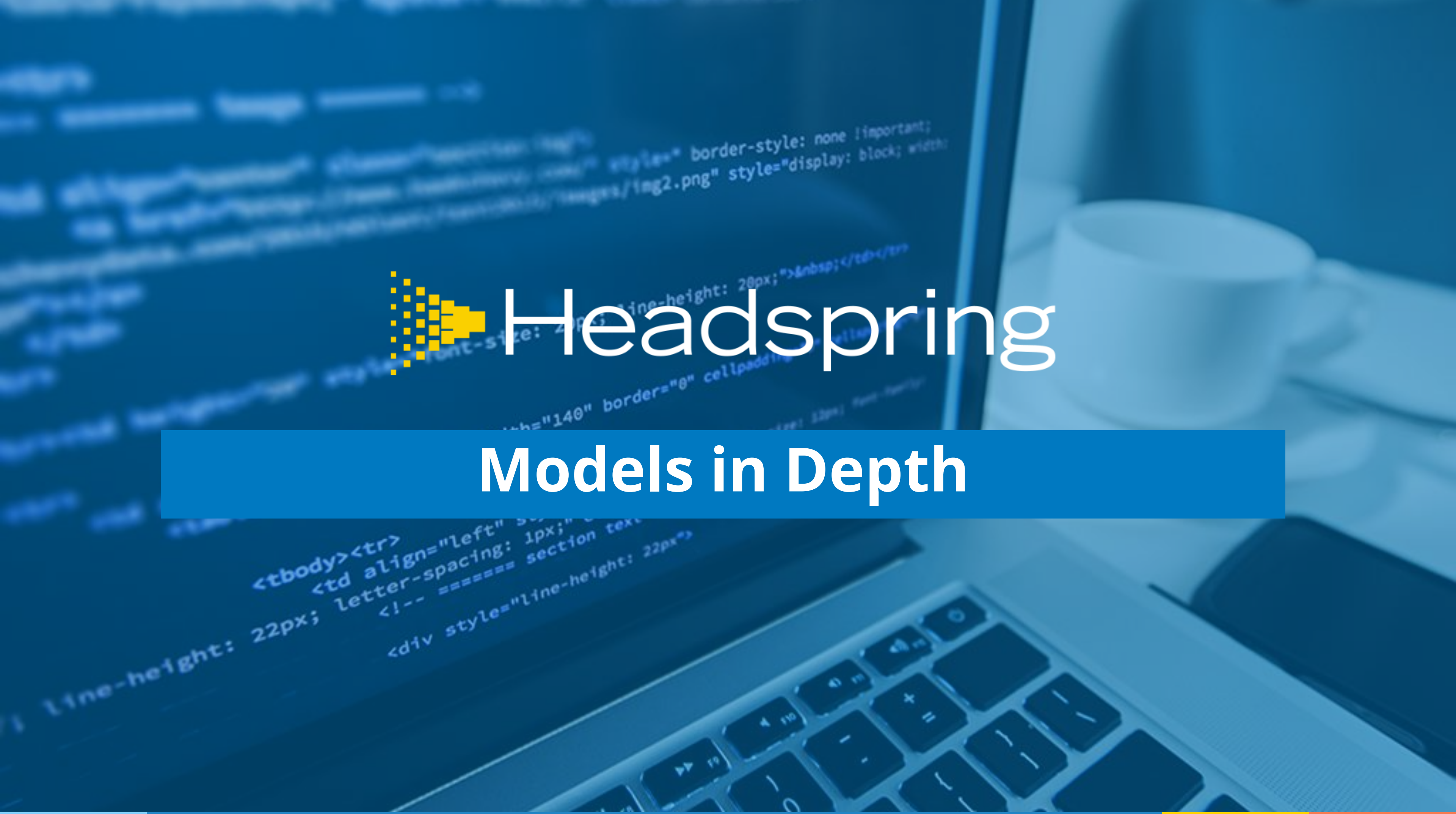
