

Увод в концепцията за дебъгване: откриване и отстраняване на проблеми

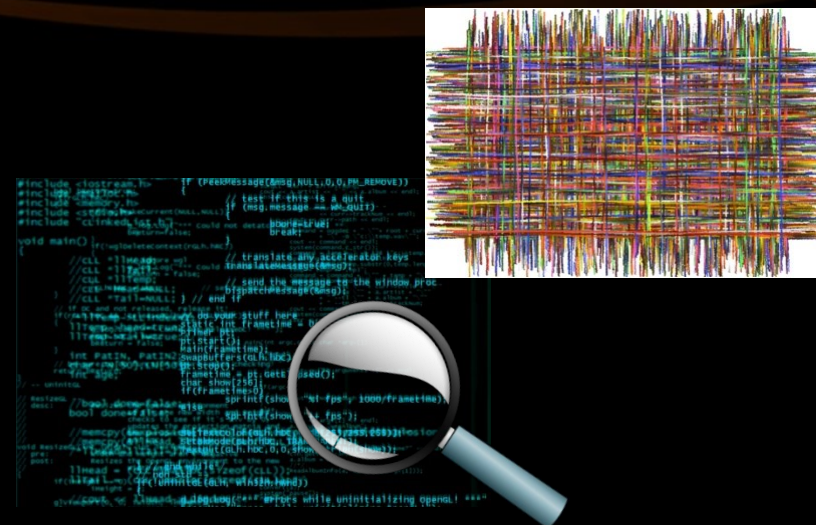
Инспектиране на данни. Нишки. Стек



Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>



Съдържание

- Инспектиране на данни
- Нишки и стек

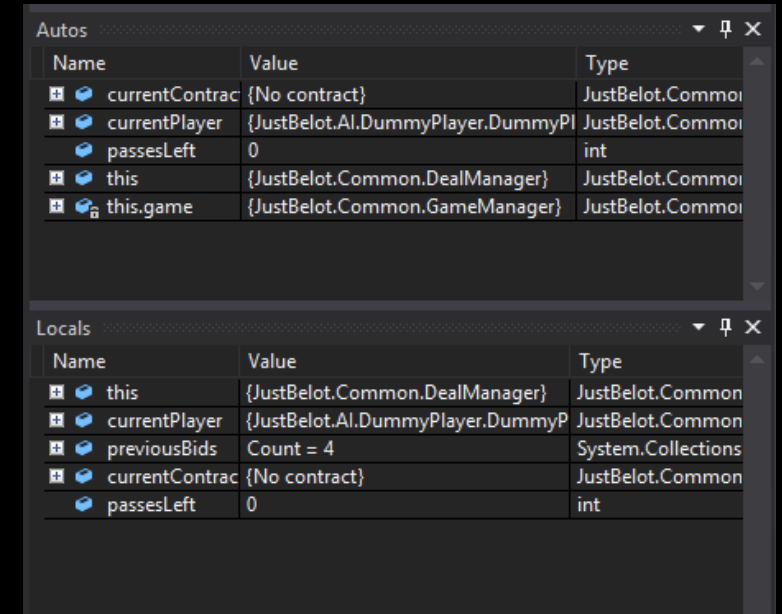


Проверка на данните – Data Inspection

- Дебъгването е изцяло за проверка на информацията
 - Какво са локалните променливи?
 - Какво значи В паметта (in Memory)?
 - Какво е поток на кода (Code Flow)?
 - По принцип - Какво е състоянието на процеса точно в този момент и какво се случва?
- В същност, лекотата на изследване на данните е ключът към бързо разрешаване на проблеми

Инспектиране на данни с Visual Studio Data Inspection

- Visual Studio предлага много функции за инспекция на данни в прозореца за наблюдение **Watch**
 - Авто- и Локални променливи (Autos и Locals)
 - Памет и Регистри (Memory и Registers)
 - Съвети за данните (Data Tips)
 - Прозорецът Незабавно (Immediate)



The screenshot shows the Visual Studio Data Inspection window with two tabs: 'Autos' and 'Locals'. The 'Autos' tab is active and displays a table of variables. The 'Locals' tab is also visible below it, showing a similar table of variables.

