1. **Таблица маршрутов** — это структура данных, которая хранит все маршруты, определенные в приложении. Каждый маршрут может быть представлен в виде ключ-значение, где ключом является URI, а значением — соответствующий обработчик.
2. **Задать маршруты можно с помощью атрибута Route**
3. **Конечная точка (endpoint)** — это исполняемый кода приложения, обрабатывающий запрос. Вся ссылка, ведущая к выполнению action’а.

Конечные точки определяются в приложении и настраиваются при его запуске. Процесс сопоставления конечных точек может извлекать значения из URL-адреса запроса и предоставлять эти значения для обработки запроса.

1. Шаблон маршрута, который сопоставляется с конечной точкой, может иметь параметры. Параметры имеют имя и определяются в шаблоне маршрута внутри фигурных скобок: {название\_параметра}
2. Система маршрутизации сопоставляет маршруты с запросом в том порядке, в котором они определены. Поэтому если мы определяем маршруты, которые могут пересекаться, то вначале надо определять те маршруты, которые имеют больший приоритет или более специфичные. А больше общие или стандартные маршруты должны определяться ближе к концу.
3. Сопоставление адреса URL или **URL matching** представляет процесс сопоставления запроса с конечной точкой. Данный процесс основывается на пути запроса и полученных в запросе заголовках. Данный процесс проходит ряд этапов:
4. Сначала выбираются все конечные точки, шаблон маршрута которых совпадает с путем запроса
5. Далее из полученного на предыдущем этапе набора конечных точек удаляются те, которые не соответствуют ограничениям маршрута
6. Затем из полученного на предыдущем этапе набора конечных точек удаляются те, которые не удавлетворяют политике объекта MatcherPolicy (вкратце: класс **MatcherPolicy** позволяет определить порядок сравнения конечных точек и адреса URL)
7. И в самом конце применяется объект **EndpointSelector** для выбора из полученного на предыдущем этапе списка конечной точки, которая в конечном счете и будет обрабатывать запрос
8. **Приоритет конечных точек зависит от двух факторов:**

* Порядок следования в наборе конечных точек
* Приоритетность шаблона маршрута

1. **Приоритетность шаблонов маршрута зависит от специфичности шаблона.** Специфичность шаблона определяется на основе следующих критериев:

* Шаблон маршрута с большим количеством сегментов более специфичен, чем шаблон меньшим количеством сегментов
* Сегмент с текстовым литералом (статический сегмент) более специфичен, чем сегмент с параметром маршрута
* Сегмент с параметром, к которому применяется ограничение маршрута, более специфичен, чем сегмент с параметром без ограничения
* Комплексный сегмент более специфичен, чем сегмент с параметром с ограничением
* Параметр catch-all (параметр, который соответствует неопределенному количеству сегментов) наименее специфичен

Если в конечном счете осталось две и более конечных точек, которые соответствуют запрошенному адресу, и соответственно система маршрутизации не может выбрать, какая из этих конечных точек должна обрабатывать маршрут, то генерируется исключение.

1. **UseRouting** добавляет соответствие маршрута в конвейер ПО промежуточного слоя. Это ПО промежуточного слоя обращается к набору конечных точек, определенных в приложении, и выбирает наиболее подходящее на основе запроса.
2. **UseEndpoints** добавляет выполнение конечной точки в конвейер ПО промежуточного слоя. Он запускает делегат, связанный с выбранной конечной точкой.
3. Фреймворк MVC позволяет использовать в приложении маршрутизацию на основе атрибутов. Такой тип маршрутизации еще называется **Attribute-Based Routing**. Атрибуты предоставляют более гибкий способ определения маршрутов. Маршруты, определенные с помощью атрибутов, имеют приоритет по сравнению с маршрутами, определенными в классе Startup.

В качестве параметра атрибут Route принимает шаблон URL, с которым будет сопоставляться запрошенный адрес.

Если в проекте планируется использовать только маршрутизацию на основе атрибутов, то в классе Startup мы можем не определять никаких маршрутов.

1. От всех параметров шаблона маршрутов в атрибутах отличаются два параметра controller и action, которые ссылаются соответственно на контроллер и его действие. При использовании их надо помещать в квадратные скобки, а не в фигурные, как другие параметры: [Route("[controller]/[action]/{id?}")].

С помощью атрибутов можно задать несколько маршрутов для одного метода.

ПОПРОЩЕ

 **Таблица маршрутов** — это список всех маршрутов (путей), которые есть в приложении. Каждый маршрут связывает адрес (URI) с обработчиком (кодом, который выполняется).

 **Маршруты можно задавать** с помощью атрибута Route.

 **Конечная точка (endpoint)** — это место, где выполняется код приложения, обрабатывающий запрос. Это вся ссылка, которая ведет к выполнению действия (action). Конечные точки определяются при запуске приложения и могут извлекать значения из URL, чтобы обработать запрос.

 **Шаблон маршрута** может содержать параметры, которые обозначаются в фигурных скобках: {название\_параметра}.

 **Система маршрутизации** проверяет маршруты в том порядке, в котором они указаны. Если маршруты перекрываются, более специфичные маршруты должны быть указаны первыми, а более общие — ближе к концу.

 **Сопоставление адреса URL** — это процесс, когда система находит, какой код должен обработать запрос. Этот процесс включает несколько шагов:

1. Сначала выбираются конечные точки, которые соответствуют пути запроса.
2. Затем удаляются те, которые не соответствуют условиям маршрута.
3. Далее убираются те, которые не подходят под политику сопоставления.
4. В конце выбирается конечная точка, которая будет обрабатывать запрос.

 **Приоритет конечных точек** зависит от:

* Порядка в наборе конечных точек.
* Специфичности шаблона маршрута.

 **Специфичность шаблона** определяется по следующим критериям:

* Чем больше сегментов в шаблоне, тем он специфичнее.
* Сегмент с текстом более специфичен, чем сегмент с параметром.
* Сегмент с ограниченным параметром более специфичен, чем без ограничений.
* Комплексный сегмент более специфичен, чем простой параметр.
* Параметр, который совпадает с несколькими сегментами (catch-all), наименее специфичен.
* Если остаются несколько подходящих конечных точек, возникает ошибка.

 **UseRouting** добавляет маршрутизацию в систему обработки запросов. Это программное обеспечение выбирает подходящую конечную точку на основе запроса.

 **UseEndpoints** добавляет выполнение конечной точки в обработку запросов. Оно запускает код, связанный с выбранной конечной точкой.

 **MVC позволяет использовать маршрутизацию на основе атрибутов** (Attribute-Based Routing). Это более гибкий способ определения маршрутов, который имеет приоритет над маршрутами, заданными в классе Startup. Атрибут Route принимает шаблон URL для сопоставления с запросом. Если вы используете только маршрутизацию на основе атрибутов, маршруты в классе Startup можно не определять.

 **В атрибутах есть два специальных параметра**: controller и action, которые ссылаются на контроллер и его действие. Их нужно помещать в квадратные скобки: [Route("[controller]/[action]/{id?}")]. С помощью атрибутов можно задать несколько маршрутов для одного метода.