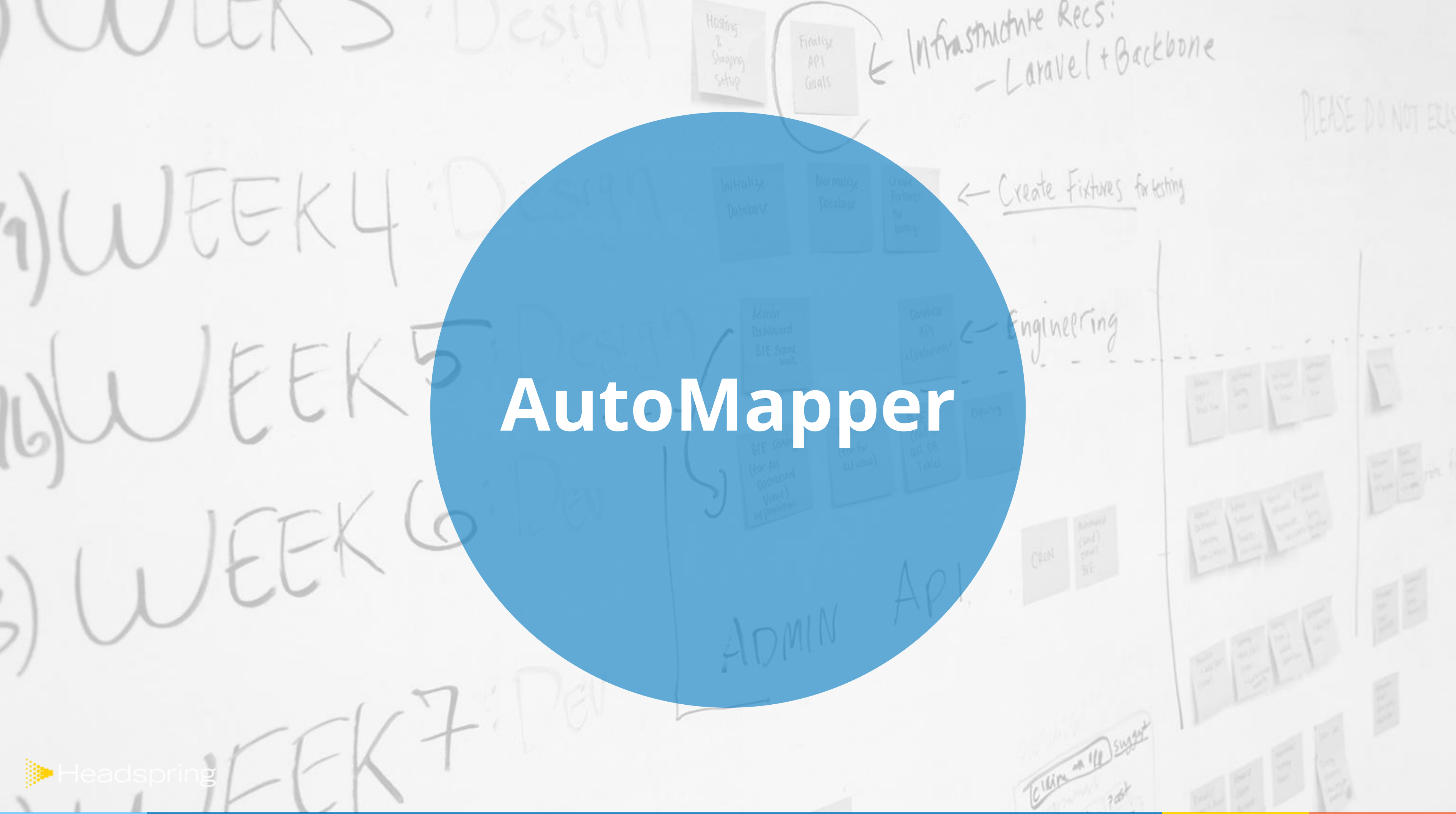


