папке Pages.