Name	Value	Type
currentContract	{No contract}	JustBelot.Common
currentPlayer	{JustBelot.AI.DummyPlayer.DummyPI	JustBelot.Common
passesLeft	0	int
this	{JustBelot.Common.DealManager}	JustBelot.Common
this.game	{JustBelot.Common.GameManager}	JustBelot.Common

Name	Value	Type
this	{JustBelot.Common.DealManager}	JustBelot.Common
currentPlayer	{JustBelot.AI.DummyPlayer.DummyP	JustBelot.Common
previousBids	Count = 4	System.Collections
currentContract	{No contract}	JustBelot.Common
passesLeft	0	int

Прозорецът за Наблюдение – Watch Window

- Позволява ви да се запознаете с различни състояния на вашето приложение
- Няколко различни "типови" прозорци за наблюдение
 - Autos
 - Locals
- Прозорец за наблюдение "По избор" също е възможен
 - Съдържа само променливи, които сте избрали да добавите
 - Кликнете с десния бутон върху променливата и изберете "Add to Watch"

Прозорците Авто и Локални – Autos and Locals

- **Локални (Locals)** е прозорец за наблюдение, съдържащ локални променливи за специфична част (фрейм) от стека
 - Debug -> Windows -> Locals
 - Показва: името на променливата, стойността и типа ѝ
 - Позволява детайлизиране на обекта чрез кликване върху знака „+“ в дървото за управление
- **Авто (Autos)** позволява на дебъгера да реши кои променливи да се показват в прозореца
 - Свободно, на база на текущия и предишен блок

Памет и Регистри

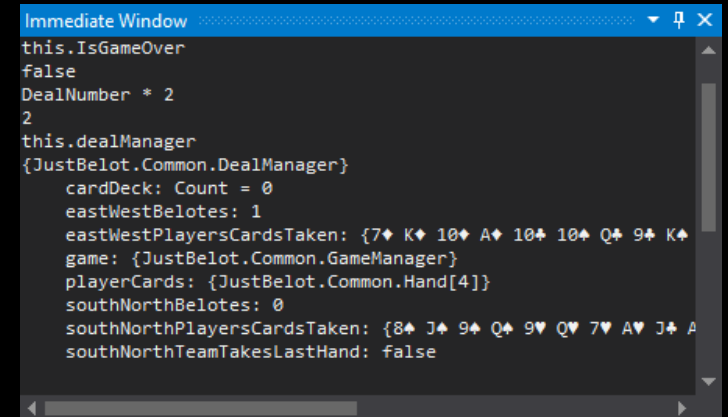
- Прозорецът Памет (Memory) може да се използва за изследване на паметта в обхвата на процеса
 - Адресното поле може да бъде указател или израз
 - Може да влачите и пускате променлива от прозореца с кода
 - Броят на показваните колони може да се конфигурира
 - Форматът на данните може да бъде конфигуриран
- Прозорецът Регистри може да се използва да инспектирате регистрите на процесора

Съвети за данни – Data Tips

- Предоставя информация за променливи
 - Променливите трябва да са в рамките на обхвата на текущото изпълнение
- Поставете показалеца на мишката върху коя да е променлива
- Променливите могат да бъдат отваряни като дърво със знака +
- Може да „забождате“ данните, така че винаги да стоят отворени
- Може да добавяте пояснителни коментари
- Съветите за данните поддържат влачене и пускане
- Импорт и експорт на съвети за данните

Прозорец за моментално изпълнение – Immediate Window

- Използва се когато се дебъгва голям израз, който трябва да бъде изпълнен
 - За извеждане на стойност на променлива <name of variable>
 - За задаване стойност на променлива, пише се: <name of variable>=<value>
 - За извикване на метод, използва се <name of variable>.<method>(arguments)
- Прилича на регулярен израз
- Поддържа IntelliSense