Headspring

Models in Depth



AutoMapper

Headspring

Domain Model
=> View Model

Choices

- View-specific queries/DB views/tables
- Projection (SQL, ORM, LINQ)
- Mapping from domain layer

We want:

- Strong, POCO domain model
- Strongly-typed views
- View-specific ViewModels
- To not tear our hair out

```
new EventListModel
    {
        Name = c.Name,
        AttendeeCount = c.GetAttendees().Count().ToString(),
        SessionCount = c.GetSessions().Count().ToString()
    };
```

Our solution: AutoMapper

Goals

- Convention-based
- Flattening (no NREs)
- Support edge cases
- Support LINQ
- Configuration validation

Basic usage

- Mapper.Initialize
- Mapper.Map
- ProjectTo
- new MapperConfiguration()

Flattening

```
var customer = new Customer
{
    Name = "Jimmy Bogard"
};
var order = new Order
{
    Customer = customer
};
var product = new Product()
{
    Name = "SmacBook Fro",
    Price = 1499.99m
};
order.AddOrderLineItem(product, 3);

var dto = Mapper.Map<Order, OrderDto>(order);

dto.CustomerName.ShouldEqual("Jimmy Bogard");
dto.LineItemsSum.ShouldEqual(4497.97);
```

Headspring

Edge Cases

```
Mapper.Initialize(cfg =>
{
    cfg.CreateMap<CalendarEvent, CalendarEventForm>()
        .ForMember(m => m.EventDate, opt => opt.MapFrom(src => src.EventDate.Date))
        .ForMember(m => m.EventHour, opt => opt.MapFrom(src => src.EventDate.Hour))
        .ForMember(m => m.EventDate, opt => opt.MapFrom(src => src.EventDate.Minute));

    cfg.CreateMap<Source, Destination>()
        .ForMember(m => m.Total, opt => opt.ResolveUsing<TotalResolver>());

    cfg.CreateMap<int, string>().ConvertUsing(src => src.ToString("N"));
});
```

Value Resolvers

```
public class TotalResolver
    : IValueResolver<Source, Destination, int>
{
    public int Resolve(Source source, Destination destination, int destMember,
        ResolutionContext context)
    {
        return destMember > 0 ? destMember : source.CalculateTotal();
    }
}
```

Type Converters

```
public class GuidToStringConverter
    : ITypeConverter<Guid, string>
{
    public string Convert(Guid source, string destination,
        ResolutionContext context)
    {
        return source.ToString("B");
    }
}
```

Dependency Injection

```
public class IntToStudentConverter
    : ITypeConverter<int, Student>
{
    private readonly MyDbContext _dbContext;

    public IntToStudentConverter(MyDbContext dbContext)
    {
        _dbContext = dbContext;
    }

    public Student Convert(int source, Student destination,
        ResolutionContext context)
    {
        return _dbContext.Students.SingleOrDefault(s => s.ID == source);
    }
}
```

Mappings Supported

- Mapped types (CreateMap)
- Strings
- Enums
- Arrays/Lists/Enumerables
- Dictionaries
- TypeConverter
- Conversion operators
- Dynamic
- Expressions (LINQ/OData)

Configuration validation

- Checks all destination members mapped
- Configurable through CreateMap

```
Mapper.AssertConfigurationIsValid();
```

Profile Configuration

```
public class HomeProfile : Profile
{
    public HomeProfile()
    {
        CreateMap<CalendarEvent, CalendarEventForm>()
            .ForMember(m => m.EventDate, opt => opt.MapFrom(src => src.EventDate.Date))
            .ForMember(m => m.EventHour, opt => opt.MapFrom(src => src.EventDate.Hour))
            .ForMember(m => m.EventDate, opt => opt.MapFrom(src => src.EventDate.Minute));

        CreateMap<Source, Destination>()
            .ForMember(m => m.Total, opt => opt.ResolveUsing<TotalResolver>());

        CreateMap<int, string>().ConvertUsing(src => src.ToString("N"));
    }
}
```

```
Mapper.Initialize(cfg =>
{
    cfg.AddProfile<HomeProfile>();
    cfg.AddProfile<ManageProfile>();
});
```

