**Добрый день, 16а группа!**

Продолжаем общаться дистанционно.

Сегодня мы разберем создание компьютернойые модели различных процессов

Задать вопросы, а также прислать ответы вы можете

1. на адрес электронной почты: [ddrmx@ya.ru](mailto:ddrmx@ya.ru)
2. через соцсеть <https://vk.com/ddrmx>

С уважением, Максим Андреевич.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ ПО ТЕМЕ:

Компьютерные модели. (1 ЧАС)

Код реализации паттерна MVC (Model View Controller):

Представление:

1. **namespace** MVC
2. {
3. **public** partial **class** View : Form
4. {
5. // создаем объект класса Controller
6. Controller controller = **new** Controller();
7. **public** View()
8. {
9. InitializeComponent();
10. }
12. **private** **void** button1\_Click(**object** sender, EventArgs e)
13. {
14. **if** (textBox1.Text != "")
15. // выводим результат
16. MessageBox.Show(controller.Question(textBox1.Text));
17. }
18. }
19. }

Контроллер:

1. **namespace** MVC
2. {
3. **class** Controller
4. {
5. **public** **string** Question(**string** msg)//msg - то что ищем
6. {
7. Model model =  **new** Model();
8. **return** "На даный момент у нас товар "
9. + model.GetAnser(msg);
10. }
11. }
12. }

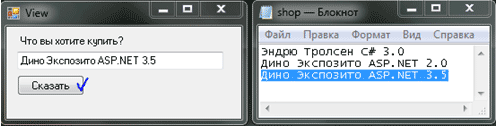
Модель:

1. **namespace** MVC
2. {
3. **class** Model
4. {
5. //question - то что ищем
6. **public** **string** GetAnser(**string** question)
7. {
8. **return** DataBase.GetAnser(question);
9. }
10. }
11. }

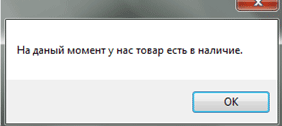
Класс помощник. Умеет общаться с базой данных:

1. **namespace** MVC
2. {
3. **static** **class** DataBase
4. {
5. //question - то что ищем
6. **public** **static** **string** GetAnser(**string** question)
7. {
8. **string** path = "D:\\shop.txt";// путь к базе данных ):
10. // создаем объект (читатель)
11. **using** (StreamReader sr =
12. **new** StreamReader(path, Encoding.Default))
13. {
14. **string** str = **null**;
15. **while** (**true**)
16. {
17. //если пустая, то это конец
18. str = sr.ReadLine();
19. **if** (str != **null**)
20. {
21. //если считаная строчка
22. //равна строке для поиска
23. **if** (str == question)
24. **return** "есть в наличие.";
25. }
26. **else**
27. **return** "нет в наличие.";
28. }
29. }
30. }
31. }
32. }

Тестируем:



Видим, что в текстовом файле есть такая запись, по этому ответ следующий:



***Контрольные вопросы***

* 1. Что такое MVC?
  2. Опишите MVC в других языках программирования.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ ПО ТЕМЕ:

Конструирование программ. (2 ЧАСА)

# Создание простого приложения с помощью Visual C#

Выполняя данное пошаговое руководство, вы ознакомитесь со многими инструментами, диалоговыми окнами и конструкторами, которые можно использовать для создания приложений с помощью Visual Studio. Вам предстоит создать простое приложение типа “Hello, World”, разработать пользовательский интерфейс, добавить код и отладить ошибки, одновременно приобретая навыки работы в интегрированной среде разработки (IDE).

## Настройка интегрированной среды разработки (IDE)

После запуска Visual Studio далее необходимо выбрать сочетание предопределенных параметров интегрированной среды разработки. Каждое сочетание параметров было разработано так, чтобы сделать разработку приложений более простой.

Это пошаговое руководство предполагает, что действуют **Обычные параметры разработки**, что соответствует минимальному объему настройки интегрированной среды разработки.

