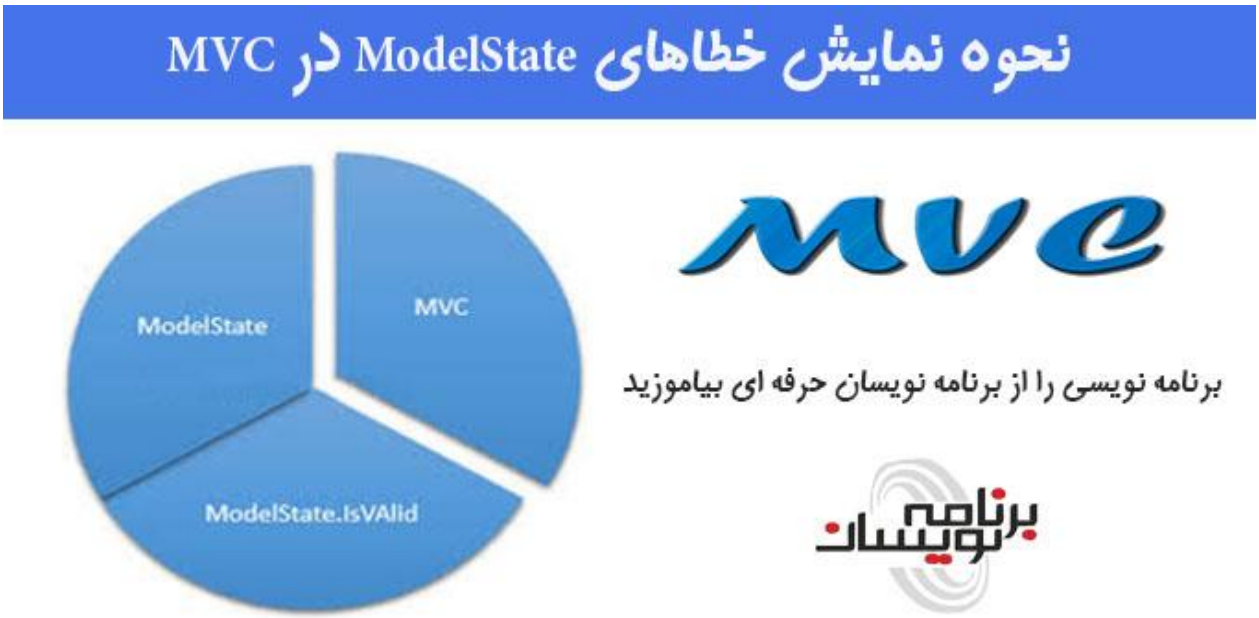
ممکن است در برنامه خود نیاز داشته باشید اطلاعات کاربر را در Controllerبررسی کنید و پیام مناسبی به کاربر خود نمایش دهید ModelState .روشی برای بررسی خطاها در سمت سرور است و از طریق آن می توان اطلاعات ارسالی را در Controller کنترل کرد که در صورت بروز خطا پیغام مناسب را به کاربر نشان می دهد. در این مقاله می خواهیم نحوه نمایش خطاهای ModelStateرا بررسی کنیم .



امروزه API ها در همه جا استفاده استفاده می شوند. ما در حال حاضر می توانیم از طریق ابزارها و پلت فرم های مختلفی مانند لپ تاپ، تبلت، گوشی هوشمند و غیره به وب دسترسی داشته باشیم. اغلب اوقات آنها بخش کلیدی از معماری ما هستند. چه چیز آنها را مهم می سازد؟ بله، داده ها. وقتی که ما یک اپلیکیشن موبایل یا اپلیکیشن وب ایجاد می کنیم، داده ها نقش کلیدی را ایفا می کنند. فقط چیزی که مهم است APIها داده ها نا معتبر را قبول نمی کنند. فیلدهای مورد نیاز یا required fieldداده های نا معتبر را به اپلیکیشن منتقل نمی کنند .مثلا طول رمز عبوری که وارد کرده ایم به اشتباه وارد شده باشد. هر نوع APIکه استفاده می کنیم باید داده را اعتبار سنجی کنند. در دنیای دات نت خطا ها را با استفاده از ModelStateضبط می کنیم. مشکل این است که ModelStateدر MVCیک مقدار متفاوت از WebAPIاست. چگونه می توانیم آنها را از هم تشخیص دهیم؟

همه چیز در مورد ModelState

واقعیت این است که فرایند Binding بین Web API و MVC متفاوت است .از قالب بندی ها برای deserialiseکردن بدنه درخواست ها استفاده می کند و نتایج در ساختار متفاوت بدست می آیند. در WebAPIنام کلاس شامل شونده بخشی از error Key می باشد. این بدین معناست که باید WebAPI ModelState خود را در اپلیکیشن MVCبه نحوی ترجمه کنیم. سپس بعد از آن ما قادر به نمایش پیام های اعتبار سنجی در سمت کلاینت هستیم.

در بخش اول، فراخوانی WebAPI controllerدر MVC را توضیح می دهیم. در بخش دوم ما بررسی می کنیم که چگونه داه ای به WebAPI controllerپست کنیم. اجازه دهید بررسی کنیم که اگر نیاز داشتیم داده پست شده به اپلیکیشن را اعتبار سنجی کنیم، چه عملی را باید انجام دهیم.