LINQ Projection

```
var model = db.Students
    .Where(s => s.ID == id)
    .Select(s => new StudentDetailsModel
    {
        ID = s.ID,
        EnrollmentDate = s.EnrollmentDate,
        FirstMidName = s.FirstMidName,
        LastName = s.LastName,
        Enrollments = s.Enrollments.Select(e =>
            new StudentDetailsModel.Enrollment
            {
                CourseTitle = e.Course.Title,
                Grade = e.Grade
            }).ToList()
    })
    .SingleOrDefault();
```

→

```
var model = db.Students
    .Where(s => s.ID == id)
    .ProjectTo<StudentDetailsModel>()
    .SingleOrDefault();
```

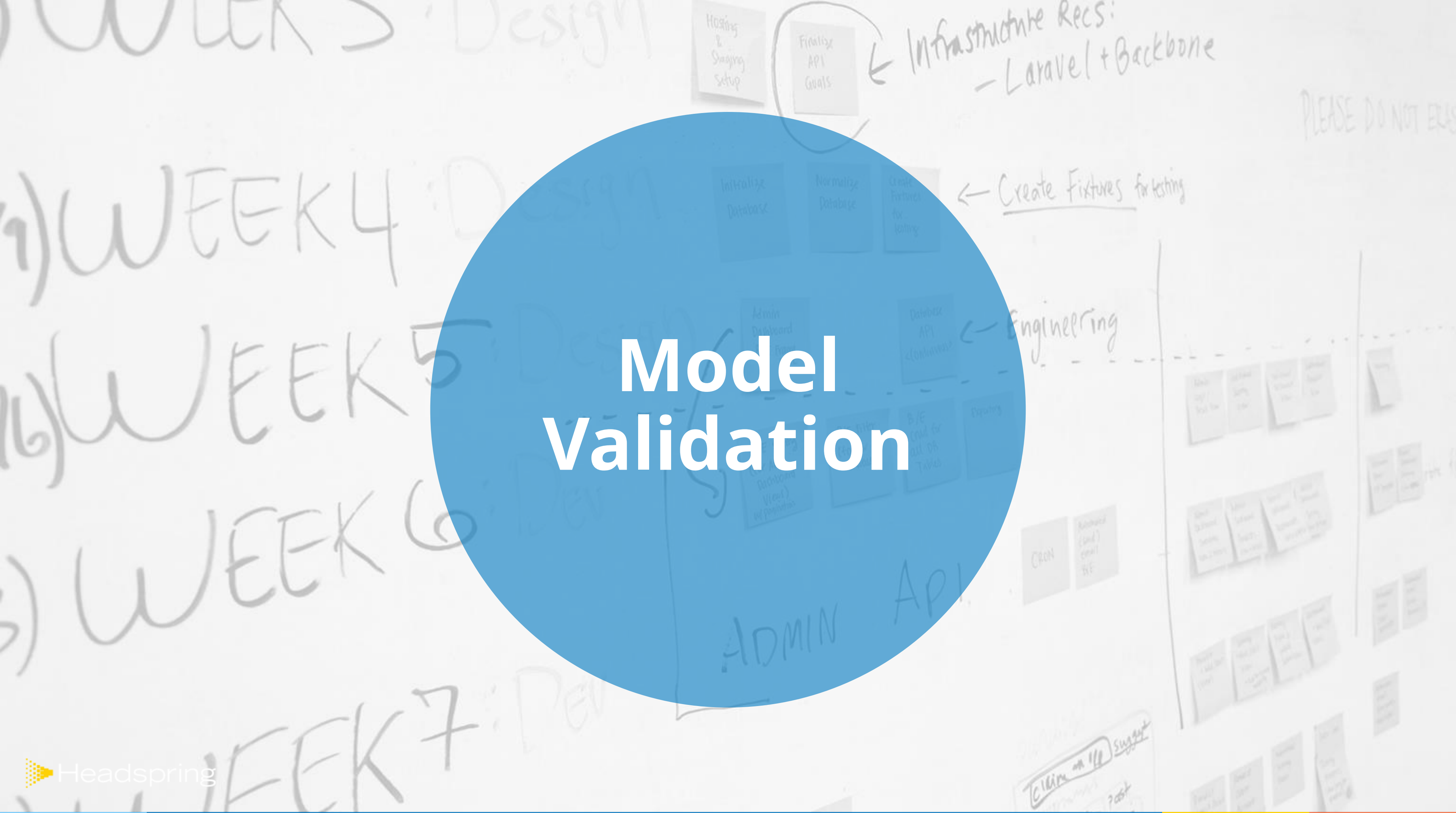
Initializing with ASP.NET MVC Core

```
> Install-Package AutoMapper.Extensions.Microsoft.DependencyInjection
```

```
public void ConfigureServices(IServiceCollection services)
{
    // Add framework services.
    services.AddDbContext<SchoolContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

    services.AddMvc();

    services.AddAutoMapper();
}
```

