Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем

ОТЧЕТ

ПО ИНДИВИДУАЛЬНОЙ ПРАКТИЧЕСКОЙ РАБОТЕ №2

по дисциплине
«Современные технологии проектирования информационных систем»

специальности 1-40 05 01-10 «Информационные системы и технологии (в бизнес-менеджменте)»

Выполнил:
студент группы №994351
Богомаз Д. Л.

Проверила: Левченя Ж.Б.

Минск 2022

Цель: создать программные средства мониторинга и консультативной поддержки процесса изучения дисциплин вуза, реализовать паттерн проектирования (в данном случае factory method).

В данной работе выполняется вариант номер 3: Программные средства мониторинга и консультативной поддержки процесса изучения дисциплин вуза.

Ссылка на проект: https://github.com/bogomazdmitry/LmsCopy

Теоретические сведенья

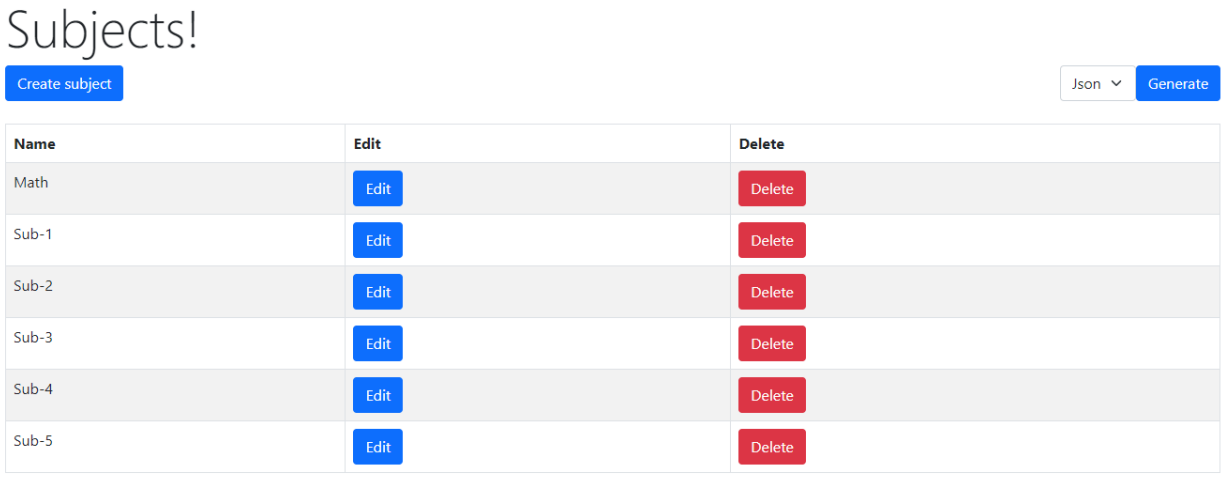
Фабричный метод — это порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

Когда надо применять паттерн

- Когда заранее неизвестно, объекты каких типов необходимо создавать
- Когда система должна быть независимой от процесса создания новых объектов и расширяемой: в нее можно легко вводить новые классы, объекты которых система должна создавать.
- Когда создание новых объектов необходимо делегировать из базового класса классам наследникам.

Реализация

В проекте с помощью данного паттерна реализовано создание ответов предметов. Создан выпадающий список с типом генерируемого файла:



После выбора типа файла и нажатия на кнопку Generate вызовется метод:

[HttpGet]

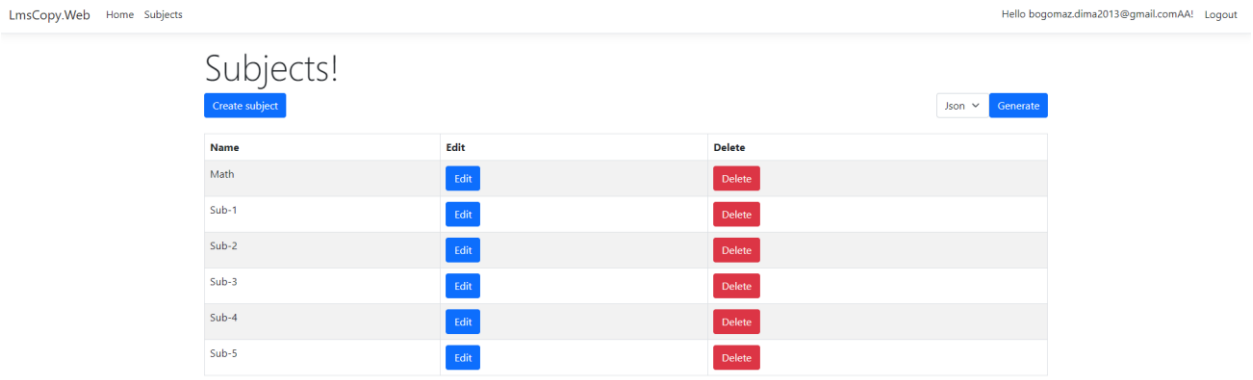
```
[Authorize(Roles = UserRole.Professor)]
public IActionResult GenerateReport(String reportType)
{
   var fileReport =
FileReportFactory.GetServiceIndexRequest(reportType,
_settings.SubjectReportName);

   var file =
fileReport.GenerateReport<Subject>(_context.Subjects.ToList());
   return File(file.Data, file.ContentType, file.FileName);
}
```