После открытия Visual Studio вы увидите окна инструментов, меню и панели инструментов, а также основную область окна. Окна инструментов закреплены в левой и правой частях окна приложения, а панель **Быстрый запуск**, строка меню и стандартная панель инструментов закреплены в верхней его части. В центре окна приложения находится **Начальная страница**. При загрузке решения или проекта редакторы и конструкторы отображаются в области **Начальной страницы**. При разработке приложения чаще всего используется именно эта область.

Рисунок. Интегрированная среда разработки Visual Studio



С помощью диалогового окна **Параметры** можно дополнительно настроить Visual Studio, например изменить в редакторе начертание и размер шрифта текста или изменить цветовую тему интегрированной среды разработки. В зависимости от примененного сочетания параметров некоторые элементы в этом диалоговом окне могут не отображаться автоматически. Чтобы отображались все возможные параметры, установите флажок **Показать все параметры**.

Рисунок. Диалоговое окно "Параметры"



#### Изменение цветовой темы интегрированной среды разработки

1. Откройте диалоговое окно **Параметры**, выбрав в меню **Сервис** вверху пункт **Параметры...** .



1. Измените **Цветовую тему** на **Темную**, а затем щелкните **ОК**.



Цвета в Visual Studio должны соответствовать следующему рисунку:



На изображениях в остальной части этого пошагового руководства используется светлая тема.

## Создание простого приложения

### Создание проекта

При создании приложения в Visual Studio необходимо сначала создать проект и решение. Этот пример демонстрирует создание решения Windows Presentation Foundation.

##### Создание проекта WPF

1. Создайте новый проект. В меню последовательно выберите пункты **Файл**, **Создать**, **Проект...** .



Кроме того, можно ввести **Создать проект** в окне **Быстрый запуск**, чтобы выполнить то же самое действие.



1. Выберите шаблон приложения WPF Visual Basic или Visual C#, воспользовавшись, например, путем по меню в левой области **Установлены**, **Шаблоны**, **Visual C#**, **Windows**, а затем выбрав «Приложение WPF» в средней области. В нижней части диалогового окна нового проекта назовите проект HelloWPFApp.





Visual Studio создает решение и проект HelloWPFApp, и **Обозреватель решений** показывает различные файлы. Конструктор WPF отображает представление конструирования и представление XAML файла MainWindow.xaml в одном разделенном представлении. Сдвигая разделитель, можно делать любое из представлений больше или меньше. Можно выбрать для просмотра только визуальное представление или только представление XAML. В **Обозревателе решений** отображаются следующие элементы.

Рисунок. Элементы проекта



После создания проекта его можно настраивать. С помощью окна **Свойства** (в меню **Вид**) можно отображать и изменять параметры элементов проекта, элементов управления и других элементов в приложении. С помощью свойств проекта и страниц свойств можно отображать и изменять параметры проектов и решений.

##### Изменение имени MainWindow.xaml

1. В следующей процедуре вы дадите MainWindow более конкретное имя. В **Обозревателе решений**выберите файл MainWindow.xaml. Должно отображаться окно **Свойства**, но если его нет, выберите в меню **Вид** пункт **Окно свойств** . Измените значение свойства **Имя файла** на Greetings.xaml.



**Обозреватель решений** показывает, что файл теперь называется Greetings.xaml, и если развернуть узел MainWindow.xaml (переместив курсор на узел и нажав клавишу стрелки вправо), то видно, что файл MainWindow.xaml.vb (или MainWindow.xaml.cs) теперь называется Greetings.xaml.vb (или Greetings.xaml.cs). Этот файл с текстом программы вложен в узел файла .xaml, что означает их тесную связь.

**Предупреждение**

Это изменение вызовет ошибку; сведения по ее отладке и исправлению будут предоставлены позднее.

1. В **Обозревателе решений** откройте файл Greetings.xaml в представлении конструктора (путем нажатия клавиши ВВОД при выбранном узле) и выберите заголовок окна с помощью мыши.
2. В окне **Свойства** измените значение свойства **Заголовок** на Greetings.

Заголовок окна для MainWindow.xaml теперь содержит текст Greetings.

