Lab 03: Truyền dữ liệu Controller – View

1 Mục tiêu

- ✓ Làm quen với mô hình MVC.
- ✓ Cú pháp Razor & Tag Helpers.
- ✓ Truyền dữ liệu giữa Controller với View và ngược lại

2 Các bước thực hiện

2.1 Đọc ghi file

❖ **MỤC TIÊU**

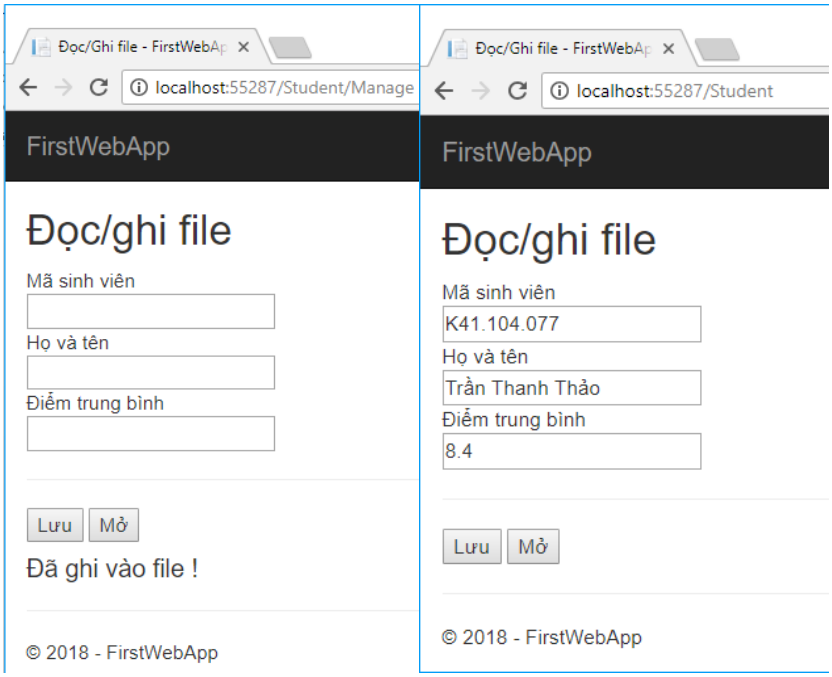
Kết thúc bài thực hành, bạn có khả năng:

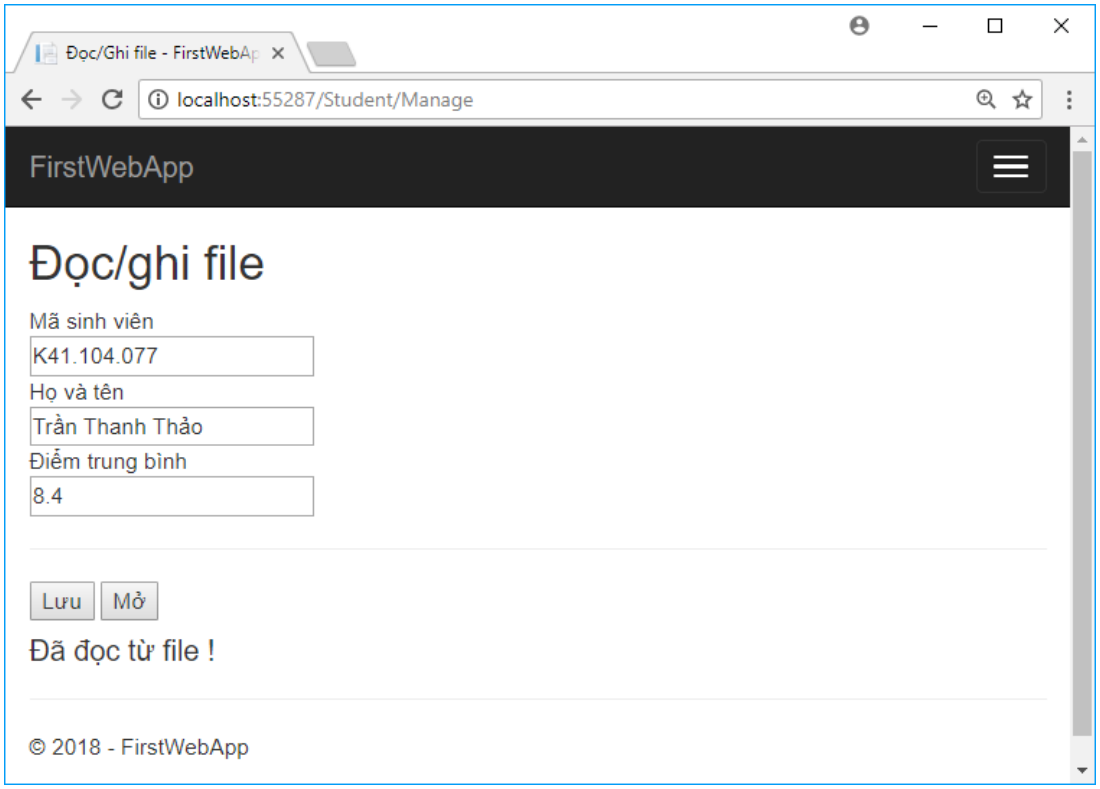
- Kết hợp đối số và model để nhận tham số
- Đọc/ghi mảng từ/vào file văn bản

