



Переходим на mono

ИЛИ КАК ЭТО БЫЛО

Елизавета Голенок

twitter: @marmothetka, mail: golenok-ea5@narod.ru

Введение

Сентябрь, 2016 год:

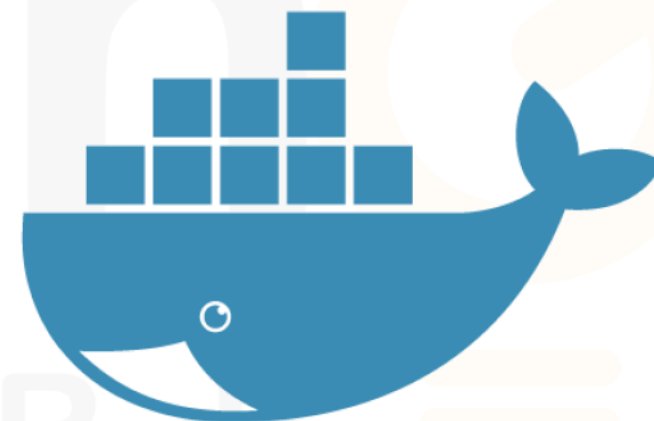


Задача – отказаться от ОС Windows, как от инструмента разработки, так и от серверной ОС



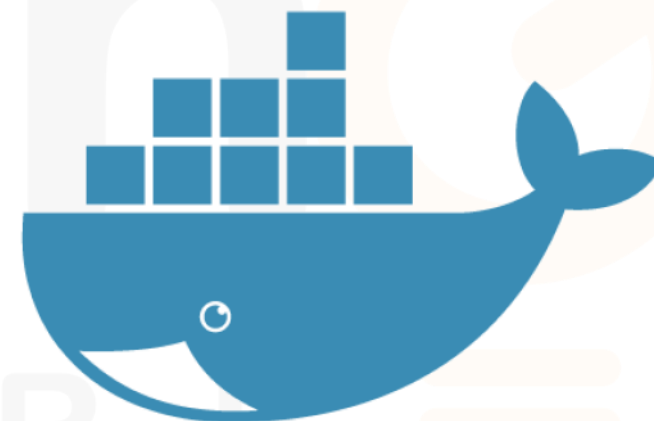
Цель и основные задачи

- Цель - отказаться от Windows в целом



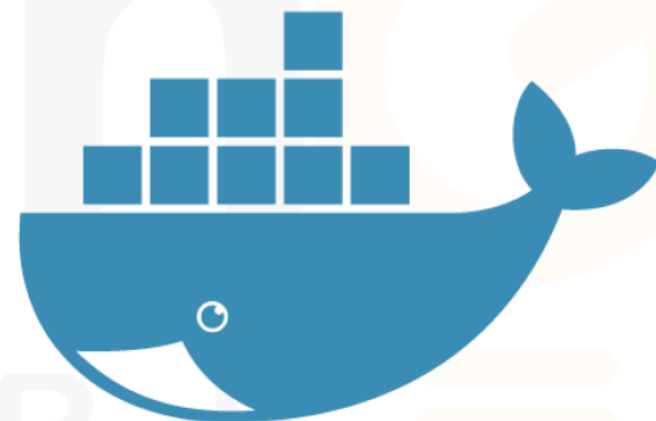
Цель и основные задачи

- Цель - отказаться от Windows в целом
- Задачи:
 - 1. Исследовать возможности переноса большого legacy проекта под Linux



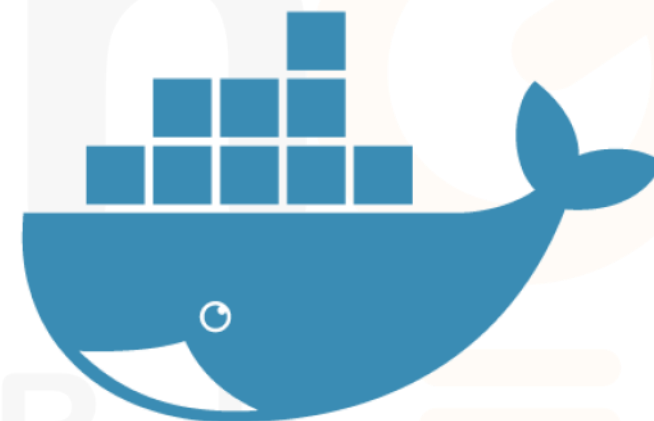
Цель и основные задачи

- Цель - отказаться от Windows в целом
- Задачи:
 - 1. Исследовать возможности переноса большого legacy проекта под Linux
 - 2. Исследовать инструменты для разработки под Mac OS



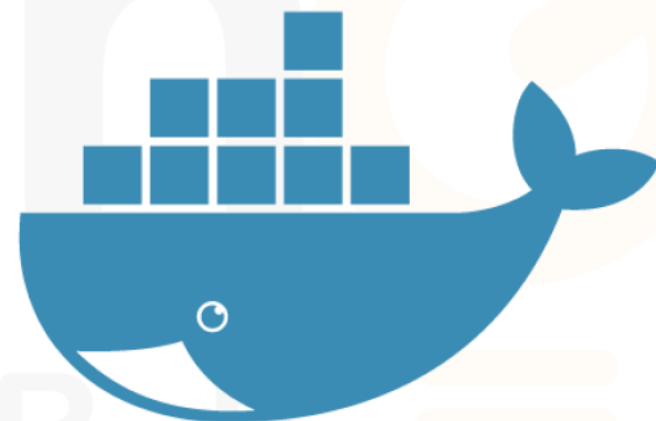
Цель и основные задачи

- Цель - отказаться от Windows в целом
- Задачи:
 - 1. Исследовать возможности переноса большого legacy проекта под Linux
 - 2. Исследовать инструменты для разработки под Mac OS
 - 3. Разобраться с Docker и с SQL Server



Цель и основные задачи

- Цель - отказаться от Windows в целом
- Задачи:
 - 1. Исследовать возможности переноса большого legacy проекта под Linux
 - 2. Исследовать инструменты для разработки под Mac OS
 - 3. Разобраться с Docker и с SQL Server
 - 4. Выбрать веб-сервер для решений под Linux



Выбор средств для разработки ПО под Mac OS

IDE	Достоинства	Недостатки
VS Code	Легковесность, модульность	Долго собирать различные плагины для разработки

Выбор средств для разработки ПО под Mac OS

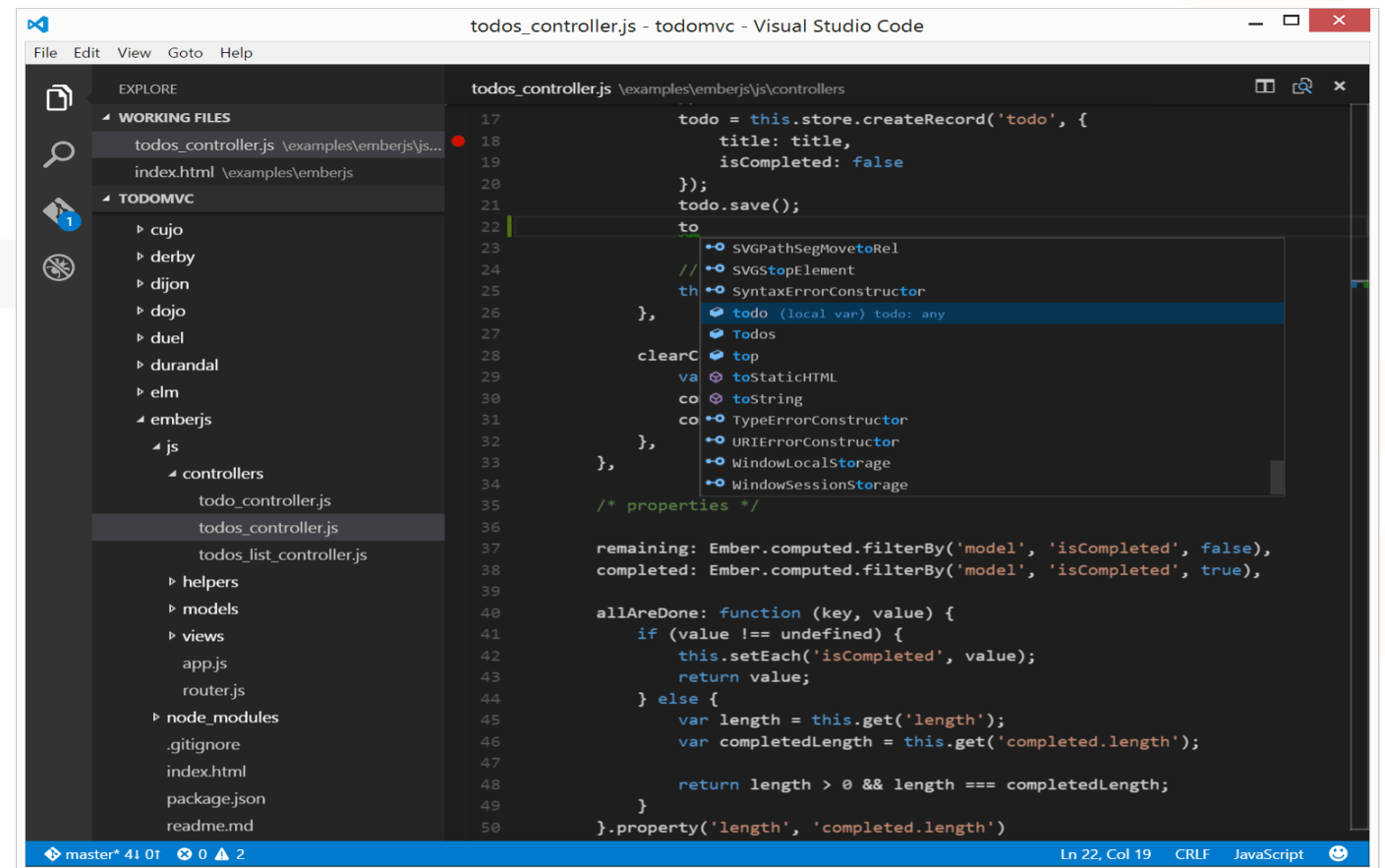
IDE	Достоинства	Недостатки
VS Code	Легковесность, модульность	Долго собирать различные плагины для разработки
VS for Mac	Решение из коробки, есть базовые инструменты для отладки приложения	Usability и скорость разработки в целом

Выбор средств для разработки ПО под Mac OS

IDE	Достоинства	Недостатки
VS Code	Легковесность, модульность	Долго собирать различные плагины для разработки
VS for Mac	Решение из коробки, есть базовые инструменты для отладки приложения	Usability и скорость разработки в целом
Rider	Решение из коробки, привычные хоткеи, скорость разработки	