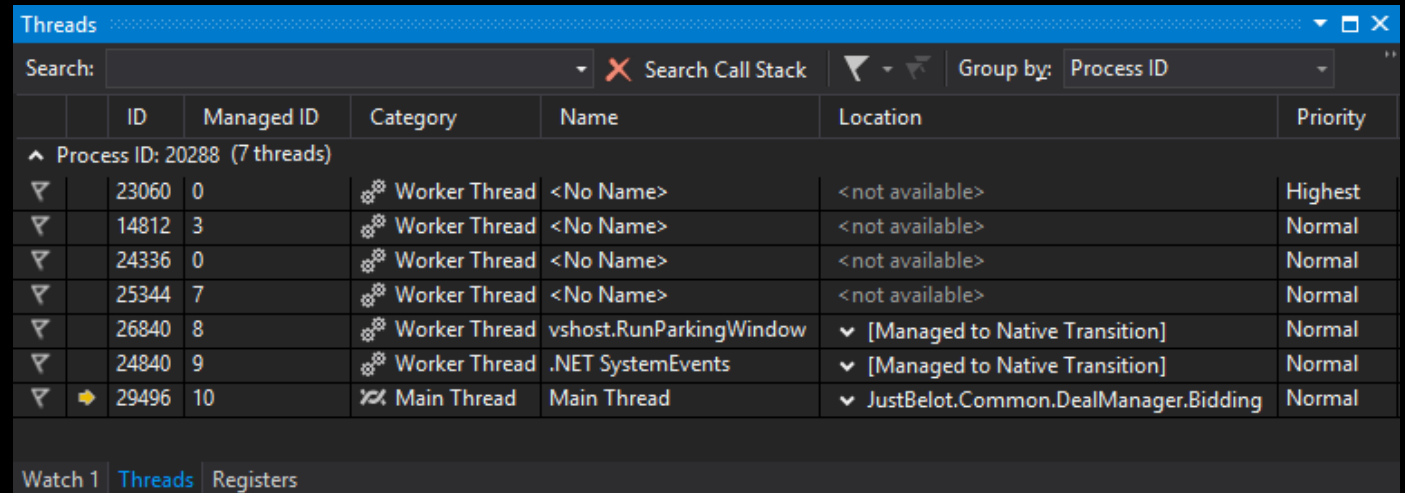
```
Immediate Window
this.IsGameOver
false
DealNumber * 2
2
this.dealManager
{JustBelot.Common.DealManager}
  cardDeck: Count = 0
  eastWestBelotes: 1
  eastWestPlayersCardsTaken: {7♦ K♦ 10♦ A♦ 10♠ 10♣ 9♠ K♠}
  game: {JustBelot.Common.GameManager}
  playerCards: {JustBelot.Common.Hand[4]}
  southNorthBelotes: 0
  southNorthPlayersCardsTaken: {8♠ J♠ 9♠ Q♠ 9♥ Q♥ 7♥ A♥ J♠ A}
  southNorthTeamTakesLastHand: false
```

Нишки – Threads

- Основни единици от изпълняващ се код
- Общо, програмите използват повече от една нишка
 - В .NET винаги има повече от една нишка
- Всяка нишка има **област от паметта** свързана с нея, известна като **стек**
 - Пазеща локални променливи
 - Пазеща област със специфична информация
- Област от паметта, работеща на принципа LIFO – последно влезнал – първи излезнал

Прозорецът с нишките на процеса, които се изпълняват – Threads Window

- Преглед на дейността на нишките в процеса
- Включва основна информация за всяка нишка
 - Идентификатор (номер) на нишка
 - Категория
 - Име
 - Разположение (място)
 - Приоритет