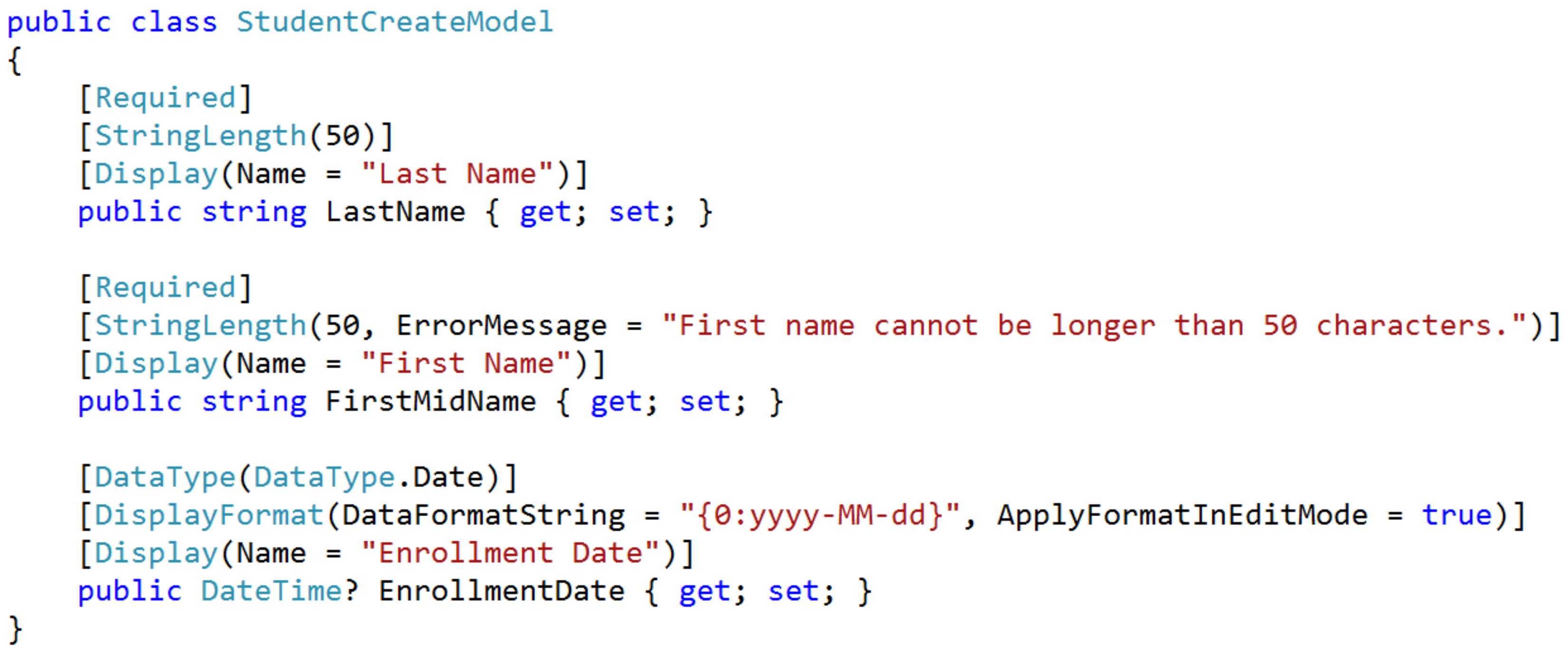


Model Validation

Validation in MVC Core

- Performed through Data Annotation library
- Decorate our model with attributes
- Check model state validation in controller action
- Extensible model

Data Annotation Attributes



```
public class StudentCreateModel
{
    [Required]
    [StringLength(50)]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }

    [Required]
    [StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]
    [Display(Name = "First Name")]
    public string FirstMidName { get; set; }

    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    [Display(Name = "Enrollment Date")]
    public DateTime? EnrollmentDate { get; set; }
}
```

Headspring

Built-In Validation

- **Compare** – validates two properties match
- **CreditCard** – validates credit card format
- **EmailAddress** – validates email format
- **EnumDataType** – validates string is enum value
- **FileExtensions** – validates path is allowed extension
- **MaxLength** – validates length less than or equal value
- **MinLength** – validates length greater or equal than value

Built-In Validation

- **Phone** – validates phone format
- **Range** – validates value is within range, inclusive
- **RegularExpression** – validates value matches expression
- **Required** – validates value is not null
- **StringLength** – validates string is less than or equal to length
- **Url** – validates URL format

Custom Property Validator

```
public class IsCoolNameAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(
        object value,
        ValidationContext validationContext)
    {
        var name = (string) value;

        if (name == "Jimmy")
            return ValidationResult.Success;

        return new ValidationResult("Only Jimmy is cool");
    }
}
```

Custom Model Validator

```
public class LegacyStudentAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(
        object value,