VS Code и плагины

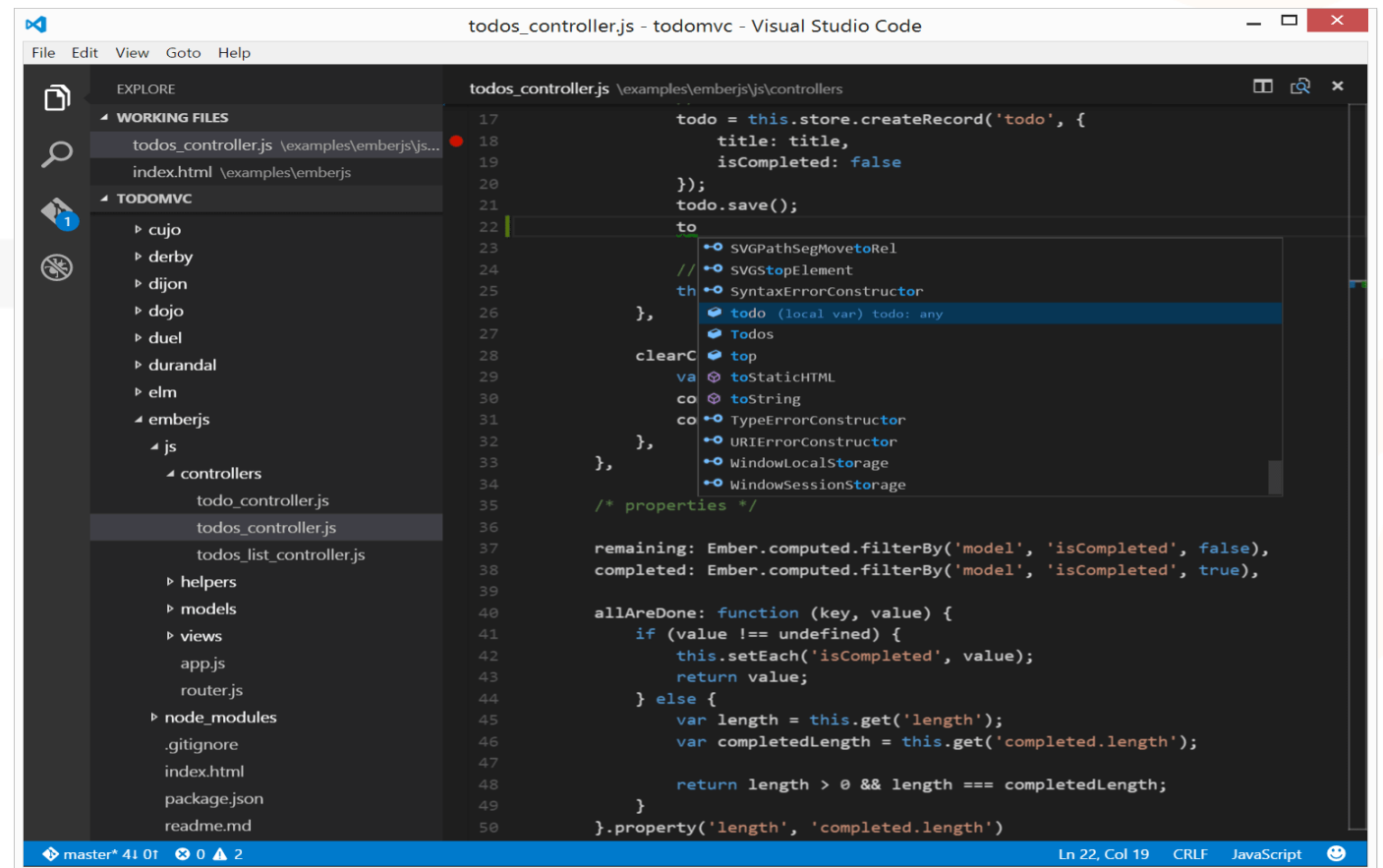
При редактировании C#-проекта VSCode автоматически запускает OmniSharp



VS Code и плагины

При редактировании C#-проекта VSCode автоматически запускает OmniSharp

Для отладки моно-приложений VSCode использует расширение VS Code Mono Debug.

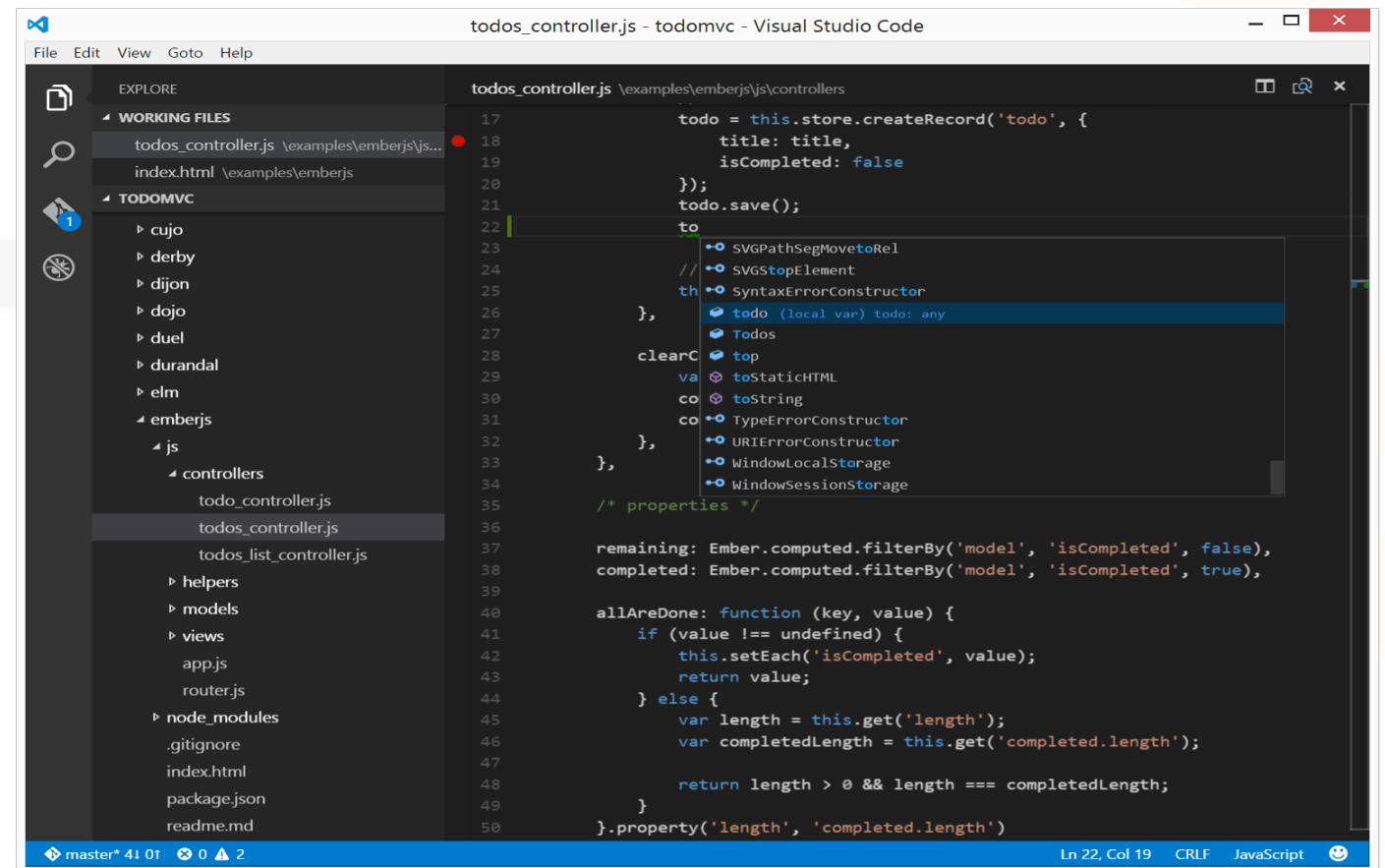


VS Code и плагины

При редактировании C#-проекта VSCode автоматически запускает OmniSharp

Для отладки моно-приложений VSCode использует расширение VS Code Mono Debug.

VS Code Mono Debug – реализация SDB CLI



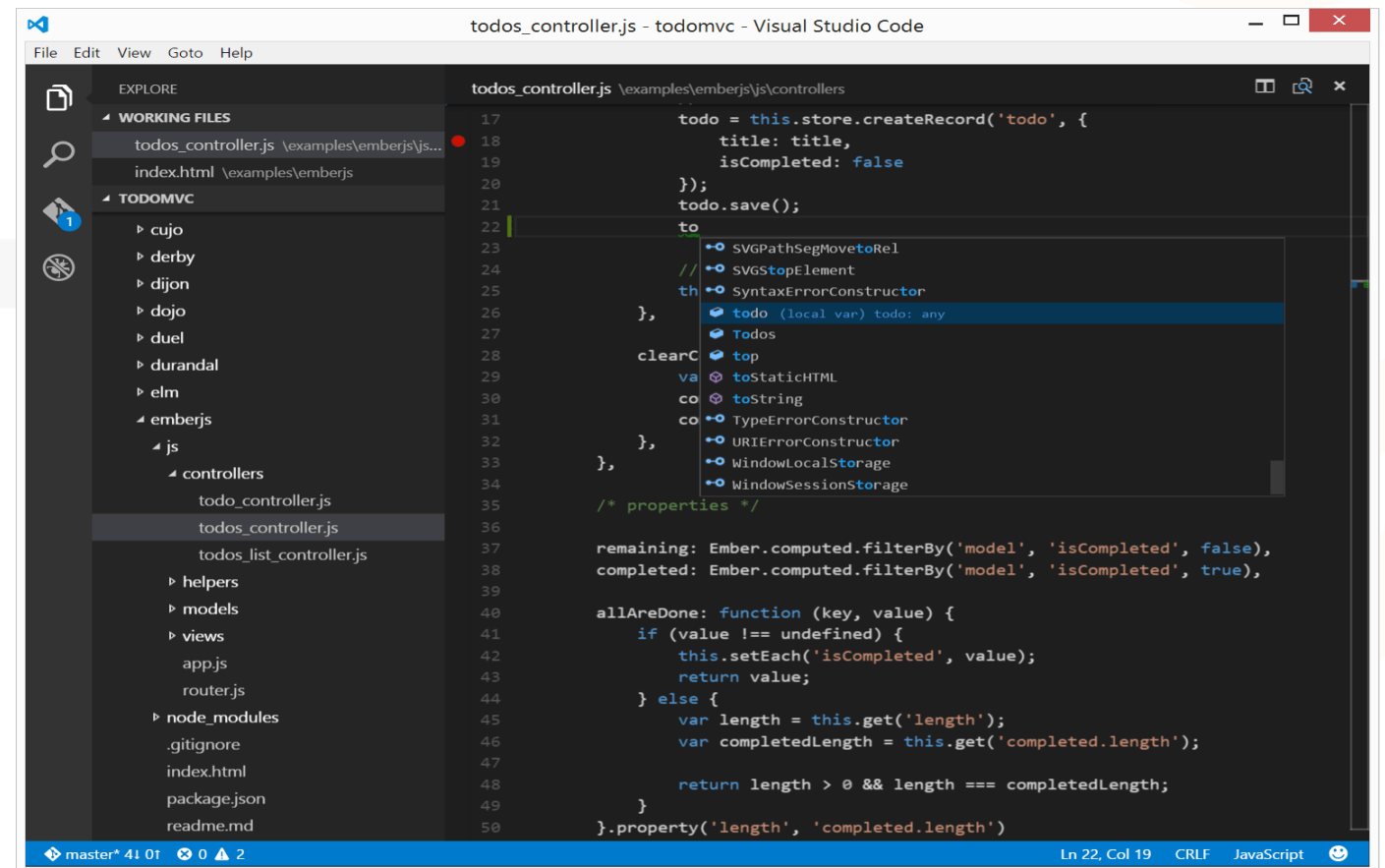
VS Code и плагины

При редактировании C#-проекта VSCode автоматически запускает OmniSharp

Для отладки моно-приложений VSCode использует расширение VS Code Mono Debug.

