OTUS

C# Developer. Professional

Best Practice по разработке на C# и .NET Framework с практикой Scrum-методики

Длительность курса: 134 академических часа

1 C# Advanced

1 Знакомство, рассказ о формате Scrum, краткий обзор курса

Цели занятия:

познакомиться.

знакомство с командой курса;
знакомство со студентам;
цели курса;
формат Scrum-команд;
краткий обзор модулей.

Домашние задания

1 Знакомимся с командой и гит репозиторием

Цель: В этом ДЗ мы прорабатываем организационные моменты и лучше знакомимся друг с другом.

Для всех:
1) Зарегистрироваться на Гитлабе и сделать коммит в общий проект, при сдаче ДЗ необходимо приложить ссылку на коммит в gitlab репозитории; Если присоединились к курсу после первого занятия, то нужно написать в slack-канал потока о том, что начали заниматься на курсе после первого занятия и хотели бы присоединиться к scrum-команде, можно продублировать сообщение руководителю программы Алексею Ягур (Aleksey Yagur) в slack, он с радостью поможет найти подходящую команду.
2) Написать “о себе” в закрытой группе команды, при сдаче ДЗ также приложить ссылку на сообщение в Slack;
3) Написать три способа траты баллов.

Для SCRUM-мастеров:
4) Выбрать название для команды;
5) Создать закрытый чат команды в слаке (добавить в него @Alex Yagur);
6) Создать проект на Гитлабе;
7) Выбрать проект для команды;
8) Создать доски задач: беклог, в работе, сделано;
9) Наполнить верхнеуровневый беклог.

2 **Архитектура проекта**

Цели занятия:

формировать архитектуру приложения;
разделять код на слои и звенья;
принимать более взвешенные решения о структуре проекта.

слои;
звенья;
интерфейсы;
API.

---

3 **Базы данных: организация работы с потоками данных**

Цели занятия:

объяснить основы работы с базами данных.

SQL базы данных;
принципы работы SQL баз данных;
отличие NO SQL от SQL баз данных;
ACID;
ORM.

Домашние задания

1 Подключаем базы данных к проекту

Цель: В этом домашнем задании вы научитесь создавать базу данных с таблицами, а также писать скрипты наполнения таблиц данными. Но самое главное - вы создадите приложение, способное получать данные из базы и добавлять новые.

1. Создать базу данных PostgreSQL для одной из компаний на выбор: Авито, СберБанк, Otus или eBay. Написать скрипт создания 3 таблиц, которые должны иметь первичные ключи и быть соединены внешними ключами.
2. Написать скрипт заполнения таблиц данными, минимум по пять строк в каждую.
3. Создать консольную программу, которая выводит содержимое всех таблиц.
4. Добавить в программу возможность добавления в таблицу на выбор.

---

4 **Базы данных: реляционные базы и работа с ними**

Цели занятия:

научиться работать с реляционными базами данных.

Dapper;
Multiselect, одновременная нагрузка;
архитектура в oltp/olap нагрузке и как нужно когда строить;
в каких случаях использовать for json, а в каких api;
когда нужно делать все на базе, а когда можно и entity.

---

5 **Базы данных: NoSQL базы и их особенности**

Цели занятия:

научиться работать с NoSQL базами данных

история происхождения;
основные черты;
типы систем:
- ключ-значение
- семейство столбцов
- документоориентированная
- графовая

LiteDB как встроенная NoSQL база данных.

---

6 **Ретроспектива и планирование**

---

7 **Отражение (Reflection)**

Цели занятия:

обрабатывать экземпляры разных или неизвестных заранее классов;
создавать универсальные тесты;
писать свои компоненты.

использование механизмов доступа к параметрам типов через механизмы отражения;
динамическая загрузка сборок и позднее связывание.

Домашние задания

1 Рефлексия и её применение

Цель: Написать свой класс-сериализатор данных любого типа в формат CSV, сравнение его быстродействия с типовыми механизмами серализации.
Полезно для изучения возможностей Reflection, а

может и для применения данного класса в будущем.