### Конструирование пользовательского интерфейса (ИП)

В приложение будет добавлено три типа элементов управления: элемент управления TextBlock, два элемента управления RadioButton, и элемент управления Button.

##### Добавление элемента управления TextBlock

1. Откройте окно **Панель элементов**, выбрав в меню **Вид** пункт **Панель элементов**.
2. На **Панели элементов** найдите элемент управления TextBlock.



1. Добавьте элемент управления TextBlock на поверхность разработки, выбрав элемент управления TextBlock и перетащив его на поверхность. Отцентрируйте этот элемент в верхней части окна.

Окно должно выглядеть так, как показано на следующем рисунке:

Рисунок. Окно Greetings с элементом управления TextBlock



1. В представлении XAML найдите разметку для TextBlock и измените атрибут Text: Text=”Select a message option and then choose the Display button.”
2. Если в элементе управления TextBlock не отображается весь текст в представлении конструирования, увеличьте элемент TextBlock (используя маркеры на краях).
3. Сохраните изменения, нажав клавиши Ctrl-s или с помощью меню **Файл**.

Следующий шаг — добавить в форму два элемента управления [RadioButton](https://msdn.microsoft.com/library/6c9ba847-eab7-4bba-9c74-6b56ef72067b).

##### Добавление переключателей

1. На **Панели элементов**найдите элемент управления RadioButton.



1. Добавьте два элемента управления RadioButton на поверхность разработки, выбрав элемент RadioButton и дважды перетащив его на поверхность, и переместите кнопки (выбрав их и используя клавиши со стрелками), чтобы кнопки отображались рядом и под элементом управления TextBlock.

Окно должно выглядеть следующим образом:

Рисунок. элементы управления RadioButton в окне Greetings.



1. В окне **Свойства** для левого элемента управления RadioButton измените свойство **Имя** (свойство в верхней части окна **Свойства**), задав ему значение RadioButton1. Убедитесь, что выбран элемент управления RadioButton и в форме нет фоновой сетки; поле «Тип» в окне свойств под полем «Имя» должно иметь значение «RadioButton».
2. В окне **Свойства** для правого элемента управления RadioButton измените свойство **Имя** на RadioButton2и сохраните изменения, нажав клавиши Ctrl-s или с помощью меню **Файл**. Перед изменением и сохранением убедитесь, что выбран элемент управления RadioButton.

Теперь можно добавить отображаемый текст для каждого элемента управления RadioButton. Следующая процедура обновляет свойство **Content** элемента управления RadioButton.

##### Добавление отображаемого текста для каждого переключателя

1. Откройте контекстное меню элемента управления RadioButton1 (выбрав его и нажав правую кнопку мыши), выберите команду **Изменить текст**, а затем введите Hello.
2. Откройте контекстное меню элемента управления RadioButton2 (выбрав его и нажав правую кнопку мыши), выберите команду **Изменить текст**, а затем введите Goodbye.

Последний элемент пользовательского интерфейса, который вам предстоит добавить, — это [Button](https://msdn.microsoft.com/library/a9d8f5a5-c98c-463e-808a-5a4e63173098).

##### Добавление элемента управления Button

1. В **Панели инструментов**найдите элемент управления **Кнопка** и добавьте его на поверхность разработки под элементами управления RadioButton, выбрав кнопку и перетащив ее на форму в представлении конструирования.
2. В представлении XAML измените значение свойства **Содержимое** элемента управления «Кнопка» с Content=”Button” на Content=”Display”и сохраните изменения (используя Ctrl-s или меню **Файл** ).

Окно должно выглядеть так, как показано на следующем рисунке.

Рисунок. Окончательный пользовательский интерфейс Greetings



### Добавление кода к кнопке Display

После запуска приложения окно сообщения появится только тогда, когда пользователь сначала выберет переключатель, а затем нажмет кнопку **Display**. Одно окно сообщения появится для Hello, и другое — для Goodbye. Для создания такого поведения добавьте код в событие Button\_Click в файле Greetings.xaml.vb или Greetings.xaml.cs.