VS Code Mono Debug – реализация SDB CLI

SDB – Soft Debugger

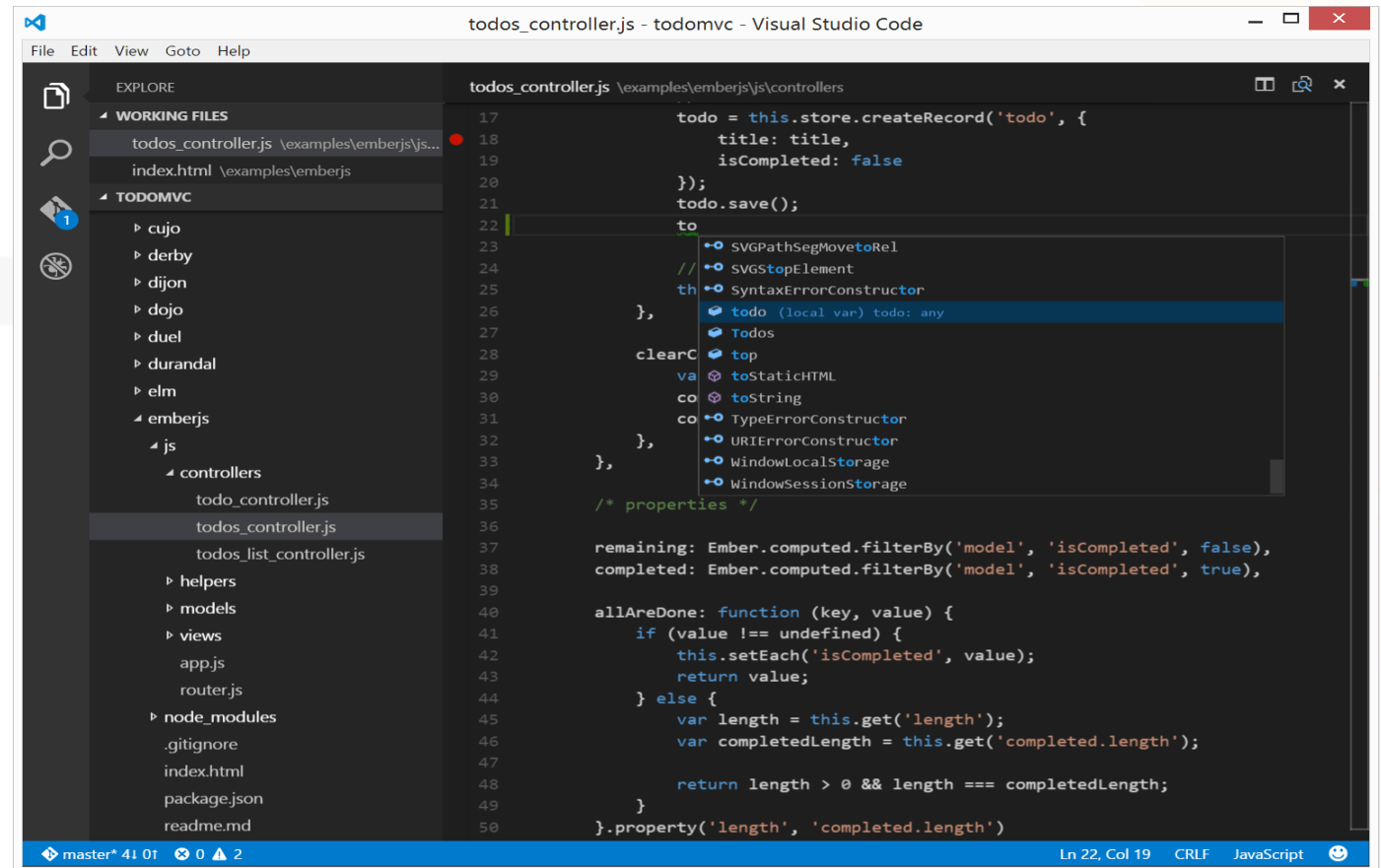


```
File Edit View Goto Help
EXPLORE
WORKING FILES
  todos_controller.js \examples\emberjs\js...
  index.html \examples\emberjs
TODOMVC
  cujo
  derby
  dijon
  dojo
  duel
  durandal
  elm
  emberjs
    js
      controllers
        todos_controller.js
        todos_list_controller.js
      helpers
      models
      views
      app.js
      router.js
    node_modules
    .gitignore
    index.html
    package.json
    readme.md

todos_controller.js \examples\emberjs\js\controllers
17      todo = this.store.createRecord('todo', {
18        title: title,
19        isCompleted: false
20      });
21      todo.save();
22      to
23      //
24      // SVGPathSegMoveToRel
25      // SVGStopElement
26      // SyntaxErrorConstructor
27      // todo (local var) todo: any
28      // Todos
29      // top
30      // toStaticHTML
31      // toString
32      // TypeErrorConstructor
33      // URIErrorConstructor
34      // WindowLocalStorage
35      // WindowSessionStorage
36      /* properties */
37      remaining: Ember.computed.filterBy('model', 'isCompleted', false),
38      completed: Ember.computed.filterBy('model', 'isCompleted', true),
39
40      allAreDone: function (key, value) {
41        if (value !== undefined) {
42          this.setEach('isCompleted', value);
43          return value;
44        } else {
45          var length = this.get('length');
46          var completedLength = this.get('completed.length');
47
48          return length > 0 && length === completedLength;
49        }
50      }.property('length', 'completed.length')
```