        ValidationContext validationContext)
    {
        var model = (StudentCreateModel) validationContext.ObjectInstance;

        if (model.EnrollmentDate?.Year < 1990)
            return ValidationResult.Success;

        if (string.IsNullOrEmpty(model.FirstMidName))
            return new ValidationResult("First name required for enrollments after 1990");

        return ValidationResult.Success;
    }
}
```

Self-Validating Models

```
public class StudentCreateModel : IValidatableObject
{
    public IEnumerable<ValidationResult> Validate(
        ValidationContext validationContext)
    {
        if (EnrollmentDate?.Year < 1990)
            yield break;

        if (string.IsNullOrEmpty(FirstMidName))
            yield return
                new ValidationResult("First name required for enrollments after 1990");
    }
```

Accessing Services

```
public class IsUniqueNameAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(
        object value,
        ValidationContext validationContext)
    {
        var dbContext = validationContext.GetService<MyDbContext>();
        var student = (StudentCreateModel)validationContext.ObjectInstance;

        if (dbContext.Students.Any(s =>
            s.FirstMidName == student.FirstMidName
            && s.LastName == student.LastName))
        {
            return new ValidationResult("Student must have unique first and last names");
        }

        return ValidationResult.Success;
    }
}
```

Validation in Controller Actions

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(StudentCreateModel model)
{
    if (ModelState.IsValid)
    {
        var student = new Student
        {
            FirstMidName = model.FirstMidName,
            LastName = model.LastName,
            EnrollmentDate = model.EnrollmentDate.GetValueOrDefault()
        };
        db.Students.Add(student);
        db.SaveChanges();
        return RedirectToAction(nameof(Index));
    }
    return View(model);
}
```