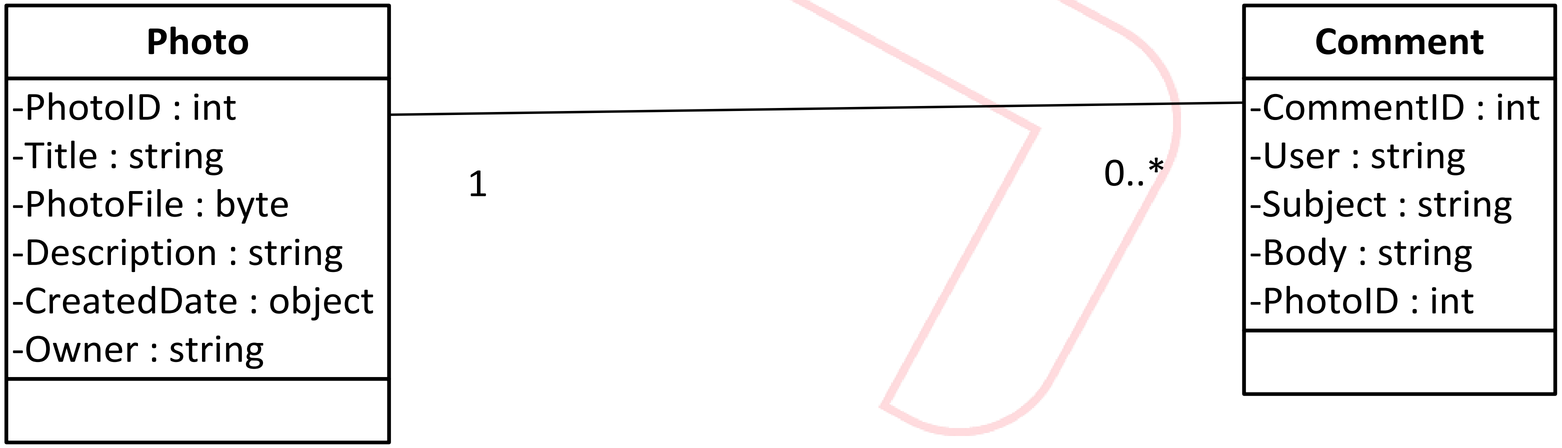
Developing ASP.NET MVC 5 Models

Module Overview

- Creating MVC Models
- Working with Data

Developing Models



```
public class Photo
{
    public int PhotoID { get; set; }
    public string Title { get; set; }
    public byte[] PhotoFile { get; set; }
    public string Description { get; set; }
    public DateTime CreatedDate { get; set; }
    public string Owner { get; set; }
    public virtual ICollection<Comment>
        Comments { get; set; }
}
```

Developing Models (Continued)

```
public class Comment {
        public int CommentID { get; set; }
        public int PhotoID { get; set; }
        public string UserName { get; set; }
        public string Subject { get; set; }
        public string Body { get; set; }
        public virtual Photo Photo { get; set; }
}

    Comment newComment = new Comment();
    newComment.UserName = User.Identity.Name;
    newComment.Subject = "This is an example comment";
    return View("Display", newComment);
```

Using Display and Edit Data Annotations on Properties

```
public class Photo
{
   public int PhotoID { get; set; }
   public string Title { get; set; }
   [DisplayName("Picture")]
   public byte[] PhotoFile { get; set; }
   [DataType(DataType.MultilineText)]
   public string Description { get; set; }
   [DataType(DataType.DateTime)]
   [DisplayName("Created Date")]
   [DisplayFormat(DataFormatString = "{0:MM/dd/yy}",
      ApplyFormatInEditMode = true)]
   public DateTime CreatedDate { get; set; }
   public string UserName { get; set; }
   public virtual ICollection<Comment>
      Comments { get; set; }
}
```

Validating User Input with Data Annotations

```
public class Person
{
   public int PersonID { get; set; }

   [Required(ErrorMessage="Please enter a name.")]
   public string Name { get; set; }

   [Range(0, 400)]
   public int Height { get; set; }

   [Required]
   [RegularExpression(".+\\@.+\\..+")]
   public string EmailAddress { get; set; }
}
```

What Are Model Binders?

- The Default Controller Action Invoker uses model binders to determine how parameters are passed to actions
- The Default Model Binder passes parameters by using the following logic:
  - The binder examines the definition of the action that it must pass parameters to
  - The binder searches for values in the request that can be passed as parameters

Model Extensibility

- Custom validation data annotations can be used to indicate to MVC how to validate the data a user enters in a form or passes in query strings
- There are four built-in validation attributes:
  - Required
  - Range
  - StringLength
  - RegularExpression
- A custom model binder ensures that it identifies parameters in a request and passes all of them to the right parameters on the action

A Custom Validation Data Annotation

```
[AttributeUsage(AttributeTargets.Field)]
public class LargerThanValidationAttribute : ValidationAttribute
{
   public int MinimumValue { get; set; }
   public LargerThanValidationAttribute (int minimum) {
      MinimumValue = minimum;
   }
   public override Boolean IsValid (Object value) {
      var valueToCompare = (int)value;
      if (valueToCompare > MinimumValue) { return true; }
      else { return false; }
   }
}
```

Lesson 2: Working with Data

- Connecting to a Database
- The Entity Framework
- Using an Entity Framework Context
- Using LINQ to Entities
- Demonstration: How to Use Entity Framework Code
- Data Access in Models and Repositories

The Entity Framework

- Types of Entity Framework Workflows
  - Database First
  - Model First
  - Code First

- Adding an Entity Framework Context

```
public class PhotoSharingDB : DbContext
{
    public DbSet<Photo> Photos { get; set; }
    public DbSet<Comment> Comments { get; set; }
}
```

Using an Entity Framework Context

Using the Entity Framework involves:

- Using the Context in Controllers
  - After defining the Entity Framework context and model classes, you can use them in MVC controllers to pass data to views for display
- Using Initializers to Populate Databases:
  - If you are using the code-first or model-first workflow, Entity Framework creates the database the first time you run the application and access data

Using an Entity Framework Context in Controllers

```
public class PhotoController : Controller
{
    private PhotoSharingDB db = new PhotoSharingDB();
    public ActionResult Index()
    {
        return View("Index", db.Photos.ToList());
    }
    public ActionResult Details(int id = 0)
    {
        Photo photo = db.Photos.Find(id);
        if (photo == null) return HttpNotFound();
        return View("Details", photo);
    }
}
```

Using LINQ to Entities

- LINQ to Entities is the version of LINQ that works with Entity Framework
- Sample LINQ Query:

```
photos = (from p in context.Photos
          orderby p.CreatedDate descending
          select p).Take(number).ToList();
```