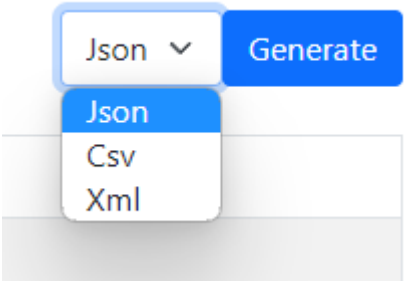
По строке будет возвращен сервис по генерации отчета для конкретного типа. Сам выбор сделан с помощью pattern matching:

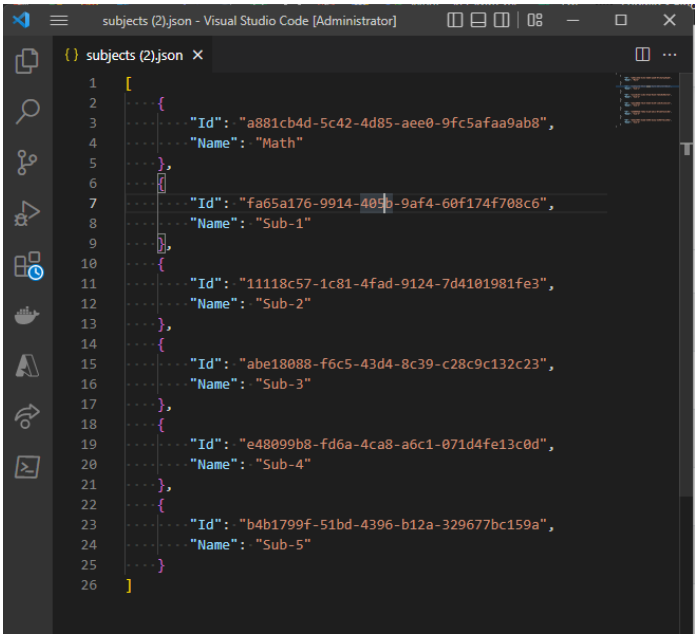
```
public static IFileReport GetServiceIndexRequest(string fileType, string
fileName)
{
   return fileType switch
   {
      "Json" => new JsonFileReport{ FileName = fileName},
      "Xml" => new XmlFileReport{ FileName = fileName},
      "Csv" => new CsvFileReport{ FileName = fileName},
   };
}
```

Таким образом в зависимости от типа файла, выбранного пользователя, будет создан соответствующий объект (все объекты унаследованы от интерфейса IFileReport). После чего обращение внутри метода контроллера происходит через интерфейс.

Снимки экрана приложения







Листинг кода

CsvFileReport:

```
using CsvHelper;
using LmsCopy.Web.Models;
using System.Globalization;
using System.Text;

namespace LmsCopy.Web.Services;

public class CsvFileReport : IFileReport
{
    public string FileName { get; set; }

    public FileModel GenerateReport<T>(List<T> items)
```

```
    {
        using var writer = new StringWriter();
        using var csv = new CsvWriter(writer, CultureInfo.InvariantCulture);

        csv.WriteRecords(items);

        return new FileModel()
        {
            Data = Encoding.ASCII.GetBytes(writer.ToString()),
            ContentType = "text/plain",
            FileName = $"{FileName}.csv"
        };
    }
}
```

JsonFileReport:

```
using LmsCopy.Web.Models;
using System.Text;
using System.Text.Json;

namespace LmsCopy.Web.Services;

public class JsonFileReport : IFileReport
{

    public string FileName { get; set; }

    public FileModel GenerateReport<T>(List<T> items)
    {
        var dataString = JsonSerializer.Serialize(items);
        return new FileModel()
        {
            Data = Encoding.ASCII.GetBytes(dataString),
            ContentType = "text/plain",
            FileName = $"{FileName}.json"
        };
    }
}
```

XmlFileReport:

```
using LmsCopy.Web.Models;
```

```
using System.Text;
using System.Xml.Serialization;

namespace LmsCopy.Web.Services;

public class XmlFileReport : IFileReport
{
   public string FileName { get; set; }

   public FileModel GenerateReport<T>(List<T> items)
   {
      var serializer = new XmlSerializer(typeof(List<T>));
      using var writer = new StringWriter();

      serializer.Serialize(writer, items);
      return new FileModel()
      {

         Data = Encoding.ASCII.GetBytes(writer.ToString()),
         ContentType = "text/plain",
         FileName = $"{FileName}.xml"
      };
   }
}
```

Вывод: фабричный метод является простым паттерном, который помогает реализовывать базовые функции изменения поведения в зависимости от входных параметров. Также хорошо сочетается с таким паттерном, как стратегия, однако не в web-приложениях, а, например, в консольных. Фабричный метод реализован на языке программирования C#.