Разработка приложений на платформе .NET

Лекция 7

Обработка исключений

Как обрабатывать ошибки?

- Возвращаемое значение
  - Не всегда возможно (конструкторы)
  - Не сразу проявляется
  - Непонятно, что конкретно произошло
  - Усложняет код
- Глобальная переменная
  - Не проходит в многопоточной среде
- Вызов определенной функции или завершение программы
  - Не проходит, если библиотеки независимы
- Обработка исключений

Генерация исключений

- Ошибке ставится в соответствие некоторый объект – исключение (**exception**)
- Объект-исключение должен иметь тип, который является потомком **System.Exception**
- При возникновении ошибочной ситуации **генерируется** (выбрасывается) исключение

```
if(j < 0 || j > 3)
  throw new IndexOutOfRangeException();
```

- Прерывается обычный ход выполнения операторов
- Если нет обработчика исключения, то приложение будет аварийно завешено.

Обработчик исключений

- Блок обработки исключений имеет следующий вид:

```
try
{
   // Код, требующий корректного восстановления или очистки ресурсов
}
[catch [(exception_type1 [name_1])]
{
   // Код восстановления, после возникновения исключения exception_type1
}
… catch(exception_typeN [name_N])
{
   // Код восстановления, после возникновения исключения exception_typeN
} ]
[finally
{
   // Код очисти ресурсов, после операций в блоке try
   // Этот код выполняется всегда, вне зависимости от наличия исключения
} ]
```

- Блок **try** – обязательный
- Блоки **catch** и **finally** могут отсутствовать, однако хотя бы один из них должен присутствовать

Выбор обработчиков

- После генерации исключения рассматриваются блоки в порядке возрастания удаленности
  - Сначала смотрится статическая вложенность – вложенность блоков кода (внутри метода)
  - Затем смотрится динамическая вложенность – порядок вызовов в стеке
- В них просматриваются все обработчики (**catch**) сверху вниз.
- Если тип сгенерированного исключения – потомок типа, указанного в блоке **catch**, блок **catch** выбирается для обработки этого исключения
- Если найден блок **catch**, соответствующий исключению, то до выполнения кода в блоке **catch** выполняются все внутренние блоки **finally** (если имеются).
- После обработки исключения выполнение в каком-то блоке **catch**, все остальные блоки **catch** игнорируются.
- Блок **finally** выполняется всегда при выходе из блока обработки исключения, независимо от того возникло ли исключение или нет, обработано ли оно или нет. Блок **finally** выполнится всегда.
- Если исключение возникло и обработано, то выполнение продолжается с кода, следующего за обработавшим исключение **try-catch-finally** блоком (сначала выполнив блок **finally**)
- Если при возникновении исключения не находится подходящий блок **catch**, т.е. исключение остается не обработанным, то выполнение программы аварийно завершается.

Фильтр исключений (C# 6)

- В **C# 6** добавлена возможность фильтрации исключений

```
try
{
   // Код, требующий корректного восстановления или очистки ресурсов
}
[catch (Exception ex) when (condition)
{
   // Код восстановления, после возникновения исключения Exception при
   условии выполнения выражения condition
}
… catch (Exception ex)
   // Код восстановления, после возникновения исключения Exception при
   условии, что не подошли предыдущие блоки catch (с их условиями)
} ]
[finally
{
   // Код очисти ресурсов, после операций в блоке try
   // Этот код выполняется всегда, вне зависимости от наличия исключения
} ]
```

Варианты действий

- Способы восстановления после исключения

```
try
{
     // Читаем файл
}
catch (FileNotFoundException e)
{
    throw e;
}
catch (FileLoadException)
{
   throw;
}
catch (IOException e)
{
   throw new MyException("Что-то не так при
                работе с файлом", e);
}
catch (Exception)
{
  // Все ок. Не генерируем новое  исключения
}
```

Еще раз сгенерировать то же исключение для передачи информации о нем коду, расположенному выше в стеке;

Сгенерировать исключение другого типа для передачи дополнительной информации коду, расположенному выше в стеке;

Позволить программному потоку выйти из блока catch естественным образом

System.Exeption

- Базовый класс для всех исключений

- Поле **Message** – описание возникшего исключения
- Поле **Source** – ссылка на объект, сгенерировавший исключение, или на сборку, в которой возникло исключение
- Поле **StackTrace** – стек вызовов
- Поле **InnerExeption** – “предыдущее” исключение (при дальнейшей передачи исключения)

- При создании своего типа исключения необходимо наследоваться от типа **System.Exeption**

Демонстрации

Исключения