Лабораторная работа №3

Основы работы с XML/Json на платформе .NET

Введение:

В предыдущей лабораторной работе вы разработали сервис для мониторинга директорий и манипуляции с файлами (архивирование, шифрование, передача). Но все конфигурируемые свойства, например, SourceFolder, TargetFolder, NeedToArchive, CompressingLevel и т.д., были захардкожены, или в лучшем случае вводились с клавиатуры. Пришло время это исправить!

Постановка задачи:

Необходимо реализовать переносимую/изменяемую систему конфигурации сервиса (далее «джоба») в формате XML **И** Json.

XML: Реализация конфигурационного файла config.xml, возможность валидации с использованием config.xsd. В случае отсутствия файла валидации – использование «сырого» config.xml.

Json: Реализация конфигурационного файла appsettings.json.

Требования:

1. Набор необходимых конфигурационных свойств определяется исходя из количества фич, заимплеменченых в предыдущей лабораторной работе (чем **больше**, тем **выше оценка**).
2. ВСЕ видимые исключительные ситуации должны быть обработаны и **залогированы** (последующее поведение приложения (после возникнования исключительной ситуации) определить самостоятельно исходя из **здравого смысла**).
3. **Обработчик XML** файла должен быть **расширеямым**, т.е. имелась **возможность добавления новых конфигурационных** свойств (с соответствующим **изменением модели** на стороне .NET)
4. **Обработчик Json** файла должен быть **расширяемым**, т.е. имелась **возможность добавления новых конфигурационных** свойств (при, например, соответствующем **изменением модели** на стороне .NET или её **переработке**).
5. **Содержимому** конфигурационных файлов должна **быть предоставлена соответствующая модель на стороне .NET**, именование модели будет следующим **Etl[Xml\Json]Options**.

6. Для **каждого логически отдельного** блока конфигураций должна быть соответствующая **модель на стороне .NET**, например ArchiveOptions, CryptingOptions и т.д.
7. **Обязательно наличия менеджера конфигурации**, который при наличии определённого типа конфигурационного файла (XML/JSON) будет принимать решение о том, какой конфигурационный файл использовать. **Примечание**: API менеджера представляет собой как минимум 1 метод, являющийся дженериком, который будет принимать на вход какой-либо тип и в соответствии с ним возвращать наполненный объект конфигурации или бросать эксепшион (в самом простом варианте). Помимо всего прочего, должна быть реализована возможность доставать конфигурацию не только целиком из всего файла, а кусками (секциями) передавая соответственно, либо полную модель с вложенными в неё другими моделями, либо отдельно модели соответствующие секциям, как показано в примере ниже.

```
            var configOptions = optionsManager.GetOptions<ArchiveOptions>();
        }
    }

    1 reference | 0 changes | 0 authors, 0 changes
    public class ArchiveOptions
    {

    }
```

8. Менеджер конфигурации получает на вход путь к файлу (желательно попробовать определять его программно, с использованием AppDomain и т.д., но необязательно, т.к. оба варианта имеют место).
9. Ваша джоба должна «напихиваться» конфигом и далее запускаться.

ИСПОЛЬЗОВАНИЕ DI-контейнеров ЗАПРЕЩЕНО

10. Обязательно предоставить следующую возможность – расширение модели конфигурации (например добавление нового поля в класс ArchiveOptions) и соответствующее расширение (или его отсутствие) конфигурационного файла (добавление нового тэга/секции в xml или объекта/поля в json) не должно повлеч за собой изменение кода обработчика. Ваш обработчик должен быть универсальным хотя бы в рамках типа файла.
11. Наименования свойств модели должны совпадать с наименованием тэгов/секций в XML/JSON файлах. **Примечание:** для повышения оценки разрешается использование аттрибутов, с помощью которых, например, в случае с JSON можно проводить валидацию, а так же давать свойствам

модели псевдонимы, по которым обработчик будет их искать в конфигурационных файлах.