Лекция 3 Методы отладки программного обеспечения

Интегрированные средства отладки

Большинство современных сред программирования (Delphi, Builder C++, Visual Studio и т.д.) включают средства отладки, которые обеспечивают максимально эффективную отладку. Они позволяют:

- выполнять программу по шагам, причем как с заходом в подпрограммы, так и выполняя их целиком;
- предусматривать точки останова;
- выполнять программу до оператора, указанного курсором;
- отображать содержимое любых переменных при пошаговом выполнении;
- отслеживать поток сообщений и т.п.

Применять интегрированные средства в рамках среды достаточно просто. Используют разные приемы в зависимости от проявлений ошибки. Если получено сообщение об ошибке, то сначала уточняют, при выполнении какого оператора программы оно получено. Для этого устанавливают точку останова в начало фрагмента, в котором проявляется ошибка, и выполняют операторы в пошаговом режиме до проявления ошибки. Аналогично поступают при "зависании" компьютера. Если получены неправильные результаты, то локализовать ошибку обычно существенно сложнее. В этом случае сначала определяют фрагмент, при выполнении которого получаются неправильные результаты. Для этого последовательно проверяют интересующие значения в узловых точках. Обнаружив значения, отличающиеся от ожидаемых, по шагам трассируют соответствующий фрагмент до выявления оператора, выполнение которого дает неверный результат. Для уточнения природы ошибки возможен анализ машинных кодов, флагов и представления программы и значений памяти в 16-ричном виде. Причину ошибки определяют, используя один из рассмотренных методов. При этом для проверки гипотез также можно использовать интегрированные средства отладки.

Отладка с использованием независимых отладчиков

При отладке программ иногда используют специальные программы - отладчики, которые позволяют выполнить любой фрагмент программы в пошаговом режиме и проверить содержимое интересующих программиста переменных. Как правило такие отладчики позволяют отлаживать программу только в машинных командах, представленных в 16-ричном коде.

Инструменты отладки

Отладчик представляет из себя программный инструмент, позволяющий программисту наблюдать за выполнением исследуемой программы, останавливать и перезапускать её, прогонять в замедленном темпе, изменять значения в памяти и даже, в некоторых случаях, возвращать назад по времени.

Также полезными инструментами в руках программиста могут оказаться:

- Профилировщики. Они позволят определить сколько времени выполняется тот или иной участок кода, а анализ покрытия позволит выявить неисполняемые участки кода.
- API логгеры позволяют программисту отследить взаимодействие программы и Windows API при помощи записи сообщений Windows в лог.
- Дизассемблеры позволят программисту посмотреть ассемблерный код исполняемого файла
- Снифферы помогут программисту проследить сетевой трафик генерируемой программой
- Снифферы аппаратных интерфейсов позволят увидеть данные которыми обменивается система и устройство.
- Логи системы.

Использование языков программирования высокого уровня, таких как Java, обычно упрощает отладку, поскольку содержат такие средства как обработка исключений, сильно облегчающие поиск источника проблемы. В некоторых низкоуровневых языках, таких как ассемблер, ошибки могут приводить к незаметным проблемам — например, повреждениям памяти или утечкам памяти, и бывает довольно трудно определить что стало первоначальной причиной ошибки. В этих случаях, могут потребоваться изощрённые приёмы и средства отладки.

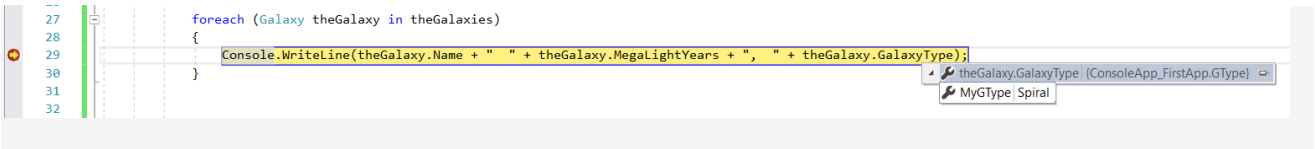
Отладка программ в Visual Studio

При выполнении приложения в отладчике (в так называемом режиме отладки) осуществляется активный мониторинг всего, что происходит во время работы программы. Кроме того, вы можете в любой точке приостановить работу приложения, исследовать его состояние и при необходимости перейти в режим пошагового выполнения, чтобы изучить необходимые строки кода более детально.