VS Code и плагины

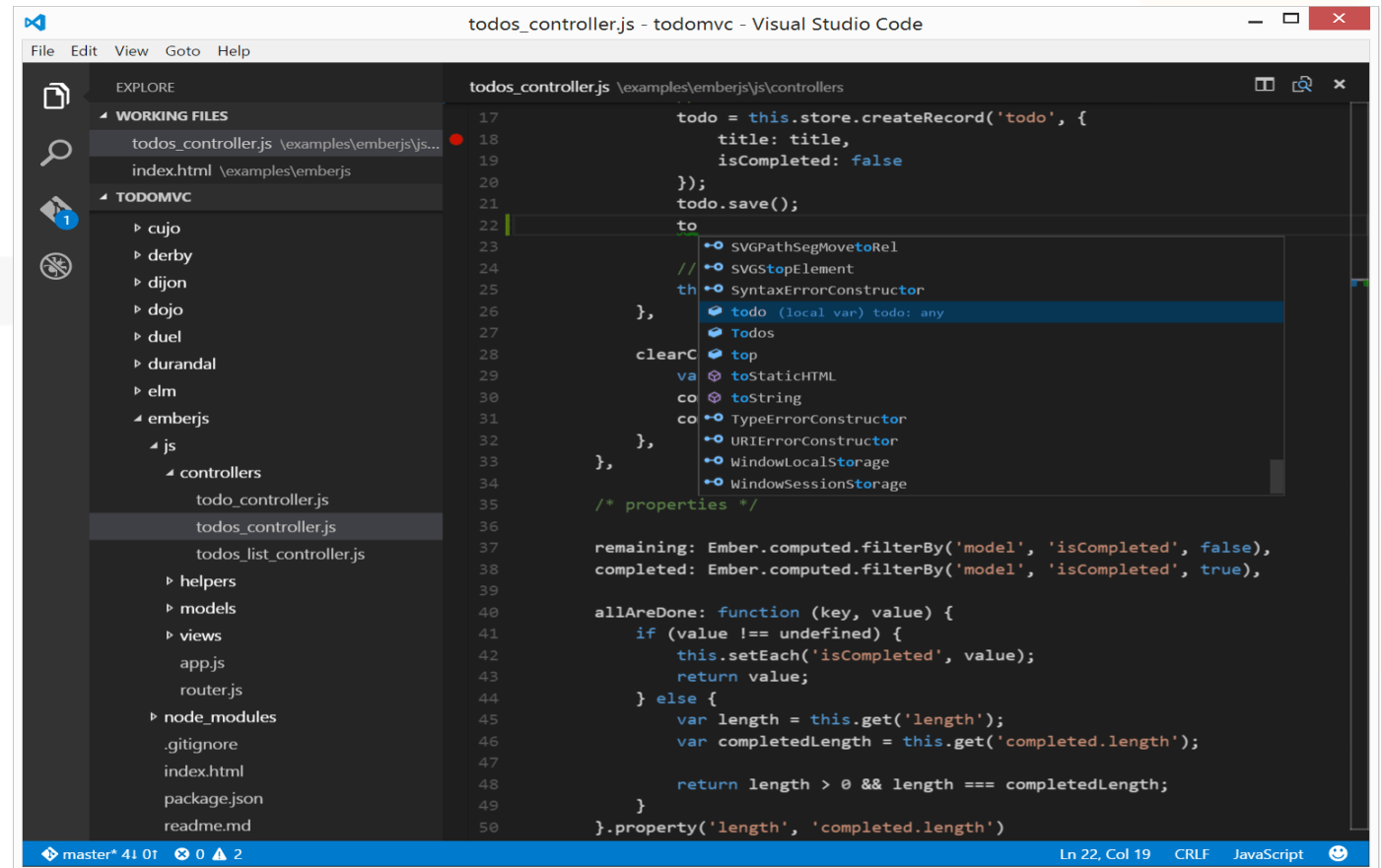
Необходимые шаги для отладки приложения



VS Code и плагины

Необходимые шаги для отладки приложения

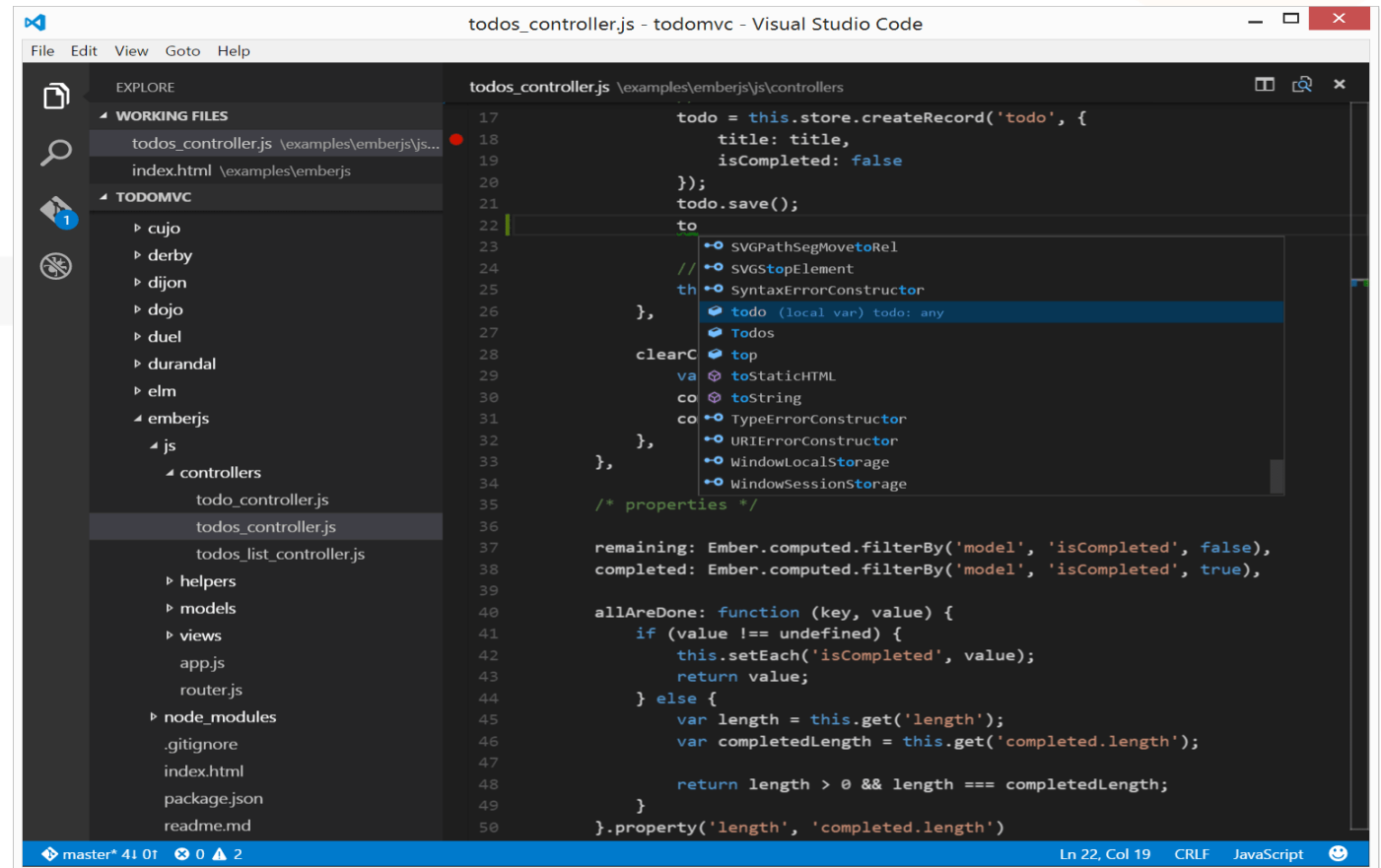
1. Поставить mono, VS Code Mono Debug



VS Code и плагины

Необходимые шаги для отладки приложения

1. Поставить mono, VS Code Mono Debug
2. Скомпилировать проект в режиме -debug



The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following files and folders:

- WORKING FILES
 - todos_controller.js \examples\emberjs\js...
 - index.html \examples\emberjs
- TODOMVC
 - cujo
 - derby
 - dijon
 - dojo
 - duel
 - durandal
 - elm
 - emberjs
 - js
 - controllers
 - todo_controller.js
 - todos_controller.js
 - todos_list_controller.js
 - helpers
 - models
 - views
 - app.js
 - router.js
 - node_modules
 - .gitignore
 - index.html
 - package.json
 - readme.md

The code editor shows the contents of `todos_controller.js` with the following code:

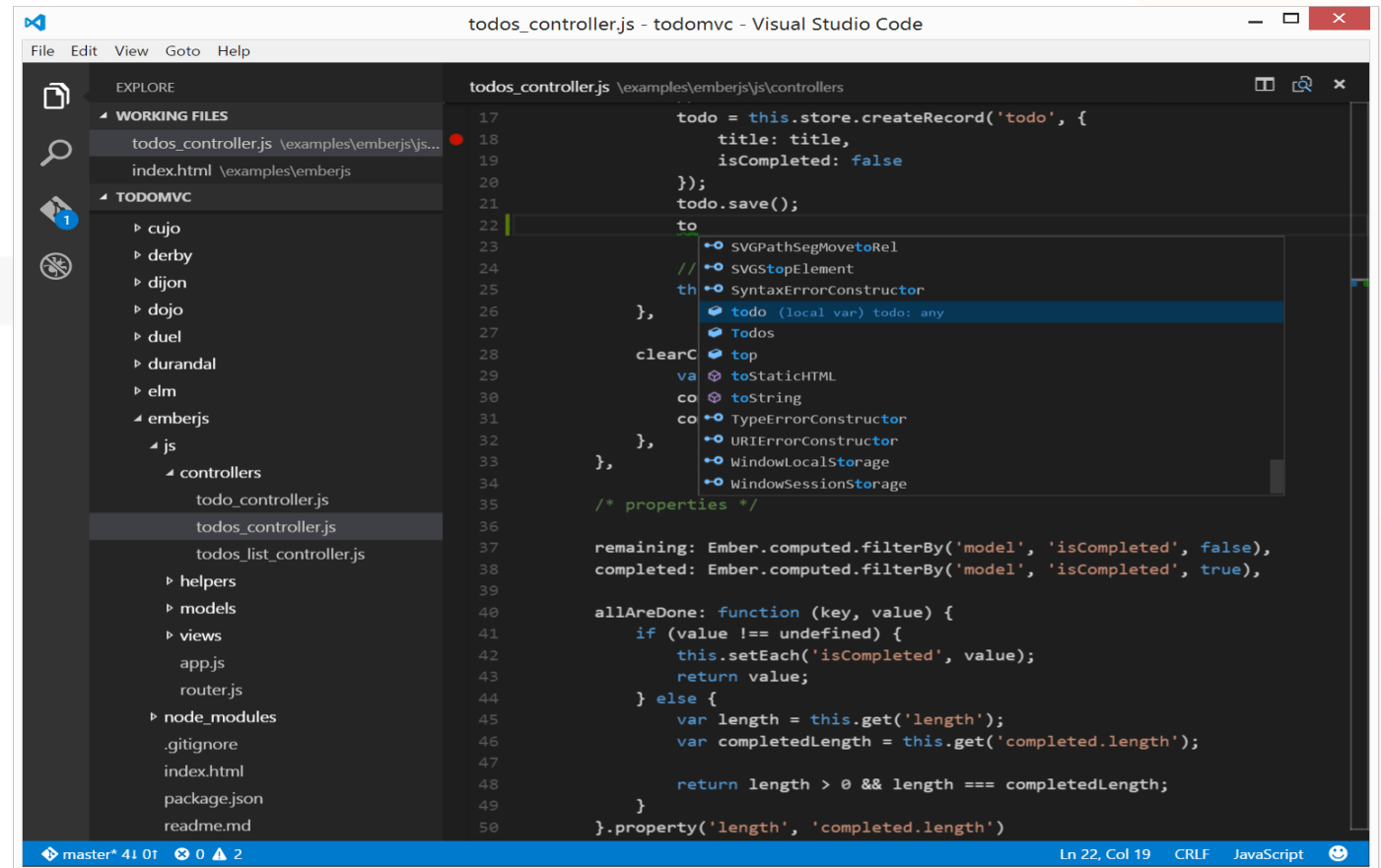
```
17 todo = this.store.createRecord('todo', {
18   title: title,
19   isCompleted: false
20 });
21 todo.save();
22
23 //
24 // SVGPathSegMoveToRel
25 // SVGStopElement
26 // SyntaxErrorConstructor
27 //
28 // todo (local var) todo: any
29 // Todos
30 // top
31 //
32 // va toStaticHTML
33 // co toString
34 // co TypeErrorConstructor
35 // URIErrorConstructor
36 // WindowLocalStorage
37 // WindowSessionStorage
38
39 /* properties */
40
41 remaining: Ember.computed.filterBy('model', 'isCompleted', false),
42 completed: Ember.computed.filterBy('model', 'isCompleted', true),
43
44 allAreDone: function (key, value) {
45   if (value !== undefined) {
46     this.setEach('isCompleted', value);
47     return value;
48   } else {
49     var length = this.get('length');
50     var completedLength = this.get('completed.length');
51
52     return length > 0 && length === completedLength;
53   }
54 }
55 }.property('length', 'completed.length')
```

The status bar at the bottom shows the current file is `master* 41 01` with 0 errors and 2 warnings. The cursor is at line 22, column 19. The language is set to JavaScript.

VS Code и плагины

Необходимые шаги для отладки приложения

1. Поставить mono, VS Code Mono Debug
2. Скомпилировать проект в режиме -debug
3. Присоединиться к процессу для дальнейшей отладки