##### Добавление кода для отображения окон сообщений

1. На поверхности разработки дважды щелкните кнопку **Display**.

Будет открыт файл Greetings.xaml.vb или Greetings.xaml.cs, а курсор будет установлен в событии Button\_Click. Также можно добавить обработчик событий щелчка следующим образом (если во вставленном коде какие-либо имена подчеркнуты красной волнистой линией, то, возможно, не были выбраны и переименованы элементы управления RadioButton на поверхности разработки):

Visual C# обработчик события должен выглядеть следующим образом:

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

}

if (RadioButton1.IsChecked == true)

{

MessageBox.Show("Hello.");

}

else

{

RadioButton2.IsChecked = true;

MessageBox.Show("Goodbye.");

}

1. Сохраните приложение.

## Отладка и тестирование приложения

После этого вам предстоит отладить приложение для выявления ошибок и тестирования того, что оба окна сообщений отображаются правильно. Приведенные ниже инструкции описывают, как выполнить сборку и запустить отладчик.88888888

### Поиск и исправление ошибок

На этом шаге вам предстоит найти ошибку, которую мы намеренно допустили ранее, изменив имя файла XAML главного окна.

##### Начало отладки и поиск ошибки

1. Запустите отладчик, выбрав **Отладка**, затем **Начать отладку**.



Появится диалоговое окно, показывающее, что произошло исключение IOException "Не удалось обнаружить ресурс ‘mainwindow.xaml’".

1. Нажмите кнопку **ОК**, а затем остановите отладчик.



Файл Mainwindow.xaml был переименован в Greetings.xaml в начале этого пошагового руководства, но код по-прежнему ссылается на файл Mainwindow.xaml как на начальный универсальный код ресурса (URI) для приложения, поэтому проект не может быть запущен.

##### Задание Greetings.xaml в качестве начального универсального кода ресурса (URI)

1. В **Обозревателе решений**откройте файл App.xaml (в проекте C#) или файл Application.xaml (в проекте Visual Basic) в представлении XAML (его невозможно открыть в представлении конструирования), выбрав файл и нажав клавишу ВВОД или дважды щелкнув его.
2. Измените StartupUri="MainWindow.xaml" на StartupUri="Greetings.xaml"и сохраните изменения, нажав Ctrl-s.

Запустите отладчик снова (клавишей F5). Должно появиться окно Greetings приложения.

### Отладка с точками останова

Добавление некоторых точек останова позволяет тестировать код во время отладки. Для добавления точки останова можно выбрать в меню **Отладка** пункт **Точка останова** или щелкнуть в левой области редактора рядом со строкой кода, на которой требуется приостановить выполнение.

##### Добавление точек останова

1. Откройте файл Greetings.xaml.vb или Greetings.xaml.cs и выделите следующею строку: MessageBox.Show("Hello.")
2. Добавьте точку останова, выбрав меню **Отладка**, затем — **Точка останова**.



Рядом со строкой кода в крайнем левом поле окна редактора появится красный кружок.

1. Выделите следующую строку: MessageBox.Show("Goodbye.").
2. Нажмите клавишу F9, чтобы добавить точку останова, затем нажмите клавишу F5, чтобы начать отладку.
3. В окне **Greetings** выберите переключатель **Hello** и нажмите кнопку **Display**.

Строка MessageBox.Show("Hello.") выделяется желтым цветом. В нижней части интегрированной среды разработки окна «Видимые», «Локальные» и «Контрольные значения» закреплены вместе на левой стороне, а окна «Стек вызовов», «Точки останова», «Команда», «Интерпретация» и окно вывода закреплены вместе на правой стороне.

1. В меню **Отладка** выберите **Шаг с выходом**.

Приложение возобновит выполнение, и появится окно сообщения со словом "Hello".

1. Нажмите кнопку **ОК** в окне сообщения, чтобы закрыть его.
2. В окне **Greetings** выберите переключатель **Goodbye** и нажмите кнопку **Display**.

Строка MessageBox.Show("Goodbye.") выделяется желтым цветом.