Основное задание:
1. Написать сериализацию свойств или полей класса в строку
2. Проверить на классе: class F { int i1, i2, i3, i4, i5; Get() => new F(){ i1 = 1, i2 = 2, i3 = 3, i4 = 4, i5 = 5 }; }
3. Замерить время до и после вызова функции (для большей точности можно сериализацию сделать в цикле 100-100000 раз)
4. Вывести в консоль полученную строку и разницу времен
5. Отправить в чат полученное время с указанием среды разработки и количества итераций
6. Замерить время еще раз и вывести в консоль сколько потребовалось времени на вывод текста в консоль
7. Провести сериализацию с помощью каких-нибудь стандартных механизмов (например в JSON)
8. И тоже посчитать время и прислать результат сравнения
9. Написать десериализацию/загрузку данных из строки (ini/csv-файла) в экземпляр любого класса
10. Замерить время на десериализацию
11. Общий результат прислать в чат с преподавателем в системе в таком виде:

Сериализуемый класс: class F { int i1, i2, i3, i4, i5;}
код сериализации-десериализации: ...
количество замеров: 1000 итераций
мой рефлекшен:
Время на сериализацию = 100 мс
Время на десериализацию = 100 мс
стандартный механизм (NewtonsoftJson):
Время на сериализацию = 100 мс
Время на десериализацию = 100 мс

---

8 **Сериализация**

Цели занятия:

использовать механизмы сериализации и результирующие форматы сериализации;
применять стандартные способы сериализации.

что такое сериализаци?;
какие бывают форматы, их плюсы и минусы;
интерфейс ISerializable и IFormatter;
стандартные средства сериализации.

---

9 **Атрибуты**

Цели занятия:

использовать существующие атрибуты;
проверять наличие атрибутов на классах, функциях, полях, и т.д.;
создавать свои собственные атрибуты.

что такое атрибут?;
как применяются атрибуты?;
где они уже используются?;
стандартные и кастомные атрибуты;
как создавать свои атрибуты?

---

10 **Исключения и нюансы работы с ними**

Цели занятия:

объяснить, что из себя представляет исключение;
разобраться как исключение устроено "под капотом";
научиться бросать и перехватывать исключения;
познакомиться с Best Practice в работе с исключениями.

исключение;
иерархии исключений;
перехват исключений;
обработка исключений.

---

11 **Работа с методами как с переменными (delegates, events)**

Цели занятия:

создавать делегаты для передачи методов в функции;
создавать события и подписки на них;
писать менее связный код и упростить расширение функционала ПО.

делегаты и их использование;
применение событий;
отличия событий и делегатов;
стандартизация событий в .NET

Домашние задания

1 Делегаты и события

Цель: В этом задании требуется реализовать механизмы делегатов и событий для получения практического навыка их применения

1. Написать обобщённую функцию расширения, находящую и возвращающую максимальный элемент коллекции.
Функция должна принимать на вход делегат, преобразующий входной тип в число для возможности поиска максимального значения.
public static T GetMax<T>(this IEnumerable<T> e, Func<T, float> getParameter) where T : class;
2. Написать класс, обходящий каталог файлов и выдающий событие при нахождении каждого файла;
3. Оформить событие и его аргументы с использованием .NET соглашений:
public event EventHandler<FileArgs> FileFound;
FileArgs – будет содержать имя файла и наследоваться от EventArgs
4. Добавить возможность отмены дальнейшего поиска из обработчика;
5. Вывести в консоль сообщения, возникающие при срабатывании событий и результат поиска максимального элемента.

12 **Дженерики, их реализация и ограничения**

Цели занятия:

работать с обобщениями.

обобщённые методы;
обобщённые классы:
значения по умолчанию: default(T);
несколько обобщённых параметров: class Transaction<U, V>.
во что разворачиваются в момент выполнения;
ограничения универсальных типов:
стандартные ограничения: class, struct, new;
гибкие ограничения: классы и интерфейсы.

---

13 **Сборщик мусора, деструкторы и финализаторы, Disposable Pattern**

Цели занятия:

разобраться с тем, как хранятся объекты в памяти в .NET;
познакомиться с алгоритмом выделения физической памяти для приложений;
понять алгоритм работы Сборщика Мусора (поколения, стратегии, карточный стол);
начать отличать Деструкторы от Финализаторов;
научиться использовать Disposable Pattern.