❖ **MÔ TẢ**

Cụ thể trong bài này bạn phải xây dựng trang web có hình minh họa sau. Sau khi nhập dữ liệu vào form và nhấp nút [**Lưu**] ứng dụng sẽ lưu thông tin vào file có tên **Student.txt** đặt tại thư mục gốc của website (wwwroot).

Sau khi đã lưu 1 lần, bạn có thể đọc dữ liệu từ file và hiển thị lên form bằng cách nhấp nút [**Mở**].





❖ THỰC HIỆN

Để thực hiện ứng dụng trên, bạn cần thực hiện theo các bước sau:

Bước 1: Tạo controller **StudentController**.cs

```
public class StudentController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

Bước 2: **Tạo giao diện form nhập**

Phải chuột vào action Index() và tạo view có mã razor như sau. Mã gồm 1 form có action gọi đến action Manage() của controller StudentController và chuyển các trường Id, Name, Marks và nút command được nhấp đến action này.

Form cũng hiên thị các thuộc tính Id, Name, Marks và Message của ViewBag được chuyển từ controller lên các trường form và thông báo cuối form.

```
@{
    ViewData["Title"] = "Đọc/Ghi file";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
<h2>Đọc/ghi file</h2>

<form asp-action="Manage" asp-controller="Student" method="post">
    <div>Mã sinh viên</div>
    <input name="Id" value="@ViewBag.Id" />

    <div>Họ và tên</div>
    <input name="Name" value="@ViewBag.Name" />

    <div>Điểm trung bình</div>
    <input name="Marks" value="@ViewBag.Marks" />
    <hr />
    <input type="submit" value="Lưu" name="command" />
    <input type="submit" value="Mở" name="command" />
</form>

<h4>@ViewBag.Message</h4>
```

Bước 3: Tạo lớp model **StudentInfo.cs**

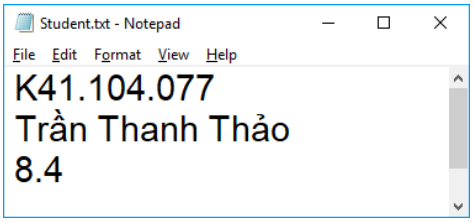
Lớp model này được sử dụng để nhận dữ liệu từ form. Các tham số sẽ chuyển vào các thuộc tính cùng tên của model.

```
public class StudentInfo
{
    public string Id { get; set; }
    public string Name { get; set; }
    public double Marks { get; set; }
}
```

Bước 4: Bổ sung action **Manage**() vào controller để điều khiển hành động [Lưu] và [Mở]

Action Manage() sử dụng model để nhận thông tin nhân viên và đối số command để nhận nút submit bị nhấp. Action sẽ phân biệt 2 trường hợp dựa vào giá trị của nút nhấn.

Nếu nhấn nút [Lưu] thì thực hiện lưu model vào file. Phương thức System.IO.File.WriteAllLines(path, lines) sẽ lưu mảng vào file. Mỗi phần tử mảng lưu trên một hàng.



Nếu nhấn nút [Mở+ thì đọc dữ liệu từ file và truyền cho view thông qua các thuộc tính Id, Name và Marks của ViewBag. Phương thức System.IO.File.ReadAllLines(path) giúp đọc mảng chuỗi từ file. Cứ mỗi hàng sẽ đọc thành 1 phần tử của mảng.

Bổ sung action Manage() để mở và đọc file:

```
public ActionResult Manage(StudentInfo model, String command)
{
    var path = Path.Combine(Directory.GetCurrentDirectory(),
                    "wwwroot", "Student.txt");

    if (command == "Lưu")
    {
        String[] lines = {model.Id, model.Name, model.Marks.ToString()};
        System.IO.File.WriteAllLines(path, lines);
        ViewBag.Message = "Đã ghi vào file !";
    }
    else if (command == "Mở")
    {
        String[] lines = System.IO.File.ReadAllLines(path);
        ViewBag.Id = lines[0];
        ViewBag.Name = lines[1];
        ViewBag.Marks = Convert.ToDouble(lines[2]);

        ViewBag.Message = "Đã đọc từ file !";
    }
    return View("Index");
}
```

Bước 5: Chạy ứng dụng

- Chạy http://localhost:55287/Student/Index
- Nhập thông tin và nhấp nút *Lưu+ sau đó kiểm tra thông tin của file được tạo ra ở thư mục gốc của website
- Nhấp nút [Mở+ để hiển thị lại thông tin đã nhập

2.2 Upload file

Kết thúc bài thực hành, bạn có khả năng:

- Tạo form upload file
- Tiếp nhận file upload và lưu vào thư mục với tên file gốc
- Hiển thị thông tin file upload

Bước 1: Tạo **FileUploadController**