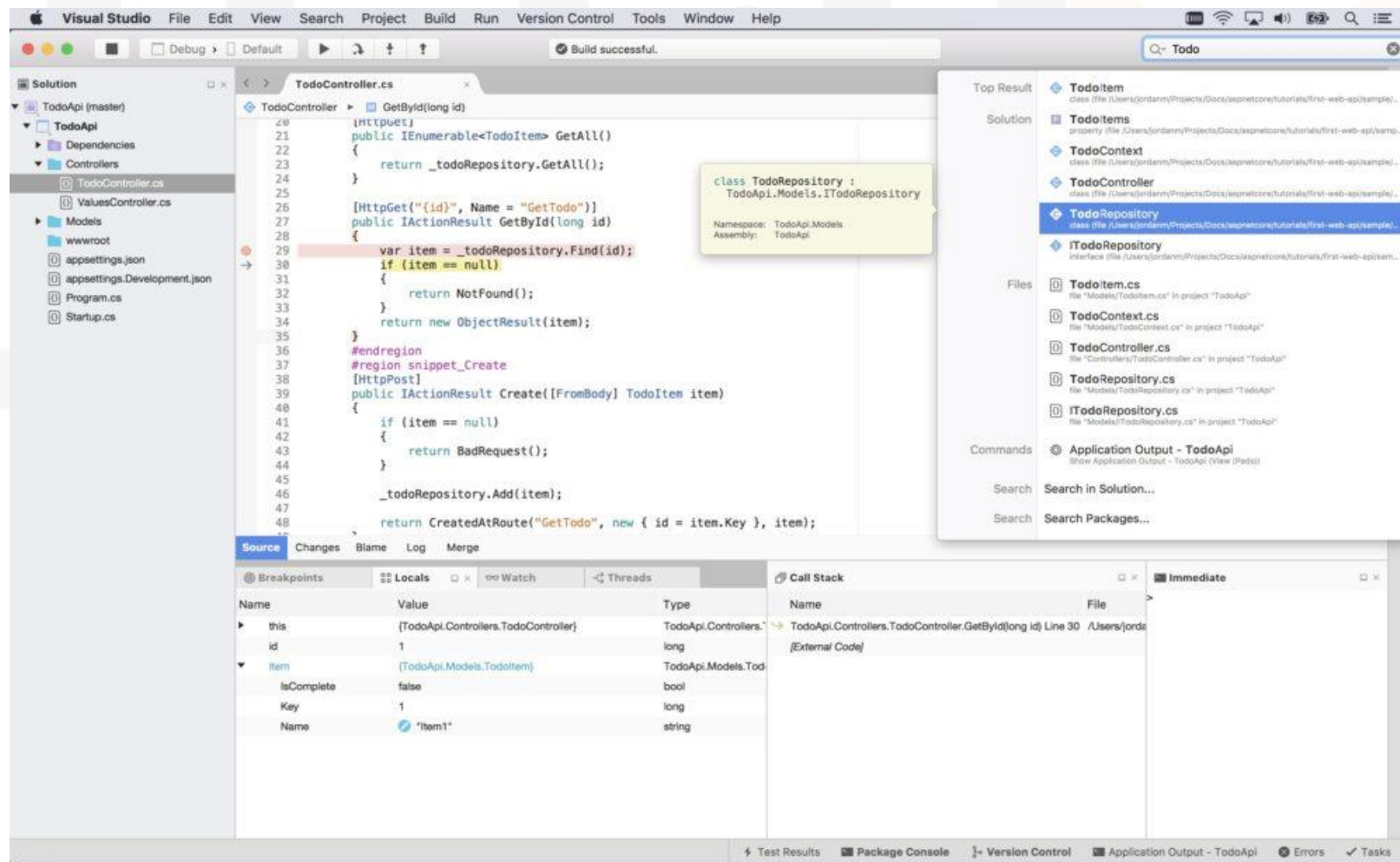


Visual Studio Code interface showing the Ember.js todos_controller.js file. The left sidebar displays the file explorer with the project structure. The main editor area shows the code for the controller, including the createRecord method and the filterBy method. A tooltip is visible over the 'todo' variable, showing its type as 'any'.

```
17  todo = this.store.createRecord('todo', {
18      title: title,
19      isCompleted: false
20  });
21  todo.save();
22  to
23  //
24  // SVGPathSegMoveToRel
25  // SVGStopElement
26  // SyntaxErrorConstructor
27  //
28  },
29  clearC
30  va
31  co
32  co
33  },
34  /* properties */
35  remaining: Ember.computed.filterBy('model', 'isCompleted', false),
36  completed: Ember.computed.filterBy('model', 'isCompleted', true),
37
38  allAreDone: function (key, value) {
39      if (value !== undefined) {
40          this.setEach('isCompleted', value);
41          return value;
42      } else {
43          var length = this.get('length');
44          var completedLength = this.get('completed.length');
45
46          return length > 0 && length === completedLength;
47      }
48  }.property('length', 'completed.length')
49
50
```