как выделяется память;
сборщик мусора;
деструкторы и финализаторы;
Disposable Pattern.

---

14 **Дополнительные возможности языка: от директив препроцессора до указателей**

Цели занятия:

писать небезопасный код и создавать указатели, если, столкнётесь с тем, что это понадобится;
использовать механизмы условной компиляции, чтобы обеспечить зависимость поведения проекта от окружения или обеспечить большее удобство работы с кодом;
применять динамические объекты и заготовки кода, чтобы ускорить написание проектов.

директивы препроцессора;
Unsafe code, pointers;
DLR - Dynamic & ExpandoObject.

---

15 **Что полезного в новых версиях C#?**

Цели занятия:

проанализировать нововведения в стандартах 4.7 - 4.8

ключевые нововведения .Net 4.8;
полезные и часто используемые новшества .Net 4.7.2.

---

16 **Ретроспектива и планирование**

Цели занятия:

уверенно участвовать в ретроспективах.

проведение ретроспективы прошедшего спринта.

2 Многопоточность и шаблоны проектирования

1 **Введение в параллелизм в .NET. Отличия процесса, потока, домена и таска**

Цели занятия:

дать описание разным примитивам параллелизма для лучшего понимания их назначения и отличий;
получить примеры практического использования инструментов параллелизма на практической задаче.

определения и понятия параллелизма с точки зрения практической задачи по обработке очереди уведомлений;
описание процессам и потокам ОС, а также рассмотрим их применение в .NET;
работа доменов приложений и размещением CLR в .NET и .NET Core;
понятия синхронности и асинхронности, а также механизм Task.

Домашние задания

1 Параллельная загрузка данных из файла

Цель: Сделать параллельный обработчик файла с данными клиентов на основе подготовленного проекта с архитектурой. Задание поможет отработать основные инструменты параллелизма на реалистичной задаче.

Программа-обработчик должна в параллельном режиме обработать файл с данными клиентов.

Каждая строка файла содержит:

id (целое число)
ФИО (строка),
Email (строка)
Телефон (строка).

Данные отсортированы по id. Нужно десериализовать данные клиента в объект и передать объект в метод класса, который сохраняет его в БД, вместо сохранения в БД можно сделать просто задержку.

Задания

1. Запуск генератора файла через создание процесса, сделать возможность выбора в коде,

как запускать генератор, процессом или через вызов метода. Если вдруг встретится баг с генерацией, то его нужно исправить и написать об этом при сдаче работы.
2. Распараллеливаем обработку файла по набору диапазонов Id, то есть нужно, чтобы файл разбивался на диапазоны по Id и обрабатывался параллельно через Thread, сколько диапазонов столько потоков. Хорошо сделать настройку с количеством потоков, чтобы можно было настроить оптимальное количество потоков под размер файла с данными. Предусмотреть обработку ошибок в обработчике и перезапуск по ошибке с указанием числа попыток. Проверить обработку на файле, в котором 1 млн. записей, при сдаче задания написать время, за которое был обработан файл и количество потоков.
3. Вместо создания потоков через new Thread() использовать ThreadPool, при сдаче задания написать время, за которое был обработан файл и количество потоков.
4. Добавить сохранение в реальную БД, можно SQL Lite для простоты тестирования или для лучшего понимания специфики загрузки полноценную базу данных (MS SQL Server, PostgreSQL, Mongo и т.д.)
5. Сделать обработку файла в CSV формате, то есть написать генератор и разбор файла.
6. Дать обратную связь по 2-м домашним заданиям других студентов на курсе.