در ابتدا این سوال پیش می آید که اگر همه چیز بد پیش برود باید چه کاری انجام دهیم؟

برای شروع، دو خصوصیت مورد نیاز (required attributes)را به مدل APIاضافه می کنیم. همچنین اگر مدل معتبر نباشد وضعیت Bad Request HTTP 401را بر میگردانیم.

```
public class ProductApiModel
{
   public int ProductId { get; set; }

   [Required]
   public string Name { get; set; }

   [Required]
   public string Description { get; set; }

   public DateTime CreatedOn { get; set; }
}

public class ProductsController : ApiController
{
   public IHttpActionResult Post(ProductApiModel model)
   {
      if (!ModelState.IsValid)
      {
```

```
            return BadRequest(ModelState);
        }

        // should do some mapping and save a product in a data store somewhere,
        // but we're not focusing on that for our little demo - just return a random int
        return Ok(7);
    }
}
```

توجه کنید که ما کل ModelState را از APIارسال می کنیم. این کار اجازه می دهد که بتوانیم آن را تجزیه کرده و خطا ها را برای نمایش در اپلیکیشن MVCاستخراج کنیم. و آنها را در خصوصیت ErrorStateنگهداری می کنیم. اجازه دهید نگاهی به آنها بیاندازیم.

```
public abstract class ApiResponse
{
    public bool StatusIsSuccessful { get; set; }
    public ErrorStateResponse ErrorState { get; set; }
    public HttpStatusCode ResponseCode { get; set; }
    public string ResponseResult { get; set; }
}

public class ErrorStateResponse
{
    public IDictionary<string, string[]> ModelState { get; set; }
}
```

رمزگشایی Error Response Goodness

حالا در این مرحله باید دو تغییر را در ClientBaseاعمال کنیم.

```
private static async Task<TResponse> CreateJsonResponse<TResponse>
    (HttpResponseMessage response) where TResponse : ApiResponse, new()
{
    var clientResponse = new TResponse
```

```
    {
        StatusIsSuccessful = response.IsSuccessStatusCode,
        ErrorState = response.IsSuccessStatusCode ? null :
            await DecodeContent<ErrorStateResponse>(response),
        ResponseCode = response.StatusCode
    };
    if (response.Content != null)
    {
        clientResponse.ResponseResult = await response.Content.ReadAsStringAsync();
    }

    return clientResponse;
}

private static async Task<TContentResponse>
    DecodeContent<TContentResponse>(HttpResponseMessage response)
{
    var result = await response.Content.ReadAsStringAsync();
    return Json.Decode<TContentResponse>(result);
}
```

مرحله نهایی: قرار دادن تمام موارد بالا در ModelState

این مرحله تمام موارد سمت API **را پوشش می دهد. اجازه دهید تغییراتی که در سمت کلاینت برای هر نوع خطایی پیش می آید را اعمال کنیم. اکشن** POST**حالا نیاز دارد کنترل کند که آیا وضعیت پاسخ موفقیت آمیز بوده است یا خیر. اگر خیر، که آن خطاهایی که از** ModelState**می آید را اضافه کند.ما این مورد را به کنترلر پایه اضافه می کنیم، این مورد را مجددا نیاز خواهیم داشت. همچنین یک** anti-forgery token **استاندارد نیز اضافه می کنیم.**

```
public class ProductController : BaseController
{
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> CreateProduct(ProductViewModel model)
    {
        var response = await productClient.CreateProduct(model);
        if (response.StatusIsSuccessful)
        {
```

```
            var productId = response.Data;
            return RedirectToAction("GetProduct", new { id = productId });
        }

        AddResponseErrorsToModelState(response);
        return View(model);
    }
}

public abstract class BaseController : Controller
{
    protected void AddResponseErrorsToModelState(ApiResponse response)
    {
        var errors = response.ErrorState.ModelState;
        if (errors == null)
        {
            return;
        }

        foreach (var error in errors)
        {
            foreach (var entry in
                from entry in ModelState
                let matchSuffix = string.Concat(".", entry.Key)
                where error.Key.EndsWith(matchSuffix)
                select entry)
            {
                ModelState.AddModelError(entry.Key, error.Value[0]);
            }
        }
    }
}
```

حالا سوال پیش می آید که آیا فراخوانی خطاهای model stateبین WebAPIو MVCمتفاوت است؟ در برای بررسی آن نگاهی به جدول زیر بیاندازید:

WebAPI Property	MVC Property
model.Name	Name
model.Description	Description

در عبارت LINQدر بالا فقط به دنبال نام MVC propertyدر WebAPIمی باشد. اگر آن را پیدا کند این خطاها را به propertyدر ModelStateاضافه می کند. اما در مقابل، .Name(برای مثال) در این جا ما Propertyهایی مانند نام (Name)و نام خانوادگی (Family)داریم. اگر داده ورودی فقط با Name مطابق شود، ما باید خطای First Name را اضافه کنیم.

فراموش نکنید که پیام های اعتبار سنجی را به فرم در CreateProduct viewاضافه کنید.

نتیجه گیری

1.در اینجا اگر WebAPI ModelStateغیر معتبر باشد یک پاسخ BadRequest را برمیگردانیم

2.خطاهای ErrorStateResponse در ApiHelperتجزیه می شوند

3.در مرحله نهایی این خطاها را به MVC ModelState خود در MVC اضافه می کنیم.