От MonoDevelop до VS for Mac

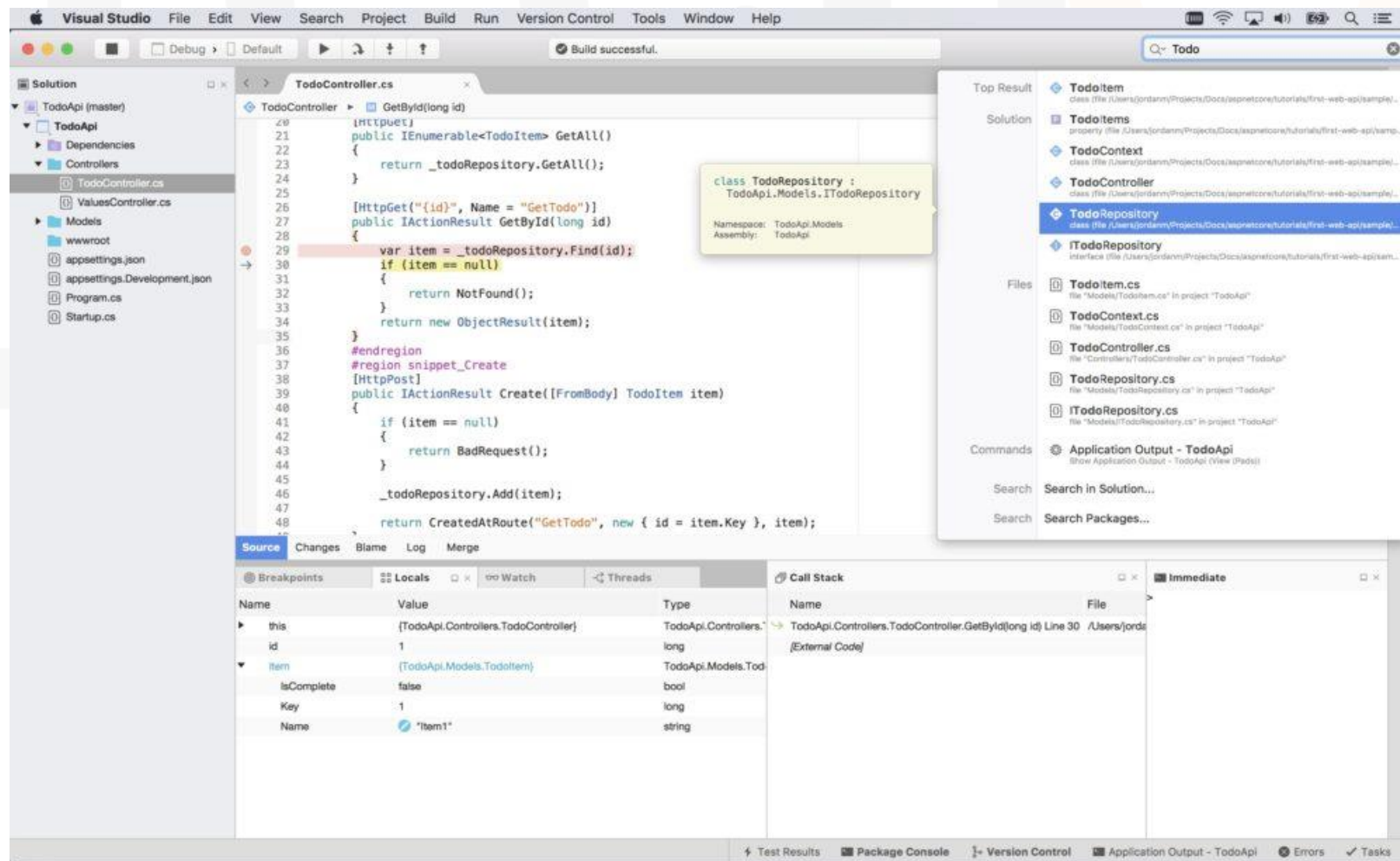
Сначала был
MonoDevelop



От MonoDevelop до VS for Mac

Сначала был
MonoDevelop

2011 год, Xamarin

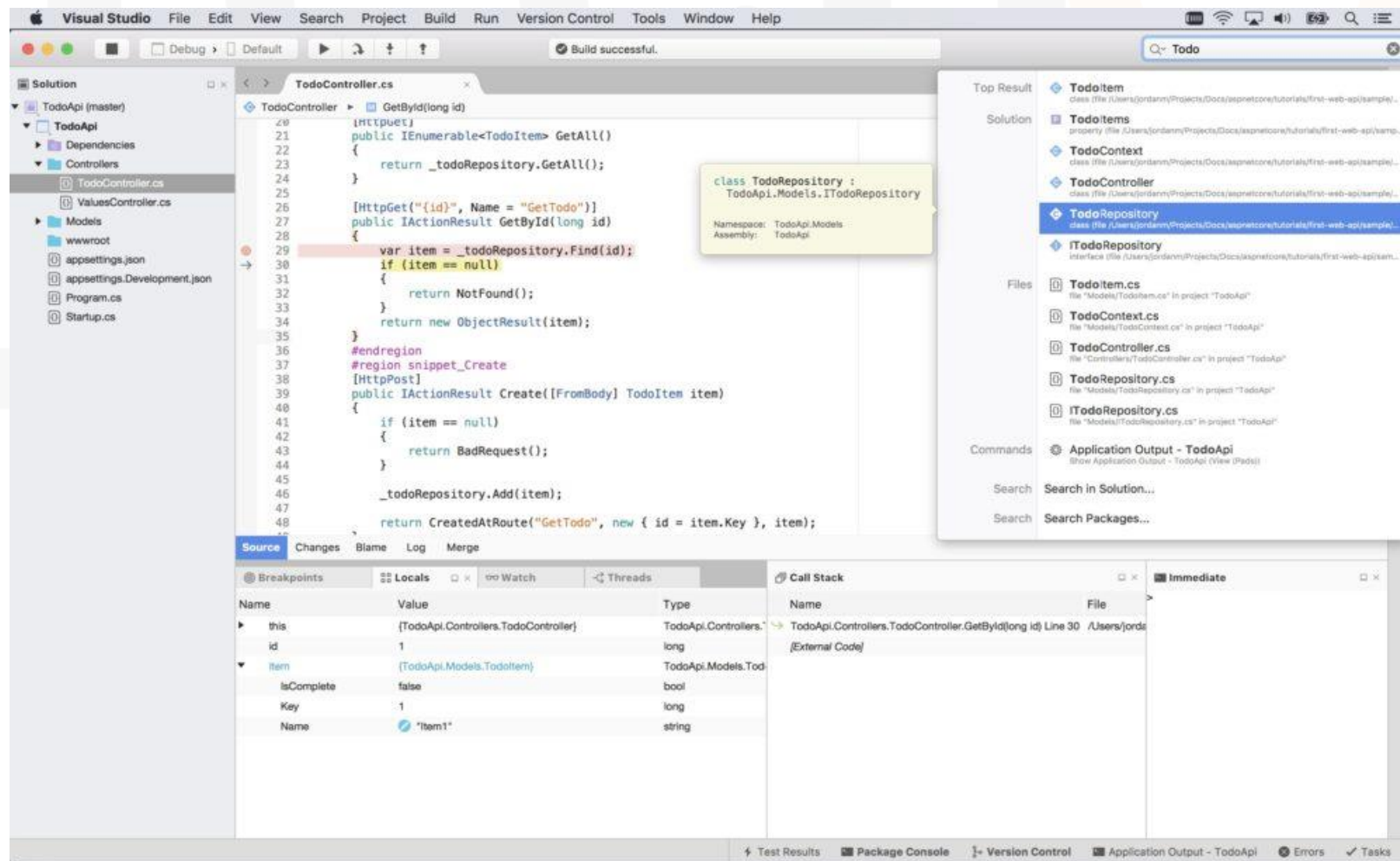


От MonoDevelop до VS for Mac

Сначала был
MonoDevelop

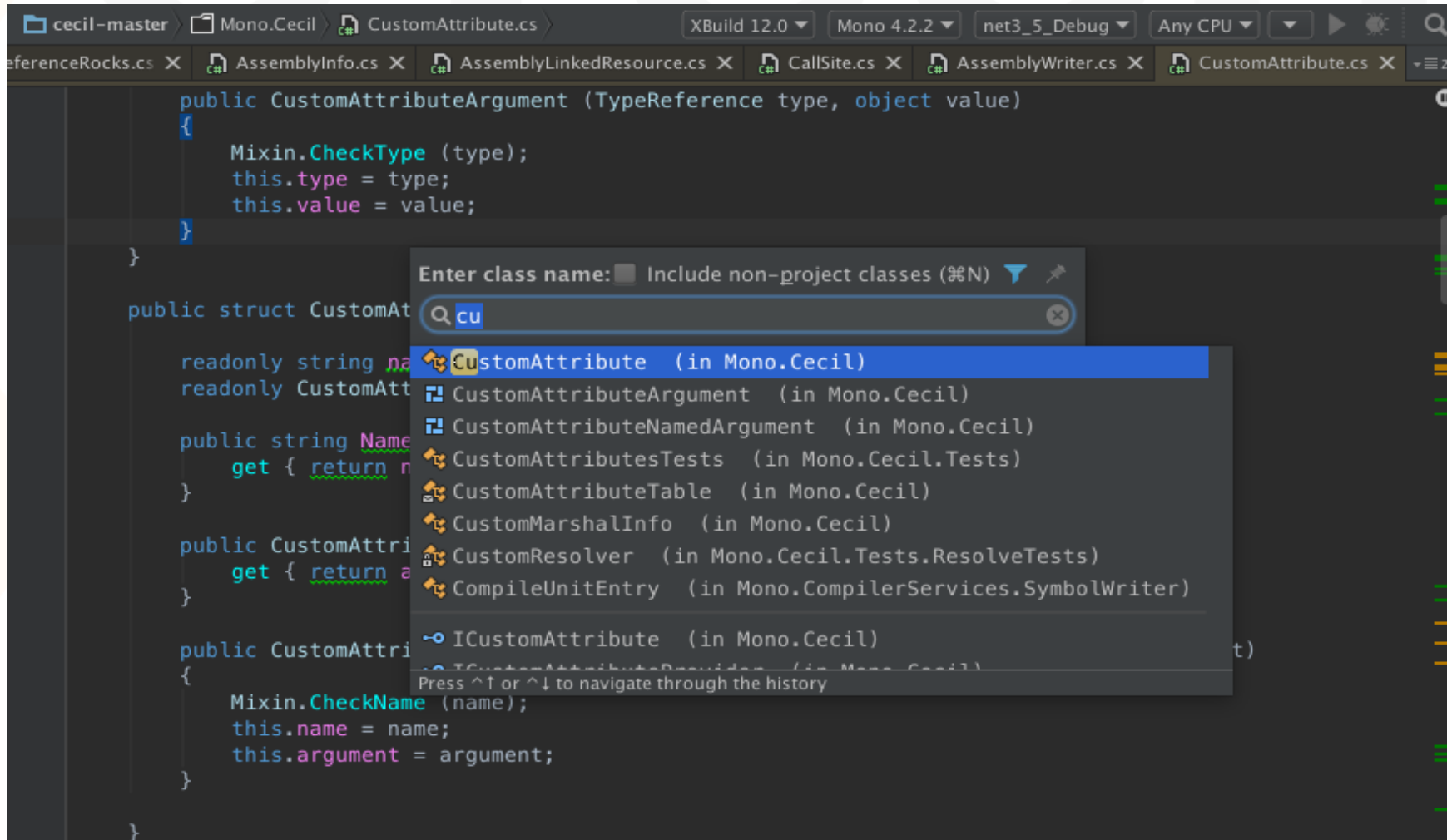
2011 год, Xamarin

2016 год, Microsoft



Rider

Кроссплатформенная IDE от JetBrains. Релиз – август 2017 года.
Работает только на x64



Почему мы выбрали mono?

Критерий	.Net Core	Mono	.Net Framework
Предназначение	Облачные нагрузки	Мобильные, десктоп и веб-приложения	Десктоп и веб-приложения

Почему мы выбрали mono?

Критерий	.Net Core	Mono	.Net Framework
Предназначение	Облачные нагрузки	Мобильные, десктоп и веб-приложения	Десктоп и веб-приложения
Кроссплатформенность	Есть	Есть	Нет

Почему мы выбрали mono?

Критерий	.Net Core	Mono	.Net Framework
Предназначение	Облачные нагрузки	Мобильные, десктоп и веб-приложения	Десктопные и веб-приложения
Кроссплатформенность	Есть	Есть	Нет
Интерфейсы API	Часть интерфейсов отличаются именами сборок, регистрами формы типов и др.	Реализована большая часть API	API

Mono или .Net Core?

- Mono - версия 4.8.0
- Net Core - версия 1.1



Что нужно перенести	Mono	Net Core 1.1	Net Core 2.0
WCF	+ (ограниченный функционал)	-	-
WebForms	+	-	-
WebAPI	+	+	+
MVC	+	+	+

Первые проблемы при переходе на mono

- References

Первые проблемы при переходе на mono

- References
- Web targets

Первые проблемы при переходе на mono

- References
- Web targets
- Bindings

Первые проблемы при переходе на mono

- References
- Web targets
- Bindings
- NHibernate

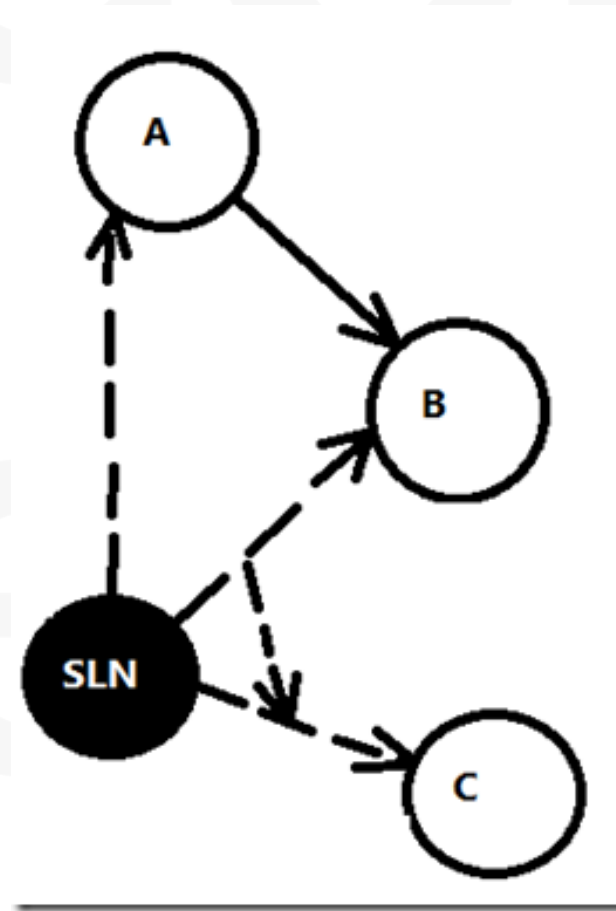
References. Возможные проблемы

- Старая версия MSBuild допускает наличие циклических ссылок в .sln
- Основные проблемы - case-зависимости:
 - Admin.csproj != admin.csproj
 - **MONO_IOMAP=case** xbuild foo.sln
- Проблемы с PreBuild, Copy, Exes при работе с cmd

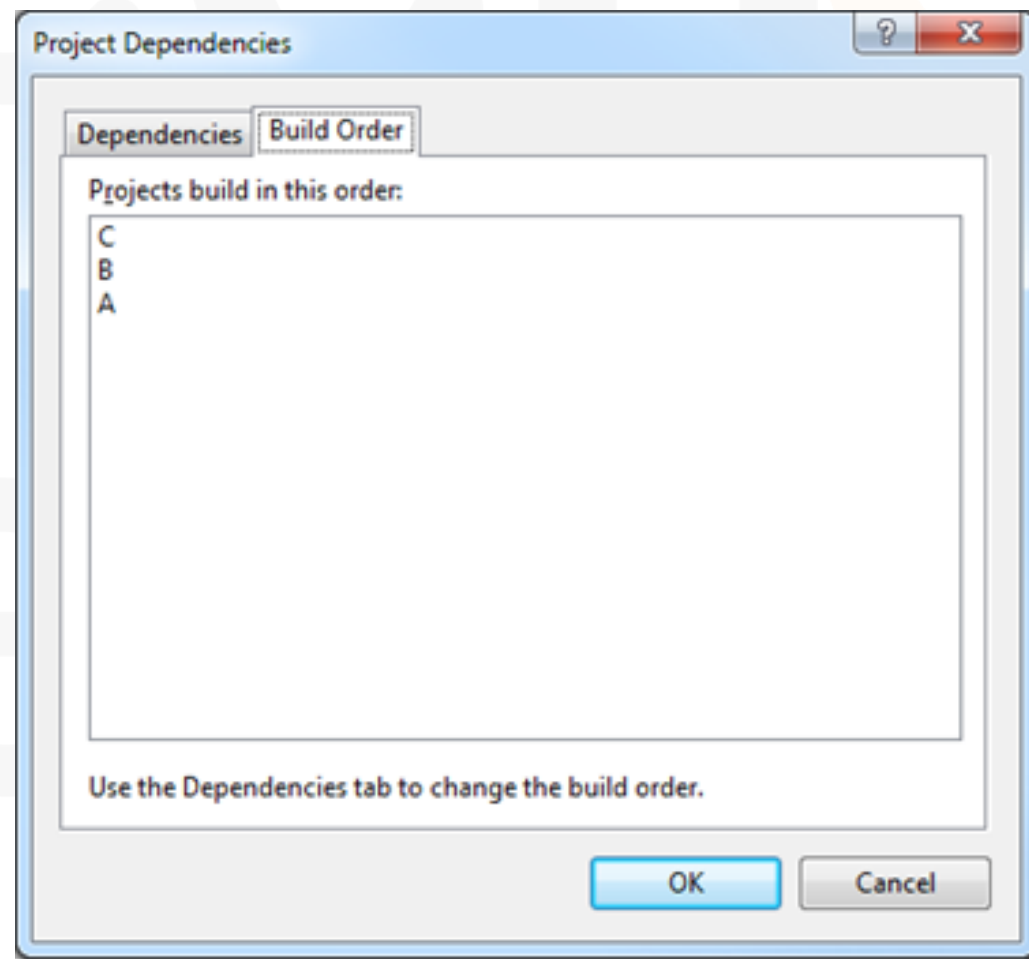
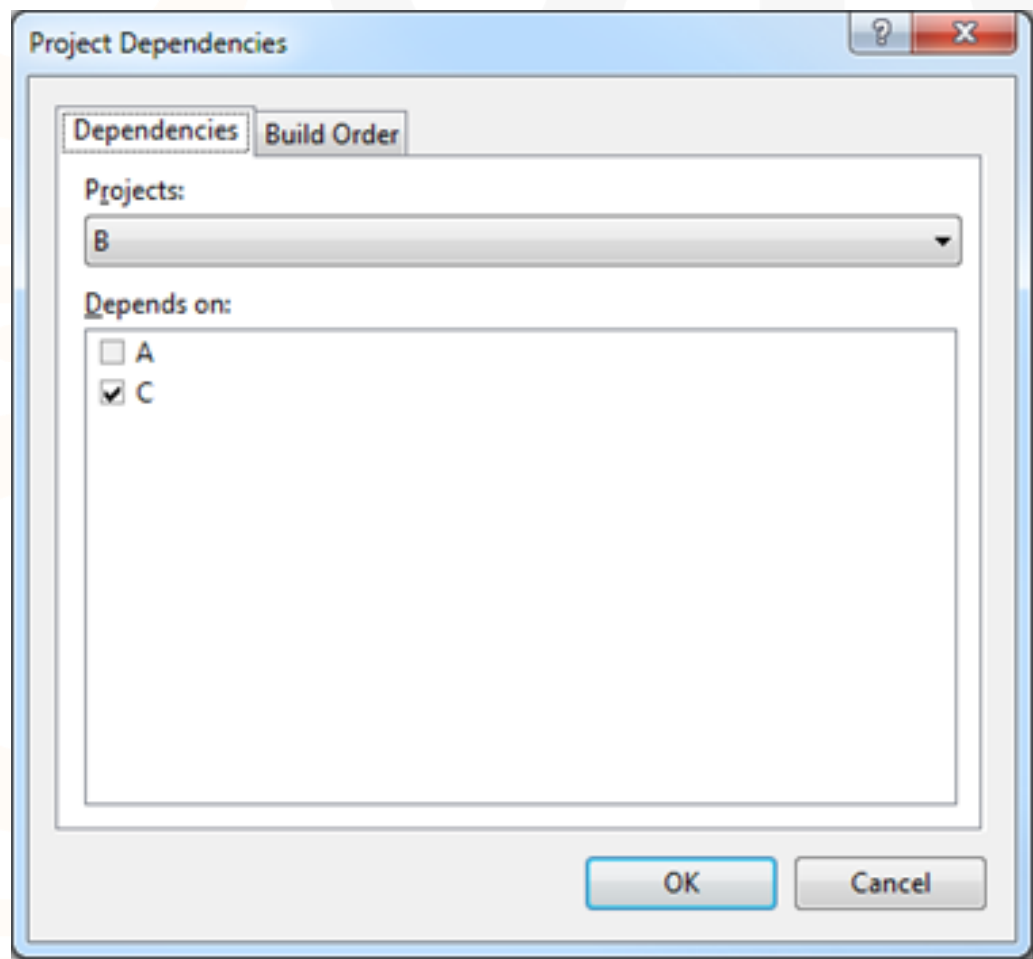
References. Возможные проблемы