Инструкция

1. Сделать форк репозитория из ссылки в материалах, можно изменить структуру проектов, классов и интерфейсов, как считаете нужным и в ReadMe.md описать за что отвечает проект, класс, интерфейс.
2. Реализовать 1 пункт задания, сделав в main проекта запуск процесса-генератора файла, его нужно будет собрать отдельно и передать в программу путь к .exe файлу, также сделать в Main вызов кода генератора из подключенного проекта, выбор между процессом или вызовом метода сделать настройкой (например аргумент командной строки или файл с настройками) со значением по умолчанию для метода.
3. Реализовать 2 пункт задания, сделав свои реализации для IDataLoader и IDataParser.
4. По желанию реализовать 3 пункт задания, сделав дополнительную реализацию IDataLoader.
5. По желанию реализовать 4 пункт задания,

сделав дополнительную реализацию для ICustomerRepository и инициализацию БД при старте приложения, можно использовать EF.
6. По желанию реализовать 5 пункт задания, сделав дополнительную реализацию для IDataParser и IDataGenerator.
7. По желанию дать обратную связь по 2-м домашним заданиям других студентов на курсе, можно найти репозитории по форкам к этому репозиторию. Обратную связь можно описать, создав issue к репозиторию, пример обратной связи можно посмотреть из ссылки на проект в материалах, который рассматривается в рамках занятия. Чтобы обратная связь была качественной обязательно нужно похвалить работу, написав, что сделано хорошо и написать, что можно улучшить с пояснениями почему это сделает работу более качественной. Эти рекомендации работают и для code review, так как позволяют более конструктивно обсуждать коммиты.

---

2 **Асинхронные операции**

Цели занятия:

писать асинхронный код.

пул потоков;
машина состояний;
отличие асинхронного подхода от конкурентного;
async/await.

---

3 **Магические слова async / await**

Цели занятия:

изучить механизм, скрытый под ключевыми словами async/await;
научиться правильному использованию этих ключевых слов.

жизненный цикл Task;
что такое контекст синхронизации;
разбор использования ConfigureAwait(false);
синхронная асинхронность.

---

4 **Синхронизация доступа к общему ресурсу**

Цели занятия:

познакомиться с примитивными и гибридными средствами управления доступа к общим ресурсам;
рассмотреть дополнительные инструменты для упрощения организации многопоточной работы;
применить полученные знания в практическом занятии.

Monitor и lock;
Mutex;
Semaphore;
ReaderWriterLock;
SpinLock;
Slim-версии примитивов.

---

5 **Взаимодействие потоков**

Цели занятия:

познакомиться с примитивами управления потоков;
понять в каких моментах использовать эти примитивы;
применить примитивы на практике.

EventWaitHandle, AutoResetEvent и ManualResetEvent;
CountdownEvent;
Barrier;
Interlocked;
SpinWait.

---

6 **Внутрипроцессное взаимодействие**

Цели занятия:

применять потоки, задачи, Parallel LINQ;
распараллеливать расчёты для ускорения вычислений;
оценивать целесообразность применения механизмов;
параллельной обработки данных.

потоки в операционной системе (на примере Windows API);
поток (Thread);
задача (Task);
Parallel For;
Parallel LINQ.

Домашние задания

1 Многопоточный проект

Цель: Применение разных способов распараллеливания задач и оценка оптимального способа реализации.

1) Напишите вычисление суммы элементов массива интов:
1а. Обычное
1б. Параллельное (для реализации использовать Thread, например List<Thread>)
1в. Параллельное с помощью LINQ

2) Замерьте время выполнения для 100 000, 1 000 000 и 10 000 000 элементов, укажите:
- Окружение
- Код последовательного вычисления и время его выполнения
- Код параллельного вычисления и время его выполнения
- Код LINQ и время его выполнения

Добавьте результаты для всех 3 замеров в таблицу:
https://docs.google.com/spreadsheets/d/1WQwijkBc_BGmKRikTBoO7hLw8EX4yHgQyI_0aTLgB7c/edit?usp=sharing

3) Пришлите в чат с преподавателем помимо ссылки на репозиторий номера своих строк в таблице.

7 **Межпроцессное взаимодействие**

Цели занятия:

изучить способы взаимодействия процессов/ программ друг с другом.

общая память;
сокеты;
именованные каналы;
WCF, REST, etc.
общие файлы;
база данных;
очереди сообщений.

---

8 **Порождающие шаблоны проектирования**

Цели занятия:

объяснить назначение изученных шаблонов проектирования;
применять их в своих проектах.

шаблоны проектирования Singleton, Prototype, Factory method, Abstract Factory и Builder.

Домашние задания

1 Реализуем паттерн "Прототип"

Цель: Создать иерархию из нескольких классов, в которых реализованы методы клонирования объектов по шаблону проектирования "Прототип".