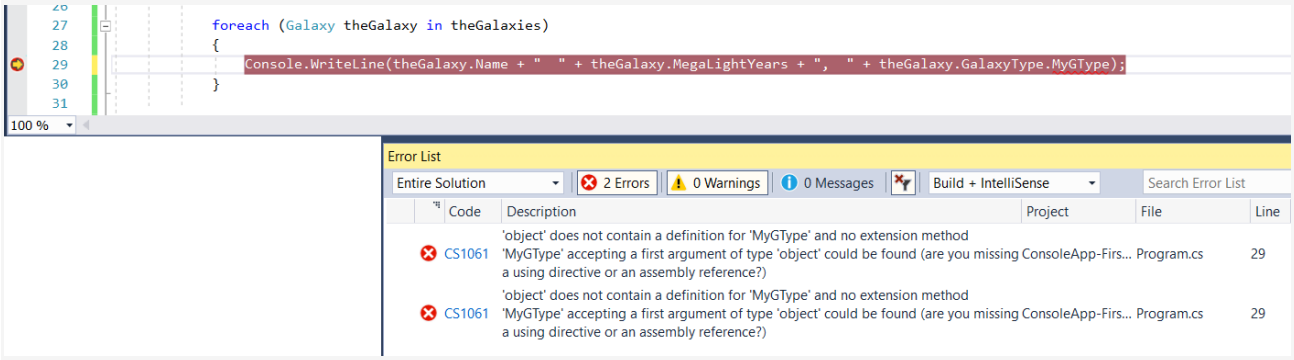
Чтобы перейти в режим отладки в Visual Studio, необходимо нажать клавишу F5 (также вы можете выбрать пункт меню Отладка > Начать отладку или нажать кнопку Начать отладку в панели инструментов "Отладка"). Если возникает исключение, помощник по исправлению ошибок Visual Studio направит вас к точке его появления и предоставит другую необходимую информацию. См. дополнительные сведения об обработке исключений в коде в разделе Приемы и инструменты отладки.

Если исключение не возникает, возможно, вам следует проанализировать определенные места в коде, которые могут являться источником проблемы. На этом этапе следует использовать точки останова в отладчике, благодаря которым вы сможете исследовать код более внимательно. Точки останова — это один из самых простых и важных компонентов надежной отладки. Точка останова указывает, где Visual Studio следует приостановить выполнение кода, чтобы вы могли проверить значения переменных, поведение памяти или последовательность выполнения кода.

Чтобы задать точку останова в Visual Studio, достаточно щелкнуть в левом поле рядом с интересующей вас строкой кода. Также для этого можно поместить указатель мыши в нужную строку и нажать клавишу F9. В месте установки точки останова в левом поле появится красный круг.



Если при внесении изменений будет допущена ошибка, компилятор выведет диагностическое сообщение и покажет место ошибки.



Переход по коду в отладчике с помощью пошаговых команд

Мы указываем сочетания клавиш для большинства команд, так как они ускоряют навигацию по коду вашего приложения. (Эквивалентные команды, например команды меню, приведены в круглых скобках.)

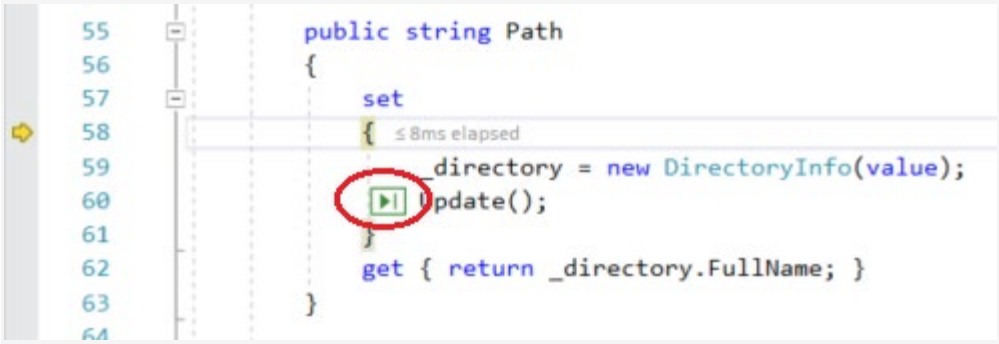
Для запуска приложения с подключенным отладчиком нажмите клавишу **F11** (**Отладка > Шаг с заходом**). F11 — это команда **Шаг с заходом**, которая выполняет приложение с переходом к следующему оператору. При запуске приложения с помощью клавиши F11 отладчик останавливается на первом выполняемом операторе.