```
public class FileUploadController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

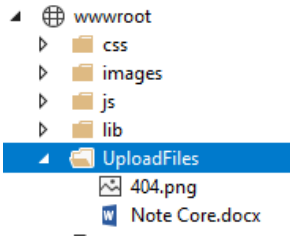
Bước 2: **Tạo form upload file**

Ở View Index() này sẽ thiết kế 2 form riêng biệt dùng để upload một file, upload nhiều file. Đặc biệt form upload luôn luôn phải thiết lập giá trị của thuộc tính method là POST và enctype là **multipart/form-data**.

```
@{
    ViewData["Title"] = "Upload file";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Upload file</h2>

<form asp-controller="FileUpload" asp-action="UploadFile" method="post"
      enctype="multipart/form-data" class="form-inline">
    <h4>Upload single-file</h4>
    <input type="file" name="file" />
    <button type="submit">Upload File</button>
</form>
<hr />
<form asp-controller="FileUpload" asp-action="UploadFiles" method="post"
      enctype="multipart/form-data">
    <h4>Upload multi-files</h4>
    <input type="file" name="files" multiple />
    <button type="submit">Upload Files</button>

</form>
```

Bước 3: Bổ sung các action `UploadFile(), UploadFiles(),UploadFileViaModel()` vào controller.

Tất cả các tập tin upload lên đều lưu vào thư mục **UploadFiles** trong **wwwroot**.




```
        [HttpPost]
        public async Task<IActionResult> UploadFile(IFormFile file)
        {
            if (file == null || file.Length == 0)
                return Content("file not selected");

            var path = Path.Combine(
                    Directory.GetCurrentDirectory(), "wwwroot", "UploadFiles",
                    file.GetFilename()
            );
```

```
            using (var stream = new FileStream(path, FileMode.Create))
            {
                await file.CopyToAsync(stream);
            }

            return RedirectToAction("ListFiles");
        }
```

```
        [HttpPost]
        public async Task<IActionResult> UploadFiles(List<IFormFile> files)
        {
            if (files == null || files.Count == 0)
                return Content("files not selected");

            foreach (var file in files)
            {
                var path = Path.Combine(
                   Directory.GetCurrentDirectory(), "wwwroot", "UploadFiles",
                   file.GetFilename()
                );

                using (var stream = new FileStream(path, FileMode.Create))
                {
                    await file.CopyToAsync(stream);
                }
            }

            return RedirectToAction("ListFiles");
        }
```

Tất cả các action trên sau khi xử lý xong đều gửi tới action ListFiles() để hiển thị danh sách file trong thư mục chỉ định thông qua lệnh `return RedirectToAction("ListFiles");`

Chú ý:

- ✓ Action UploadFile() để xử lý upload một file thì tham số truyền vào là biến đơn: `IFormFile file`, file chính là tên control input[type=file].
- ✓ Action UploadFiles() để xử lý upload nhiều file thì tham số truyền vào là danh sách: `List<IFormFile> files`, files chính là tên control input[type=file].
- ✓ Action UploadFileViaModel() xử lý upload file qua model cần truyền vào biến model: `FileInputModel model`, trong đó lớp FileInputModel định nghĩa như sau:

```
public class FileInputModel
{
   public IFormFile FileToUpload { get; set; }
 }
```

và FileToUpload chính là tên của control input[type=file]

Bước 4: Bổ sung action ListFile()

```
        public IActionResult ListFiles()
        {
            var model = new FilesViewModel();
             string[] filePaths
      = Directory.GetFiles(Path.Combine(Directory.GetCurrentDirectory(), "
      UploadFiles/"));
            foreach (var item in filePaths)
            {
                model.Files.Add(
                    new FileDetails { Name = item, Path =
Path.GetFileName(filePath); });
            }

            return View(model);
        }
```

Model chuyển qua chứa thông tin tên file và đường dẫn. Do đó cần định nghĩa thêm các lớp để lưu các thông tin này.

```
    public class FileDetails
    {
        public string Name { get; set; }
        public string Path { get; set; }
    }

    public class FilesViewModel
    {
        public List<FileDetails> Files { get; set; } = new List<FileDetails>();
    }
```

Nội dung view ListFiles.cshtml dùng để hiển thị thông tin file có trong thư mục như sau:

ListFiles.cshtml*

```
@model FirstWebApp.Models.FilesViewModel

@{
    ViewData["Title"] = "ListFiles";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>List of Files</h2>

<ul>
    @foreach (var item in Model.Files)
    {
        <li>
            <a asp-action="Download" asp-route-filename="@item.Name">
                @item.Name
            </a>
        </li>
    }
</ul>
```

Bước 5: Thực nghiệm