0. Придумать и создать 3-4 класса, которые как минимум дважды наследуются и написать краткое описание текстом.
1. Создать свой интерфейс IMyCloneable для реализации шаблона "Прототип".
2. Сделать возможность клонирования объекта для каждого из этих классов, используя вызовы родительских конструкторов.
3. Составить тесты или написать программу для демонстрации функции клонирования.
4. Добавить к каждому классу реализацию стандартного интерфейса ICloneable и реализовать его функционал через уже созданные методы.
5. Написать вывод: какие преимущества и недостатки у каждого из интерфейсов: IMyCloneable и ICloneable.

9	**Структурные шаблоны проектирования**	**Цели занятия:** объяснить назначение изученных шаблонов проектирования и применять их в своих проектах. шаблоны проектирования Proxy, Adapter, Decorator, Facade, Bridge, Composite, Flyweight.
10	**Поведенческие шаблоны проектирования**	**Цели занятия:** объяснить назначение изученных шаблонов проектирования и применять их в своих проектах. шаблоны проектирования Command, Iterator, Mediator, Memento, Observer, State, Template Method; рассмотрение на конкретном примере шаблонов Strategy и Visitor.
11	**Ретроспектива и планирование**	**Цели занятия:** уверенно участвовать в ретроспективах; планировать объём работ на будущее. проведение ретроспективы прошедшего спринта; проведение планирования будущего спринта.

3 Клиент-серверная архитектура и микросервисы

1 **WCF, ASMX, Web Api, REST**

Цели занятия:

отличать технологии удалённых вызовов друг от друга;
создавать готовый RESTful API сервер в интернете за пару кликов мышки;
реализовывать клиенты к RESTful API серверам.

понятие Удалённого вызова процедур и какие технологии существую для этого;
технологии удалённых вызовов в ASP.NET;
REST и RESTful API;
использование RESTful API на живом примере;
реализация своих клиентов к существующему API.

Домашние задания

1 Добавляем взаимодействие между клиентом и сервером

Цель: В этой домашней работе мы применим на практике полученные знания по работе с REST API. Для уже хорошо знакомого нам проекта Otus.Teaching.Concurrency.Import необходимо будет добавить два внутренних проекта - WebApi и консольный клиент. Аналогично работе на самом уроке, консольный клиент должен отправлять в WebApi запрос на сохранение случайным образом сгенерированного пользователя. Сервер, в свою очередь, при добавлении пользователя должен проверять наличие пользователя в БД и возвращать коды ответов:
- 200, если пользователь добавлен без ошибок;
- 409, если пользователь с таким Id уже существует в базе.

Также, должен быть метод получения пользователя по идентификатору, а сервер должен отдавать статус-код 200 с информацией о пользователе, если он есть, либо 404, если пользователь не был найден.

1. Доделать предыдущую работу по проекту https://gitlab.com/devgrav/Otus.Teaching.Concurrency.Import, если она ещё не закончена;
2. Создать проект WebApi с методом GET /users/{id};
3. Создать проект консольного приложения принимающего с консоли ID, запрашивающего и

отображающего данные по пользователю;
4. Добавить поддержку метода POST /users/;
5. Добавить генерацию случайного пользователя в консольном приложении и его отправку на сервер для сохранения.

---

2 **Паттерны корпоративных приложений**

Цели занятия:

сформулировать основные причины интеграции корпоративных систем и способы их интеграции;
сформулировать какой способ взаимодействия предпочтителен для различных типов задач и в чем причина популярности микросервисов.

причины и способы интеграции корпоративных приложений, зачем нужна микросервисная архитектура.

---

3 **Принципы SOLID**

Цели занятия:

изучить значение каждой буквы акронима;
узнать как применять эти принципы на практике.

принцип единственной ответственности;
принцип открытости/закрытости;
принцип подстановки Барбары Лисков;
принцип разделения интерфейса;
принцип инверсии зависимостей.

Домашние задания

1 Демонстрация SOLID принципов

Цель: Практическое применение SOLID принципов.