Желтая стрелка представляет оператор, на котором приостановлен отладчик. В этой же точке приостанавливается выполнение приложения (этот оператор пока не выполнен).

Клавишу F11 удобно использовать для более детальной проверки потока выполнения.(Мы также покажем другие варианты более быстрого перемещения по коду.) По умолчанию отладчик пропускает непользовательский код

Быстрое выполнение до точки в коде с помощью мыши

В отладчике наведите указатель мыши на строку кода, чтобы слева появилась **кнопка выполнения до щелкнутого** (Выполнить до этого места) .



Note

Кнопка **выполнения до щелкнутого** (Выполнить до этого места) доступна начиная с версии Visual Studio 2017.

Нажмите кнопку **выполнения до щелкнутого** (Выполнить до этого места). Отладчик продолжает выполнение до строки кода, которую вы щелкнули.

Использование этой кнопки аналогично установке временной точки останова. Кроме того, эта команда удобна для быстрой работы в видимой области кода приложения.**Выполнение до щелкнутого** можно использовать в любом открытом файле.

Вывод отладчика из текущей функции

В некоторых случаях может потребоваться продолжить сеанс отладки, однако полностью проведя отладчик сквозь текущую функцию.

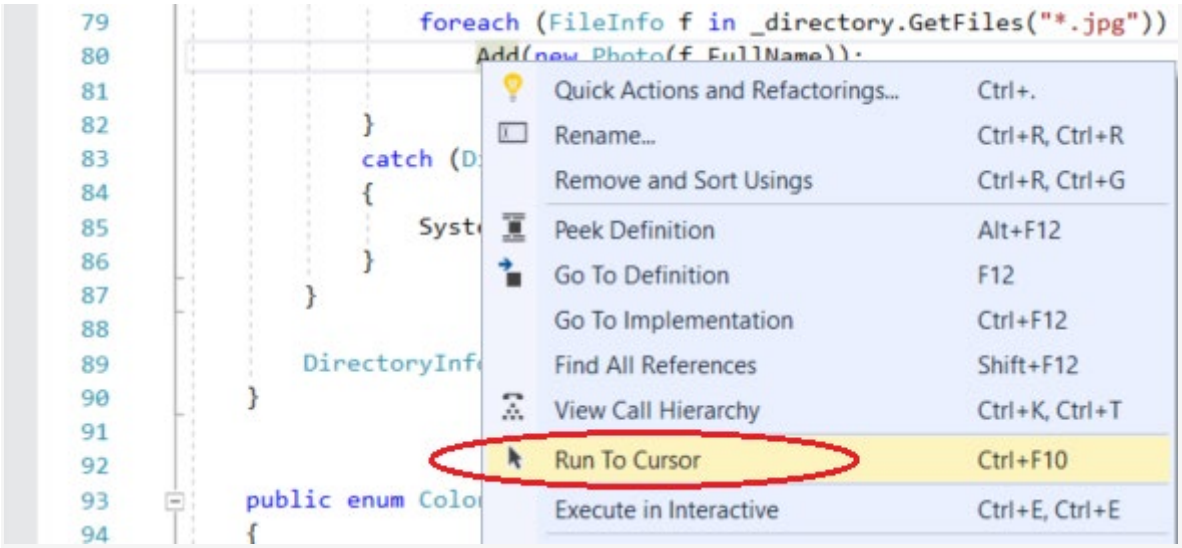
Нажмите клавиши **SHIFT+F11** (или выберите **Отладка > Шаг с выходом**).

Эта команда возобновляет выполнение приложения (и перемещает отладчик) до возврата текущей функции.

Выполнить до текущей позиции

Остановите отладчик, нажав красную кнопку **Остановить отладку** ■ или клавиши **SHIFT +F5**.

Щелкните правой кнопкой мыши строку кода в приложении и выберите команду **Выполнить до текущей позиции**. Эта команда запускает отладку и задает временную точку останова на текущей строке кода.



Если имеются заданные точки останова, отладчик приостанавливается в первой достигнутой точке останова.

Нажимайте клавишу **F5**, пока не достигнете строки кода, для которой выбрали **Выполнить до текущей позиции**.

Эта команда удобна, когда вы редактируете код и хотите быстро задать временную точку останова и одновременно запустить отладчик.

Note

Вы можете использовать функцию **Выполнить до текущей позиции** в окне **Стек вызовов** во время отладки.

Быстрый перезапуск приложения

Нажмите кнопку **Перезапустить** на панели инструментов отладки (**CTRL+SHIFT+F5**).

