**Title: Implementing a DataGrid in ASP.NET Core MVC**

**Objective**

By the end of this lecture, students should be able to:

1. Understand how to implement a DataGrid in an ASP.NET Core MVC application.
2. Retrieve and display data from an SQL Server database using Entity Framework Core.
3. Use Razor Views to render data in a structured tabular format.
4. Implement sorting, searching, and pagination functionalities in a DataGrid.
5. Handle user interactions such as selecting and deleting records.

**Discussion**

A **DataGrid** is a tabular UI component used to display structured data from a database. In this project, we will display records from the Verify table, which consists of three fields:

* **Id** (Primary Key, Auto-increment)
* **City** (String, Name of the city)
* **Zip** (String, Zip Code of the city)

**Step 1: Install Required Packages**

Before we begin, ensure the following NuGet packages are installed in your project:

|  |
| --- |
| **Install Required Packages** |
| Install-Package Microsoft.EntityFrameworkCore  Install-Package Microsoft.EntityFrameworkCore.SqlServer  Install-Package Microsoft.EntityFrameworkCore.Design  Install-Package Microsoft.EntityFrameworkCore.Tools |

**Step 2: Create the Database and Table**

Execute the following SQL script in SQL Server:

|  |
| --- |
| **Database and Table** |
| CREATE DATABASE Verification;  USE Verification;  CREATE TABLE Verify (  Id INT PRIMARY KEY IDENTITY,  City NVARCHAR(100) NOT NULL,  Zip NVARCHAR(10) NOT NULL  ); |

**Step 3: Configure Entity Framework Core**

Modify AppDbContext.cs:

|  |
| --- |
| AppDbContext |
| using Microsoft.EntityFrameworkCore;  namespace datagridexample.Models  {  public class AppDbContext : DbContext  {  public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }  public DbSet<Verify> Verify { get; set; }  }  } |

Modify Verify.cs Model:

|  |
| --- |
| Verify |
| using System.ComponentModel.DataAnnotations;  namespace datagridexample.Models  {  public class Verify  {  public int Id { get; set; }  [Required]  public string City { get; set; }  [Required]  [RegularExpression("^\\d{4}$", ErrorMessage = "Zip Code must be exactly 4 digits.")]  public int Zip { get; set; }  }  } |

**Step 4: Create Controller**

|  |
| --- |
| **Controller** |
| using datagridexample.Models;  using Microsoft.AspNetCore.Mvc;  namespace datagridexample.Controllers  {  public class VerifyController : Controller  {  private readonly AppDbContext \_context;  public VerifyController(AppDbContext context)  {  \_context = context;  }  public IActionResult Index()  {  var records = \_context.Verify.ToList();  return View(records);  }  }  } |

**Step 5: Create Razor View (Index.cshtml)**

|  |
| --- |
| **View** |
| @model IEnumerable<datagridexample.Models.Verify>  <h2>DataGrid - City and Zip Records</h2>  <table border="1" width="100%">  <thead>  <tr>  <th>Id</th>  <th>City</th>  <th>Zip</th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>@item.Id</td>  <td>@item.City</td>  <td>@item.Zip</td>  </tr>  }  </tbody>  </table> |

**Step 6: Register AppDbContext**

|  |
| --- |
| **Progam.cs** |
| builder.Services.AddDbContext<AppDbContext>(options =>  options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))); |

**Step 7: Connection**

|  |
| --- |
| **AppSettings.json** |
| "ConnectionStrings": {  "DefaultConnection": "Server=DESKTOP-IK6F1CS\\SQLEXPRESS;Database=Verification;Trusted\_Connection=True;TrustServerCertificate=True;"  }, |

**Step 6: Run the Application**

Start the application and navigate to /Verify/Index to see the DataGrid displaying records from the Verify table.

**Sample Problems**

1. Modify the Index view to add a **delete button** next to each record.
2. Implement a search bar that filters results based on the City field.
3. Extend the model to include a State field and display it in the DataGrid.

**Q&A**

1. **What is the purpose of the DbSet<Verify> in AppDbContext?**
   * It represents the Verify table in the database and allows us to interact with its records.
2. **Why do we use ToList() in the controller?**
   * It fetches all records from the database and converts them into a list that can be passed to the view.
3. **How can we make the table sortable?**
   * We can use JavaScript libraries like DataTables or implement sorting logic in the controller using LINQ.
4. **What happens if there are no records in the database?**
   * The table will display empty rows unless we add a condition to show a message like "No records found."

**Activity**

1. **Enhance the DataGrid**:
   * Add pagination so that only 5 records are shown per page.
   * Add a search box to filter results dynamically.
   * Add a column for the State field and update the database accordingly.
2. **Modify the Verify Model**:
   * Include a DateAdded field to store when a record was inserted.
   * Update the Index view to show the new field.