Lab14

Что такое процесс, домен, поток? Как они связаны между собой?

- **Процесс** — это выполняемая программа, которая содержит свой собственный адресный пространство, данные, стек и системные ресурсы, необходимые для выполнения.
- **Домен приложения** — это логическая единица, в которой выполняются .NET приложения. Каждый домен приложения изолирован, имеет собственную память и безопасность, что помогает предотвратить ошибки и сбои между приложениями.
- **Поток** — это наименьшая единица выполнения, которая выполняет инструкции программы. Потоки могут быть созданы в процессе, и несколько потоков могут работать параллельно или поочередно в рамках одного процесса.

Связь между ними:

- Один процесс может содержать несколько доменов приложений.
- Каждый домен приложения может иметь несколько потоков.
- Потоки одного процесса разделяют ресурсы этого процесса, но каждый поток имеет своё состояние выполнения.

2. Как получить информацию о процессах?

Информацию о процессах можно получить с помощью класса, который предоставляет методы и свойства для получения списка запущенных процессов на компьютере, а также для получения деталей о каждом процессе, таких как идентификатор, имя, использование процессора, память и другие характеристики.

3. Как создать и настроить домен?

Домен приложения в .NET создается с помощью специального класса, который позволяет создать новый домен, управлять его безопасностью, загрузкой сборок и сбором мусора. Для настройки домена можно указать различные параметры, такие как политики безопасности или сборки, которые нужно загрузить в домен.

4. Как создать и настроить поток?

Потоки в .NET можно создать с использованием класса, который предоставляет конструкторы для создания потоков, а также свойства для их настройки. Потоки можно настроить с использованием таких параметров, как приоритет, фоновый режим и метод, который будет выполняться в потоке.

5. В каких состояниях может быть поток?

Поток в .NET может находиться в следующих состояниях:

- **Не начат (Unstarted)** — поток был создан, но не был запущен.
- **Выполняется (Running)** — поток выполняет свои инструкции.
- **Ожидание (WaitSleepJoin)** — поток приостановлен, ожидая завершения другой операции или задачи.
- **Завершён (Stopped)** — поток завершил свою работу.
- **Запрашивающий завершение (AbortRequested)** — поток помечен для завершения, но не завершён.
- **Приостановлен (Suspended)** — поток был приостановлен вручную.

6. Какие методы управления потоками вы знаете, для чего и как их использовать?

Методы управления потоками включают:

- **Start()** — запускает выполнение потока.
- **Sleep()** — приостанавливает поток на заданное время.
- **Join()** — блокирует текущий поток до завершения работы другого потока.
- **Abort()** — пытается принудительно завершить поток (устаревший метод).
- **Suspend()** — приостанавливает поток (устаревший метод).

- **Resume()** — возобновляет приостановленный поток (устаревший метод).

Для синхронизации доступа к ресурсам можно использовать методы **Monitor.Enter()** и **Monitor.Exit()**, которые позволяют блокировать участок кода для одного потока.

7. Какие приоритеты потока вы знаете?

В .NET потоки могут иметь следующие приоритеты:

- **Lowest** — самый низкий приоритет.
- **BelowNormal** — ниже среднего приоритета.
- **Normal** — стандартный приоритет.
- **AboveNormal** — выше среднего приоритета.
- **Highest** — самый высокий приоритет.

Эти приоритеты позволяют управлять порядком выполнения потоков, когда в системе работает несколько потоков.

8. Что такое пул потоков и для чего он используется?

Пул потоков — это набор заранее созданных потоков, которые могут быть использованы для выполнения задач. Это помогает снизить накладные расходы на создание и уничтожение потоков, улучшая производительность при многократном запуске короткоживущих задач. Пул потоков автоматически управляет количеством потоков, создаваемых и используемых для выполнения операций.

9. Что такое критическая секция? Поясните использование.

Критическая секция — это участок кода, который должен быть выполнен только одним потоком в любой момент времени. Это необходимо для предотвращения конфликтов доступа к общим данным. Для синхронизации потоков, работающих с критической секцией, используются механизмы блокировки, такие как **lock** или **Monitor**, чтобы только один поток мог войти в критическую секцию одновременно.

10. Что такое мьютекс? Поясните использование.

Мьютекс — это механизм синхронизации, который предотвращает одновременный доступ к ресурсу несколькими потоками. Мьютексы могут быть как локальными (работают только внутри одного процесса), так и глобальными (могут использоваться между разными процессами). Мьютекс блокирует ресурс, чтобы только один поток мог работать с ним в любой момент времени, а другие потоки должны ожидать освобождения ресурса.

11. Что такое семафор? Поясните использование.

Семафор — это объект синхронизации, который управляет количеством потоков, которые могут одновременно получить доступ к ресурсу. Семафор работает через счётчик, который ограничивает количество потоков, имеющих доступ к ресурсу. Если все ресурсы заняты, потоки должны ожидать, пока не освободится хотя бы один ресурс. Семафоры полезны для контроля доступа к ограниченным ресурсам, например, к пулам соединений с базой данных.

12. Что такое неблокирующие средства синхронизации?

Неблокирующие средства синхронизации позволяют потокам синхронизировать свою работу без того, чтобы они блокировали выполнение при ожидании ресурса. Вместо блокировки потоки могут продолжать выполнение других операций, если доступ к ресурсу в данный момент невозможен. Примеры таких средств включают **Interlocked** (для атомарных операций) и **SpinLock** (активное ожидание).

13. Для чего можно использовать класс Timer?

Класс Timer используется для выполнения отложенных или повторяющихся задач через определённые промежутки времени. Это полезно для создания задач, которые должны выполняться периодически или через заданные интервалы времени, например, для таймеров в приложениях, проверяющих состояние сервера или

выполняющих фоновые операции.