Кнопка **Перезапустить** позволяет сэкономить время, затрачиваемое на остановку приложения и перезапуск отладчика. Отладчик приостанавливается в первой точке останова, достигнутой при выполнении кода.

Если вы хотите остановить отладчик и вернуться в редактор кода, можно нажать красную кнопку останова вместо кнопки **Перезапустить**.

Проверка переменных с помощью подсказок по данным

Теперь, когда вы немного освоились, у вас есть хорошая возможность проверить состояние приложения (переменные) с помощью отладчика. Функции, позволяющие проверять переменные, являются одними из самых полезных в отладчике. Реализовывать эту задачу можно разными способами. Часто при попытке выполнить отладку проблемы пользователь старается выяснить, хранятся ли в переменных значения, которые требуются в определенном состоянии приложения.

Приостановив работу в отладчике, наведите указатель мыши на объект, чтобы просмотреть его значение свойства по умолчанию (в этом примере имя файла market 031.jpg является значением свойства по умолчанию).

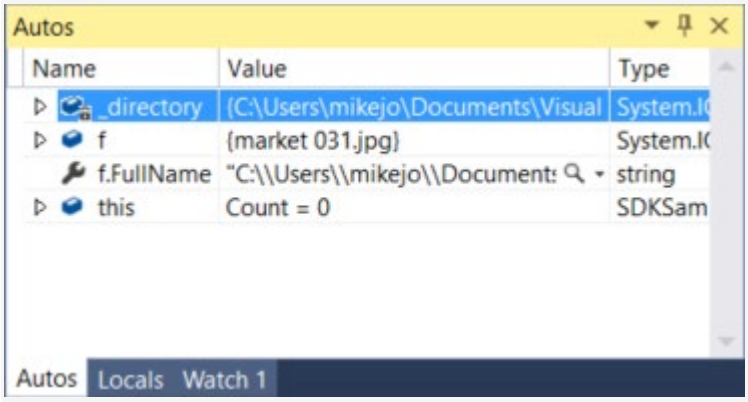
```
74          private void Update()
75          {
76              this.Clear();
77              try
78              {
79                  foreach (FileInfo f in _directory.GetFiles("*.jp
80                      Add(new Photo(f.FullName));  ≤ 3ms elapsed
81
82              }
83              catch (DirectoryNotFoundException)
84              {
85                  System.Windows.MessageBox.Show("No Such Directory
86              }
87          }
88
```

Разверните объект для просмотра его свойств (таких как свойство FullPath в этом примере).

Часто при отладке бывает необходимо быстро проверить значения свойств для объектов. Лучше всего для этого подходят подсказки по данным.

Проверка переменных с помощью окон "Видимые" и "Локальные"

Во время отладки взгляните на окно **Видимые** в нижней части редактора кода.

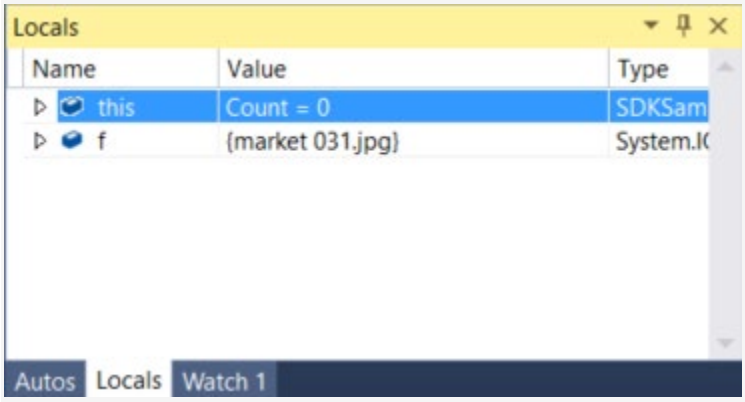


В окне **Видимые** отображаются переменные вместе с текущим значением и типом. Окно**Видимые** содержит все переменные, используемые в текущей строке или предыдущей строке (в C++ в окне отображаются переменные в трех предыдущих строках кода;сведения о зависящем от языка поведении см. в соответствующей документации).

Note

В JavaScript окно **Локальные** поддерживается, а окно **Видимые** — нет.

Взгляните в окно **Локальные**. В окне **Локальные** показаны переменные, которые находятся в текущей области.



В этом примере объекты this и f находятся в области действия. Дополнительные сведения см. в статье Проверка переменных в окнах "Видимые" и "Локальные".

Установка контрольного значения

В окне **Контрольное значение** можно указать переменную (или выражение), которую необходимо отслеживать.