Что у нас есть?

- Проекты A, B, C
- Файл решения .sln

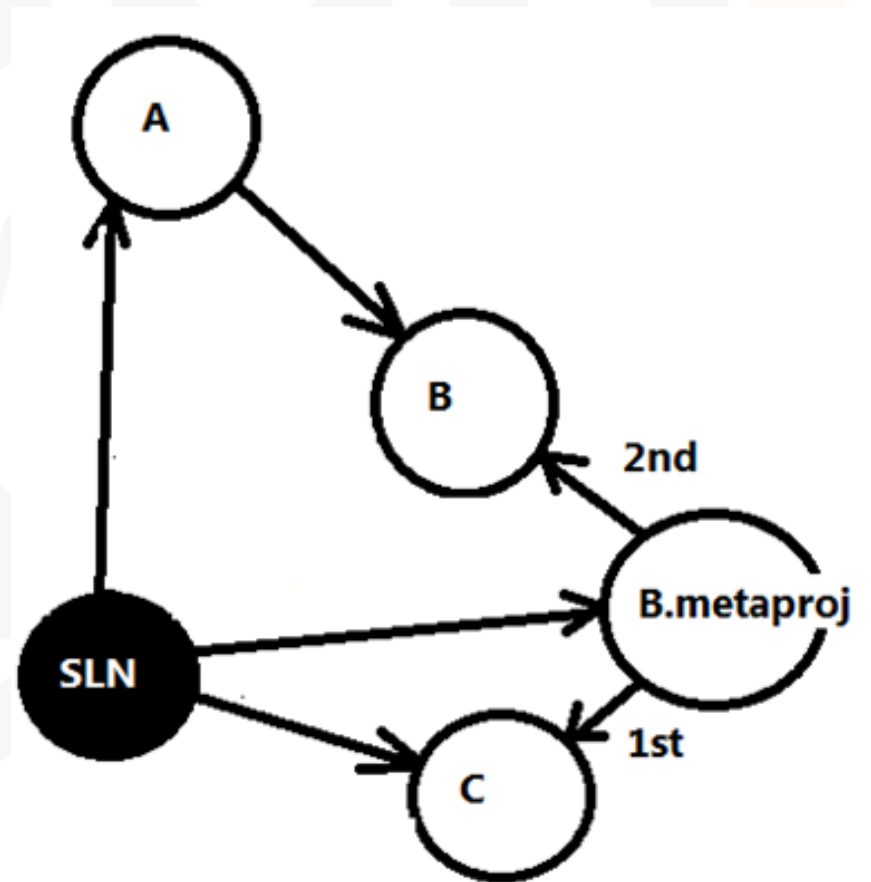


References. Возможные проблемы



References. Возможные проблемы

MS Build ничего не знал о ваших референсах, пока не начинал свою работу.



References. Возможные проблемы

Почему это работало раньше?

References. Возможные проблемы

Почему это работало раньше?

- В предыдущих версиях загружались и проверялись все файлы проекта, указанные в .sln

References. Возможные проблемы

Почему это работало раньше?

- В предыдущих версиях загружались и проверялись все файлы проекта, указанные в .sln
- Затем строился граф эквивалента MSBuild для .sln

References. Возможные проблемы

Почему это работало раньше?

- В предыдущих версиях загружались и проверялись все файлы проекта, указанные в .sln
- Затем строился граф эквивалента MSBuild для .sln
- Это было сделано для взаимодействия со старыми не MSBuild-проектами, например: .vcproj

References. Возможные проблемы

Как это пофиксить?

References. Возможные проблемы

Как это пофиксить?

- Отказаться от ссылок в файле .sln

References. Возможные проблемы

Как это пофиксить?

- Отказаться от ссылок в файле .sln
- Выражать зависимости в файлах проекта

References. Возможные проблемы

Как это пофиксить?

- Отказаться от ссылок в файле .sln
- Выражать зависимости в файлах проекта
- С помощью **<ItemGroup>** можно реализовать выполнение сборки другого проекта без ссылок

References. Возможные проблемы

- С помощью **<ItemGroup>** можно реализовать выполнение сборки другого проекта без ссылок

```
<ProjectReference Include="... foo.csproj">  
    <ReferenceOutputAssembly>false</ReferenceOutputAssembly>  
</ProjectReference>
```

References. Возможные проблемы

- Если вы используете старые версии VS Studio, то вы будете вынуждены руками добавлять metadata в .sln – файл

```
ProjectSection(ProjectDependencies) = postProject
    {B79CE0B0-565B-4BC5-8D28-8463A05F0EDC} = {B79CE0B0-565B-4BC5-8D28-8463A05F0EDC}

EndProjectSection
```

<https://blogs.msdn.microsoft.com/visualstudio/2010/12/21/incorrect-solution-build-ordering-when-using-msbuild-exe/>

Web targets

MSBuild web targets используются для деплоя приложений типа web/WebApplication на билд-машинах, конфигурируют среду для выполнения задач

Используются для деплоя 3х типов приложений:

1. All files in the project folder
2. All files in the project-file
3. All files to run the app

Web targets

Проблема – для mono 4.8.0 нет Web.Targets

Как пофиксить?

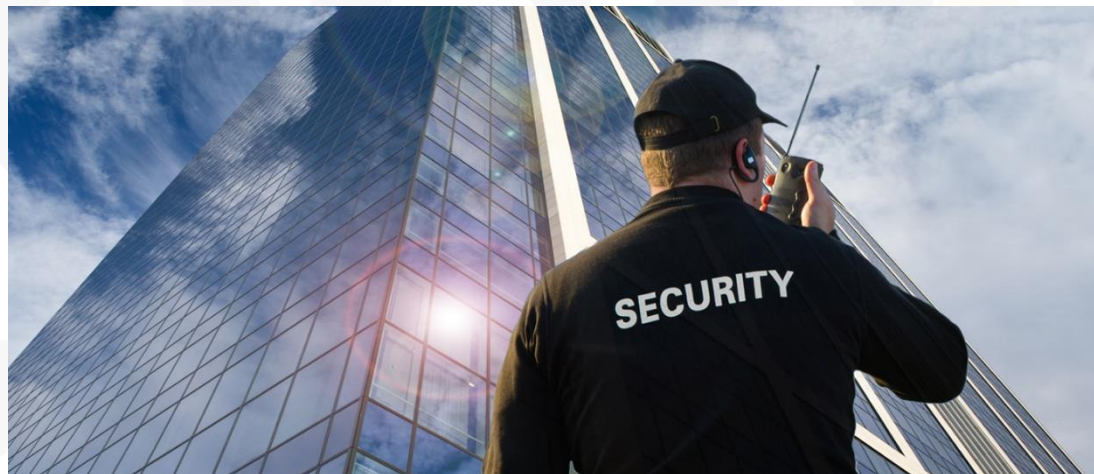
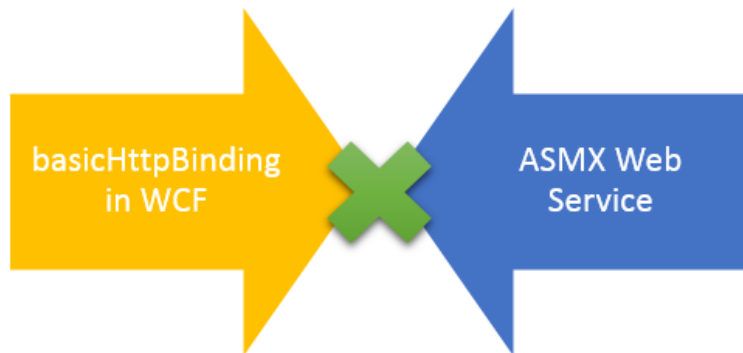
- Поменять в csproj-файлах путь к web targets
- **Скопировать web targets с предыдущей версии для xbuild в необходимые папки**

Copy / Paste

Bindings

WSHttpBinding и др:

- Для группы разработчиков временно заменяем на basicHttpBinding
- По максимуму пишем unit-тесты
- Временно поднимаем прокси (не на топо с биндингами, до окончания реализации необходимых библиотек)



Nhibernate

- TimeSpan
 - Правим конфигурацию на SQL Server

Nhibernate

- TimeSpan
 - Правим конфигурацию на SQL Server
- Assembly.Load
 - Ставим в конфиге **batch size=0**
- https://bugzilla.xamarin.com/show_bug.cgi?id=11199
- https://bugzilla.xamarin.com/show_bug.cgi?id=33728

```
var sysData = Assembly.Load("System.Data, Version=4.0.0.0,  
Culture=neutral, PublicKeyToken=b77a5c561934e089");
```

```
sqlCmdSetType =  
sysData.GetType("System.Data.SqlClient.SqlCommandSet");
```

```
Debug.Assert(sqlCmdSetType != null, "Could not find  
SqlCommandSet!");
```

Nhibernate

- TimeSpan
 - Правим конфигурацию на SQL Server
- Assembly.Load
 - Ставим в конфиге **batch size=0**
- https://bugzilla.xamarin.com/show_bug.cgi?id=11199
- https://bugzilla.xamarin.com/show_bug.cgi?id=33728

```
var sysData = Assembly.Load("System.Data, Version=4.0.0.0,  
Culture=neutral, PublicKeyToken=b77a5c561934e089");
```

```
sqlCmdSetType =  
sysData.GetType("System.Data.SqlClient.SqlCommandSet");
```

```
Debug.Assert(sqlCmdSetType != null, "Could not find  
SqlCommandSet!");
```