На примере реализации игры «Угадай число» продемонстрировать практическое применение SOLID принципов.
Программа рандомно генерирует число, пользователь должен угадать это число. При каждом вводе числа программа пишет больше или меньше отгадываемого. Кол-во попыток отгадывания и диапазон чисел должен задаваться из настроек.
В отчёте написать, что именно сделано по каждому принципу.
Приложить ссылку на проект и написать, сколько времени ушло на выполнение задачи.

4	**Брокеры сообщений**	**Цели занятия:** разобраться в сходствах и отличиях четырёх наиболее популярных брокеров сообщений; научиться писать код, получающий сообщения из очереди. что такое "Брокер сообщений"; обзор нескольких брокеров сообщений; создание кластера RabbitMQ в облаке; знакомство с интерфейсом панели администратора RabbitMQ: создание, просмотр, настройка и удаление очередей. самостоятельная реализация отправки и получения сообщений из очереди облачного RabbitMQ.
5	**CI/CD**	**Цели занятия:** разобраться в принципах CI/CD; понимать как организовать правильный CI/CD; сформулировать основные инструменты для CI и CD, их отличия; настроить собственный CI/CD. Принципы CI/CD; Правильная настройка и организация; Основные инструменты и их отличия; Настройка собственного CI/CD.
6	**Ретроспектива и планирование**	**Цели занятия:** уверенно участвовать в ретроспективах; планировать объём работ на будущее. проведение ретроспективы прошедшего спринта; проведение планирования будущего спринта.

4 Процессы и подходы

1	**Waterfall, Scrum, Kanban и прочие методологии**	**Цели занятия:** разобраться в современных методологиях, необходимости и особенностям их применения. что было до Agile? базовые Agile фреймворки; проблемы внедрения и как с ними бороться; фреймворки масштабирования.
2	**Unit, Sandbox, Blackbox, Whitebox, Integration tests**	**Цели занятия:** разбираться во всём этом многообразии типов тестов; понимать когда и какие нужно применять. unit тестирование; blackBox тестирование; whiteBox тестирование; модульное тестирование; интеграционное тестирование; регрессионное тестирование.
3	**Domain Driven Development: Основы**	**Цели занятия:** изучить теоретическую часть подхода Domain Driven Development. основные определения; концепция: ограниченные связи, целостность, взаимосвязь; Элементы DDD: - ограниченный контекст - сущность - объект-значение - агрегат - службы предметных областей Anti-corruption Layer.

4	**Domain Driven Development: Практикум**	**Цели занятия:** научиться применять подход самостоятельно на практике. практическое занятие по применению DDD.
5	**Логирование, метрики, трассировка и ведение документации**	**Цели занятия:** разбираться в инструментах исследования и анализа работы программ; выбирать и настраивать подходящую систему логирования; объяснять что из себя представляют метрики и как устроена трассировка; перечислить минимум три системы для ведения программной документации. логирование: - функции систем логирования - что необходимо логировать - распространённые ошибки логирования - выбор правильных уровней логирования - структурные логи - рекомендации по настройке метрики; трассировка; документация.

6 **Реактивное программирование**

Цели занятия:

изучить основные паттерны работы с последовательностями данных в режимах pull и push;
проанализировать интерфейс IObservable<T> и связанные с ним соглашения;
изучить основные возможности LINQ для IObservable<T>;
получить пример работы с подписками на данные в .net-приложении.

устройство IObservable<T> и IObserver<T>,
параллели с IEnumerable<T> и IEnumerator<T>;
synchronized observables;
unicast/multicast observables;
hot/cold observables;
основные методы LINQ to observables;
связь IObservable<T> с Task<T>.

5 Проектная работа

1 **Консультация по проектам и домашним заданиям**

Цели занятия:

получить ответы на вопросы по проекту, ДЗ и по курсу.

вопросы по улучшению и оптимизации работы над проектом;
затруднения при выполнении ДЗ;
вопросы по программе.

Домашние задания

1 Проектная работа

Цель: В этом домашнем задании вы продемонстрируете знания и навыки, которые приобрели на курсе.

Шаги и их состав зависят от конкретной темы проектной работы.

---

2 **Защита проектных работ**

Цели занятия:

защитить проект и получить рекомендации экспертов.

презентация проектов перед комиссией;
вопросы и комментарии по проектам.