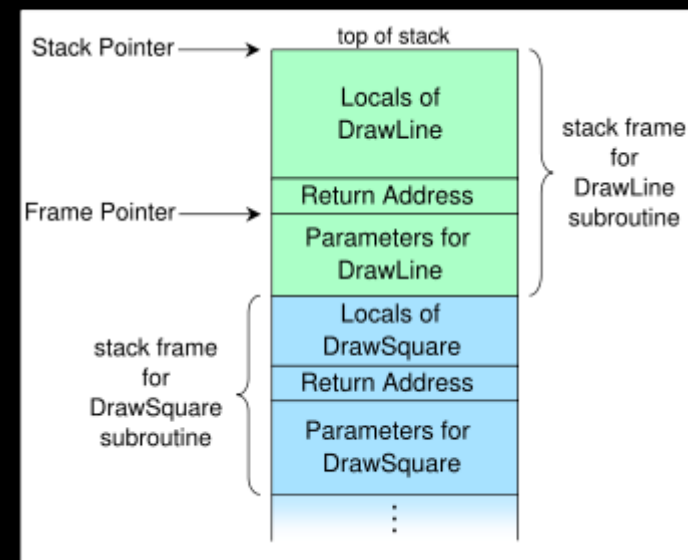
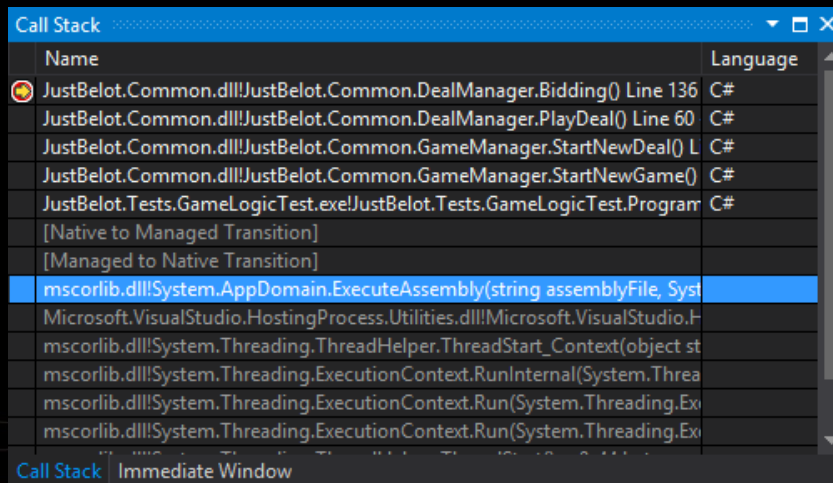
The screenshot shows the 'Threads' window in Windows Task Manager. It displays a list of threads for a specific process (ID: 20288). The window has a search bar, a 'Search Call Stack' button, and a 'Group by' dropdown menu set to 'Process ID'. The table below lists the threads with their IDs, managed IDs, categories, names, locations, and priorities.

	ID	Managed ID	Category	Name	Location	Priority
^ Process ID: 20288 (7 threads)						
▼	23060	0	Worker Thread	<No Name>	<not available>	Highest
▼	14812	3	Worker Thread	<No Name>	<not available>	Normal
▼	24336	0	Worker Thread	<No Name>	<not available>	Normal
▼	25344	7	Worker Thread	<No Name>	<not available>	Normal
▼	26840	8	Worker Thread	vshost.RunParkingWindow	▼ [Managed to Native Transition]	Normal
▼	24840	9	Worker Thread	.NET SystemEvents	▼ [Managed to Native Transition]	Normal
▼	29496	10	Main Thread	Main Thread	▼ JustBelot.Common.DealManager.Bidding	Normal

At the bottom of the window, there are tabs for 'Watch 1', 'Threads' (which is selected), and 'Registers'.

Стек с извикванията – Call Stacks

- Стекът с нишките общо се нарича Стек с извикванията (call stack)
- Visual Studio показва елементите на стека с извикванията
 - Локални променливи
 - Методи (области от кода на метода)



Какво научихме в този час?

- Как да инспектираме по-задълбочено дебъгваните данни и методи
 - Locals, Autos, Watch
- Да проследяваме състоянието на нишките в .Net
- Да работим с данните в стека



Използване на дебъгер



Въпроси?



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство
на образованието
и науката



Национална
програма
„Обучение за
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**



SoftUni
Foundation

