WebApi

* **HttpServer**: получает от хостирующей среды объект запроса HttpRequestMessage
* **HttpRoutingDispatcher**: устанавливает данные для маршрутизации для текущего запроса
* **HttpControllerDispatcher**: выбирает контроллер и вызывает его метод для обработки запроса и генерации HttpResponseMessage

#### **Атрибуты привязки:**

#### Массивы, словари передаем: [ModelBinder]int[] nums) – запрос: api/values/getvalue?nums=2&nums=3 и

api/values/setbooks?books[0][key]=id123456&books[0][value][name]=Война и мир

#### FromUri, Frombody

public Book GetValue([FromUri]Book b)

{

    return b;

}

Куки: var cookie = new CookieHeaderValue("id", "12345"); // имя куки - id, значение - 12345

    cookie.Expires = DateTimeOffset.Now.AddDays(1); // время действия куки - 1 день

    cookie.Domain = Request.RequestUri.Host; // домен куки

    cookie.Path = "/"; // путь куки

    var response = Request.CreateResponse<IEnumerable<Book>>(HttpStatusCode.OK, db.Books);

    response.Headers.AddCookies(new CookieHeaderValue[] { cookie });

Валидация:

[Range(1800, 2000, ErrorMessage = "Год должен быть в промежутке от 1800 до 2000")]

    [Required(ErrorMessage = "Укажите год издания книги")]

    public int Year { get; set; }

if (book.Name == "Война и мир")

        {

            ModelState.AddModelError("book.Name", "Недопустимое название для книги");

            ModelState.AddModelError("book.Name", "Название не должно начинаться с заглавной буквы");

        }

        if (!ModelState.IsValid)

            return BadRequest(ModelState);

**Маршрутизация: HttpRoutingDispatcher** брабатывает запрос для получения данных маршрута и добавляет эти данные в коллекцию **HttpRequestContext.RouteData**

Интерфейс IHttpRouteData определяет два свойства:

* **Route**: объект IHttpRoute, представляющий текущий маршрут
* **Values**: объект IDictionary<string, object>, содержащий данные маршрута

**Фильтр аутентификации -** Запускается первым, перед любыми другими фильтрами или методом действия, но может быть запущен снова после фильтров авторизации.

Кроме того, фильтры аутентификации выполняются после выполнения метода действия, но перед обработкой ActionResult. Метод OnAuthenticationChallenge() вызывается инфраструктурой MVC Framework всякий раз, когда запрос не прошел аутентификацию или не удовлетворил политикам авторизации для метода действия. Если в коде OnAuthentication() устанавливается значение свойства Result объекта контекста, то инфраструктура MVC Framework вызовет метод OnAuthenticationChallenge(). Если в коде OnAuthenticationChallenge() значение свойства Result объекта контекста не устанавливается, то будет выполнен один из методов OnAuthentication().Метод OnAuthentication() используется для создания результата, который сообщает пользователю об ошибке аутентификации и может затем быть переопределен методом OnAuthenticationChallenge(), чтобы запросить у пользователя учетные данные.

Если запрос не прошел фильтр авторизации, то будет вызван метод OnAuthenticationChallenge() фильтра аутентификации, что позволяет предложить пользователю ввести требуемые учетные данные. Инфраструктура ASP.NET MVC Framework вызывает метод OnAuthenticationChallenge() еще один, финальный раз после того, как метод действия был выполнен, но перед возвращением и выполнением ActionResult.