Старые проблемы с переносом WebSite project

- Конвертировать проект с WebSite to WebApplication

Старые проблемы с переносом WebSite project

- Конвертировать проект с WebSite to WebApplication
- Столкнуться с одинаковыми namespaces и именами классов

Старые проблемы с переносом WebSite project

- Конвертировать проект с WebSite to WebApplication
- Столкнуться с одинаковыми namespaces и именами классов
- Пофиксить все возможные имена

Старые проблемы с переносом WebSite project

- Конвертировать проект с WebSite to WebApplication
- Столкнуться с одинаковыми namespaces и именами классов
- Пофиксить все возможные имена
- Подтянуть нужные библиотеки из nuget и удалить старые

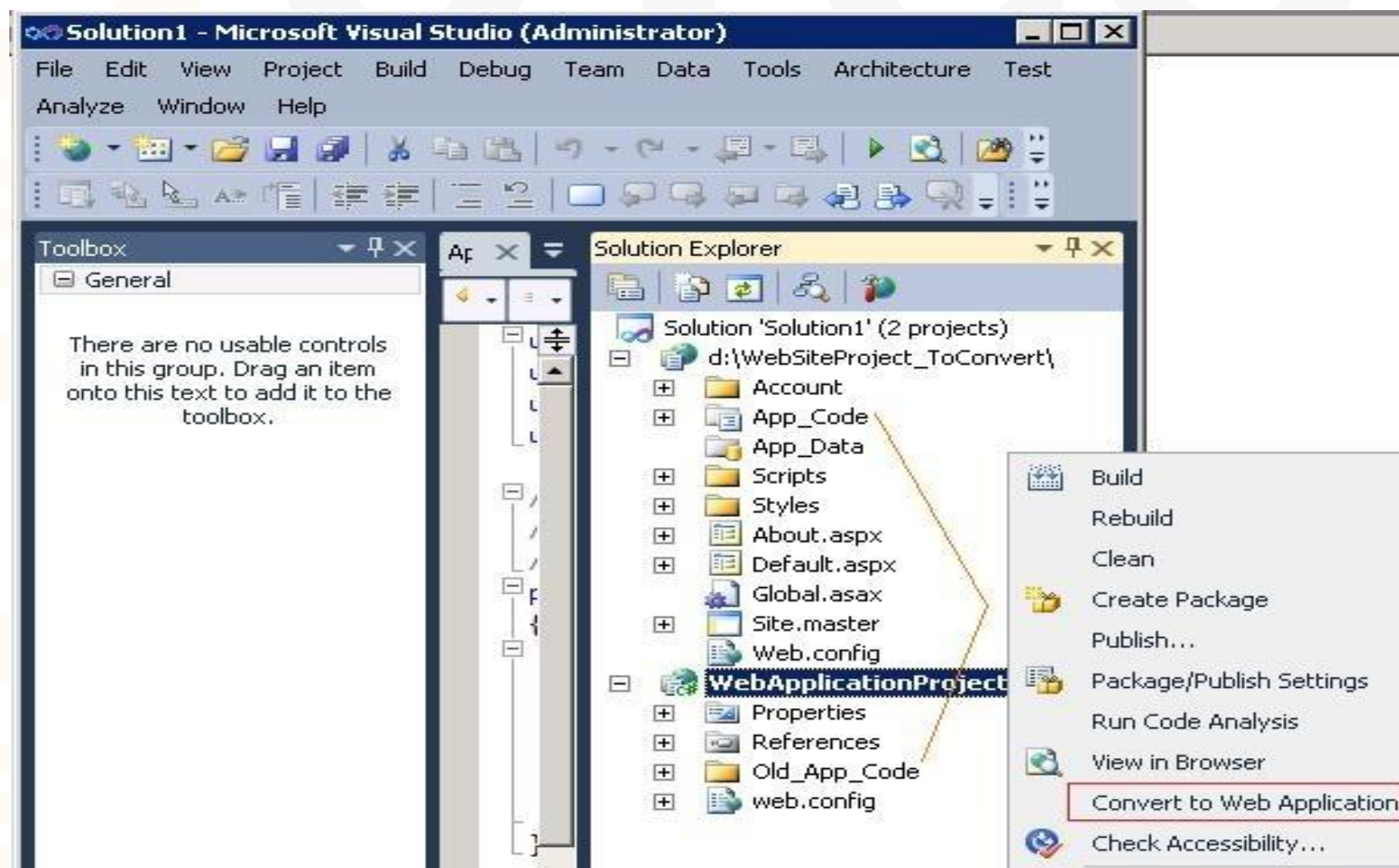
Старые проблемы с переносом WebSite project

- Конвертировать проект с WebSite to WebApplication
- Столкнуться с одинаковыми namespaces и именами классов
- Пофиксить все возможные имена
- Подтянуть нужные библиотеки из nuget и удалить старые
- Столкнуться с несовместимостью фреймворков (так как веб-формы компилируются динамически)

Старые проблемы с переносом WebSite project

- Конвертировать проект с WebSite to WebApplication
- Столкнуться с одинаковыми namespaces и именами классов
- Пофиксить все возможные имена
- Подтянуть нужные библиотеки из nuget и удалить старые
- Столкнуться с несовместимостью фреймворков (так как веб-формы компилируются динамически)
- Перенести под mono

Старые проблемы с переносом WebSite project



Какие еще бывают проблемы при переносе проекта?

- Под UNIX mono эмулирует реестр Windows
- Не работают некоторые Enterprise Services

Выбираем web-server

Название	Преимущества	Недостатки
XSP	Легковесный	Максимум – HTTP 1.0
Apache		mod_mono – часть XSP, долго конфигурируется
nginx	Высокопроизводительный http-сервер, так же может выступать в качестве мощного прокси, быстро конфигурируется	В основе FastCGI

Конфигурируем nginx и mono

```
server {  
    listen 80;  
    server_name www.domain1.xyz;  
    access_log /var/log/nginx/your.domain1.xyz.access.log;  
    root /var/www/www.domain1.xyz/;  
  
    location / {  
        index index.html index.htm default.aspx Default.aspx;  
        fastcgi_index Default.aspx;  
        fastcgi_pass 127.0.0.1:9000;  
        include /etc/nginx/fastcgi_params;  
    }  
}
```

```
fastcgi_param PATH_INFO "";  
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
```

Mono Log Profiler

ENTER: **TestDeadlock:normal_order** ()([0xb49ffb40: 0,04048 1] LEAVE: (wrapper write-barrier) object:wbarrier_noconc (intptr)
)

.....
[0xb7491700: 0,04058 2] ENTER: (wrapper managed-to-native) System.Threading.Thread:JoinInternal
(System.Threading.Thread,int)([System.Threading.Thread:0xb6c06bc8], -1,)

.....
[0xb4c1ab40: 0,04069 2] LEAVE: (wrapper synchronized) System.IO.TextWriter/SyncTextWriter:WriteLine (string)

[0xb49ffb40: 0,04071 1] ENTER: **TestDeadlock:reverse_order** ()()

.....
[0xb4c1ab40: 0,04077 2] ENTER: (wrapper managed-to-native) System.Threading.Thread:SleepInternal (int)(500,)

.....
[0xb49ffb40: 0,54096 2] LEAVE: (wrapper managed-to-native) System.Threading.Thread:SleepInternal (int)
[0xb4c1ab40: 0,54096 2] LEAVE: (wrapper managed-to-native) System.Threading.Thread:SleepInternal (int)

MS SQL Server

- 1988г – SQL Server 1.0 – реляционная СУБД с возможностью работы по локальной сети

MS SQL Server

- 1988г – SQL Server 1.0 – реляционная СУБД с возможностью работы по локальной сети
 - Ashton-Tate занимались разработкой в области баз данных

MS SQL Server

- 1988г – SQL Server 1.0 – реляционная СУБД с возможностью работы по локальной сети
 - Ashton-Tate занимались разработкой в области баз данных
 - Microsoft занималась разработкой в области сетей

MS SQL Server

- 1988г – SQL Server 1.0 – реляционная СУБД с возможностью работы по локальной сети
 - Ashton-Tate занимались разработкой в области баз данных
 - Microsoft занималась разработкой в области сетей
 - Sybase – портирование DataServer на OS/2

MS SQL Server

- 1988г – SQL Server 1.0 – реляционная СУБД с возможностью работы по локальной сети
 - Ashton-Tate занимались разработкой в области баз данных
 - Microsoft занималась разработкой в области сетей
 - Sybase – портирование DataServer на OS/2
- 1990г – SQL Server 1.1 – разработкой занимается только Microsoft

MS SQL Server

- 1988г – SQL Server 1.0 – реляционная СУБД с возможностью работы по локальной сети
 - Ashton-Tate занимались разработкой в области баз данных
 - Microsoft занималась разработкой в области сетей
 - Sybase – портирование DataServer на OS/2
- 1990г – SQL Server 1.1 – разработкой занимается только Microsoft
- 1991-1992г – SQL Server 4.2 – Microsoft&Sybase
 - Движок SQL Server портирован под UNIX

MS SQL Server

- 1988г – SQL Server 1.0 – реляционная СУБД с возможностью работы по локальной сети
 - Ashton-Tate занимались разработкой в области баз данных
 - Microsoft занималась разработкой в области сетей
 - Sybase – портирование DataServer на OS/2
- 1990г – SQL Server 1.1 – разработкой занимается только Microsoft
- 1991-1992г – SQL Server 4.2 – Microsoft&Sybase
 - Движок SQL Server портирован под UNIX
- 1992-1993г – SQL Server для Windows NT

MS SQL Server

- 1994г - Sybase – UNIX, Microsoft – Windows NT, разрыв соглашений

MS SQL Server

- 1994г - Sybase – UNIX, Microsoft – Windows NT, разрыв соглашений
- ... разработка под Windows

MS SQL Server

- 1994г - Sybase – UNIX, Microsoft – Windows NT, разрыв соглашений
- ... разработка под Windows
- 2016г – Docker, Linux и Microsoft SQL Server

MS SQL Server

<https://youtu.be/TdL6twmfCaQ>



Вы хотите сделать свой проект кроссплатформенным?

- По возможности задействуйте **.Net Core**

Вы хотите сделать свой проект кроссплатформенным?

- По возможности задействуйте **.Net Core**
- Будьте готовы, что многое придется **реализовать самим**

Вы хотите сделать свой проект кроссплатформенным?

- По возможности задействуйте **.Net Core**
- Будьте готовы, что многое придется **реализовать самим**
- Будьте готовы **выйти за рамки** SDB и Mono Log Profiler

Вы хотите сделать свой проект кроссплатформенным?

- По возможности задействуйте **.Net Core**
- Будьте готовы, что многое придется **реализовать самим**
- Будьте готовы **выйти за рамки** SDB и Mono Log Profiler
- Не забывайте об **особенностях операционных систем**

Спасибо за внимание



Елизавета Голенок

Gotech Software

twitter: @marmothetka, mail: golenok-ea5@narod.ru