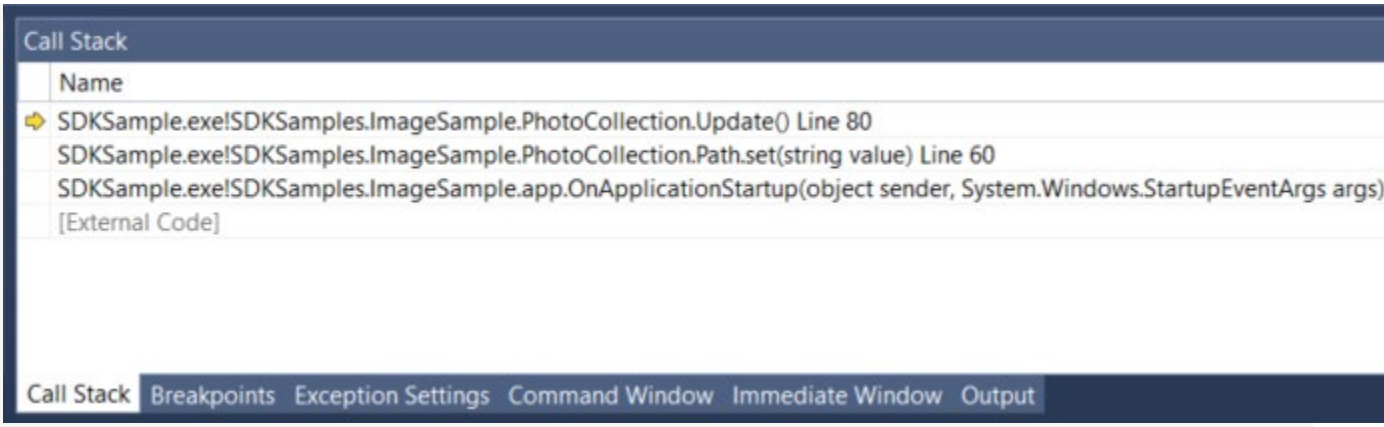
Во время отладки щелкните правой объект кнопкой мыши и выберите пункт **Добавить контрольное значение**.

В этом примере у вас есть контрольное значение, заданное для объекта f, и по мере перемещения по отладчику вы можете наблюдать за изменением его значения. В отличие от других окон переменных, в окне **Контрольное значение** всегда отображаются просматриваемые вами переменные (они выделяются серым цветом, когда находятся вне области действия).

Дополнительные сведения см. в статье Установка контрольных значений с помощью окон "Контрольное значение" и "Быстрая проверка".

Просмотр стека вызовов

Во время отладки щелкните окно **Стек вызовов**, которое по умолчанию открыто в нижней правой области.



В окне **Стек вызовов** показан порядок вызова методов и функций. В верхней строке приведена текущая функция (в данном примере метод Update). Во второй строке показано, что функция Update была вызвана из свойства Path.set и т. д. Стек вызовов хорошо подходит для изучения и анализа потока выполнения приложения.

Note

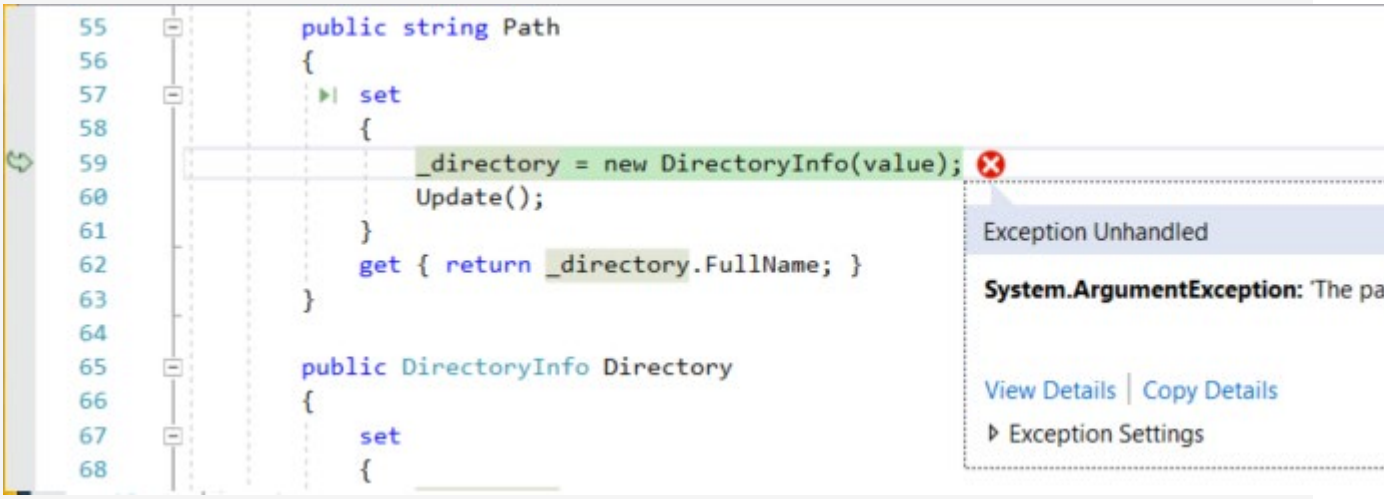
Окно **Стек вызовов** аналогично перспективе "Отладка" в некоторых интегрированных средах разработки, например Eclipse.