1. Нажмите клавишу F5, чтобы продолжить отладку. Когда появится окно сообщения, нажмите в нем кнопку **ОК**, чтобы закрыть его.
2. Нажмите комбинацию клавиш SHIFT + F5 (нажмите сначала клавишу SHIFT и, удерживая ее нажатой, нажмите клавишу F5), чтобы остановить отладку.
3. В меню **Отладка** выберите **Выключить все точки останова**.

### Сборка окончательной версии приложения

1. Выберите в главном меню **Сборка**и **Очистить решение** для удаления промежуточных файлов и выходных файлов, которые были созданы в ходе предыдущих сборок. Это не является обязательным, но очищает результаты отладочной сборки.



1. Измените конфигурацию сборки для HelloWPFApp с **Отладка** на **Выпуск** с помощью раскрывающегося списка на панели инструментов (сейчас это «Отладка»).



1. Постройте решение, выбрав **Сборка**, затем **Собрать решение** или нажмите клавишу F6.



Завершение работы, построенный EXE-файл находится в каталоге решения и проекта (... \HelloWPFApp\HelloWPFApp\bin\Release\)

ЗАНЯТИЕ ПО ТЕМЕ:

ИТ процессы. (1 ЧАС)





Посмотрите следующий видеоролик <https://youtu.be/xDJqRS5d7MQ>

Работа ИТ-специалистов и служб, как правило, ориентирована на решение локальных задач в рамках глобальных процессов. То есть, фактически, любое действие так или иначе оказывает влияние на работу системы в целом и отдельных её элементов в частности.

Все ИТ-процессы можно поделить на две категории: связанные с обеспечением качества предоставляемых сервисных услуг и не связанные с ним. При этом в комплексе любые такие процессы обеспечивают максимальную эффективность и работоспособность системы с точки зрения её практического применения.

Эффективная реализация ИТ-процессов достигается вполне объективными критериями оценки, основанные на:

* ценности предоставляемых технологических и ИТ-решений для клиента (измеряемой в конкретных показателях - оптимизации бизнеса, повышении скорости обслуживания клиентов, упрощении в ведении отчётности и пр.);
* объективном обосновании необходимых затрат, обеспечении четкого финансового контроля;
* снижении уровня рисков в сфере защиты данных до приемлемых значений.

Контроль над реализацией ИТ-процессов

Контроль за ИТ-процессами в основном сводится к мониторингу и сбору статистических данных, позволяющих своевременно выявить отклонение от нормы и при необходимости скорректировать ход процессов в нужную сторону. Впрочем, далеко не всегда коррективы приходится вносить из-за ошибок или недоработки при реализации идей и решений. Очень часто потребность в переменах становится следствием изменения внешней среды, требований к работе или появлением новых, более эффективных решений для реализации уже поставленных перед специалистами целей и задач.

Грамотно реализованные ИТ-процессы - это инвестиционный багаж, который со временем даст неплохие дивиденды. Ведь эффективные высокотехнологичные решения не только дают компании преимущества в ходе её работы, но и позволяют заметно повысить уровень её оценочной стоимости на рынке. Ведь стабильно и успешно работающие бизнес-проекты, имеющие высокотехнологичные составляющие, всегда ценятся выше требующих оптимизации и переоснащения конкурентов.

Методы контроля и оценки ИТ-процессов

* конфигурационный процесс управления ориентирован на сбор и систематизацию данных об инфраструктуре, связях, комплектующих ИТ-сервисов;
* процесс, контролирующий уровень качества обслуживания, предполагает сбор данных и выявление несоответствий относительно заявленного качества обслуживания в ИТ-сфере, а также устранение выявленных дефектов;
* процесс, контролирующий работу с пользовательскими запросами и фиксацию фактов нарушений в работе ИТ-сервисов, предполагает своевременное выявление проблем и скорейшее их решение;
* процесс управления проблемами реализует задачи выявления и устранения причин случившихся инцидентов в их основе, не "косметическим", поверхностным путём, а глобально, с поисками и решением сути проблемы.

***Письменно дайте ответ: Каким образом оценивается реализация ИТ-процессов?***