Дважды щелкните строку кода, чтобы просмотреть исходный код. При этом также изменится текущая область, проверяемая отладчиком. Это не перемещает отладчик.

Для выполнения других задач можно воспользоваться контекстными меню из окна **Стек вызовов**. Например, можно вставлять точки останова в указанные функции, перезапускать приложение с помощью функции **Выполнение до текущей позиции** и изучать исходный код. См. практическое руководство по просмотреть стек вызовов.

Изучение исключения

Когда приложение выдает исключение, отладчик переходит к строке кода, вызвавшей исключение.



В этом примере **помощник по исправлению ошибок** показывает исключениеSystem.Argument и сообщение об ошибке, где сказано, что путь имеет

недопустимую форму. Таким образом, мы знаем, что ошибка произошла в аргументе метода или функции.

В этом примере вызов DirectoryInfo выдал ошибку на пустой строке, хранящейся в переменной value.

Помощник по исправлению ошибок — это отличная функция, которая помогает отлаживать ошибки. Используя помощник по исправлению ошибок, вы также можете, например, просмотреть сведения об ошибке и добавить контрольное значение. При необходимости вы также можете изменить условия для возникновения конкретного исключения. См. дополнительные сведения об обработке исключений в коде в разделеПриемы и инструменты отладки.

Note

Помощник по исправлению ошибок заменил помощник по исключениям, который запускается в Visual Studio 2017.

Разверните узел **Параметры исключений**, чтобы просмотреть дополнительные параметры для обработки исключения этого типа, однако в рамках этого тура ничего менять не требуется.

NetBeans IDE

NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада[4] и ряда других.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведётся независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

Последние версии NetBeans IDE поддерживают рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода.

Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun JDK или J2EE SDK подходящей версии. Среда разработки NetBeans по-умолчанию поддерживала разработку для платформ J2SE и J2EE. Начиная с версии 6.0 NetBeans поддерживает разработку для мобильных платформ J2ME, C++ (только g++) и PHP без установки дополнительных компонентов.

В сентябре 2016 года Oracle передала интегрированную среду разработки NetBeans в руки фонда Apache[5].

Разработка среды NetBeans началась в 1996 году под названием Xelfi (игра букв на основе Delphi)[12][13], в качестве проекта студентов по созданию Java IDE под руководством Факультета Математики и Физики Карлова Университета в Праге. В 1997 году Роман Станек сформировал компанию вокруг проекта и стал выпускать коммерческие версии среды NetBeans до передачи всех прав на IDE корпорации Sun Microsystems в 1999 году. Sun открыла исходные коды среды разработки NetBeans IDE в июне следующего года. Сообщество NetBeans с тех пор постоянно развивается и растёт благодаря людям и компаниям, использующим и поддерживающим проект

Пользователи могут выбирать, какую сборку NetBeans IDE они хотят загрузить для работы с кодом.

NetBeans IDE Bundle for Web & Java EE[18] Сборка предоставляет инструменты для всех последних стандартов Java EE 6, включая новые: Java EE 6 Web Profile, Enterprise Java Beans (бины), сервлеты, Java Persistence API, веб-сервисы и аннотации. NetBeans также поддерживает JSF 2.0 (Facelets), JavaServer Pages (JSP), Hibernate, Spring, и Struts-фреймворки. А также веб-серверы, такие, как GlassFish и Apache Tomcat.

NetBeans IDE Bundle for PHP. Начиная с версии 6.5 Netbeans поддерживает PHP. Сборка для PHP включает:

подсветка синтаксиса, автозавершение кода, подсветка вхождений и ошибок.

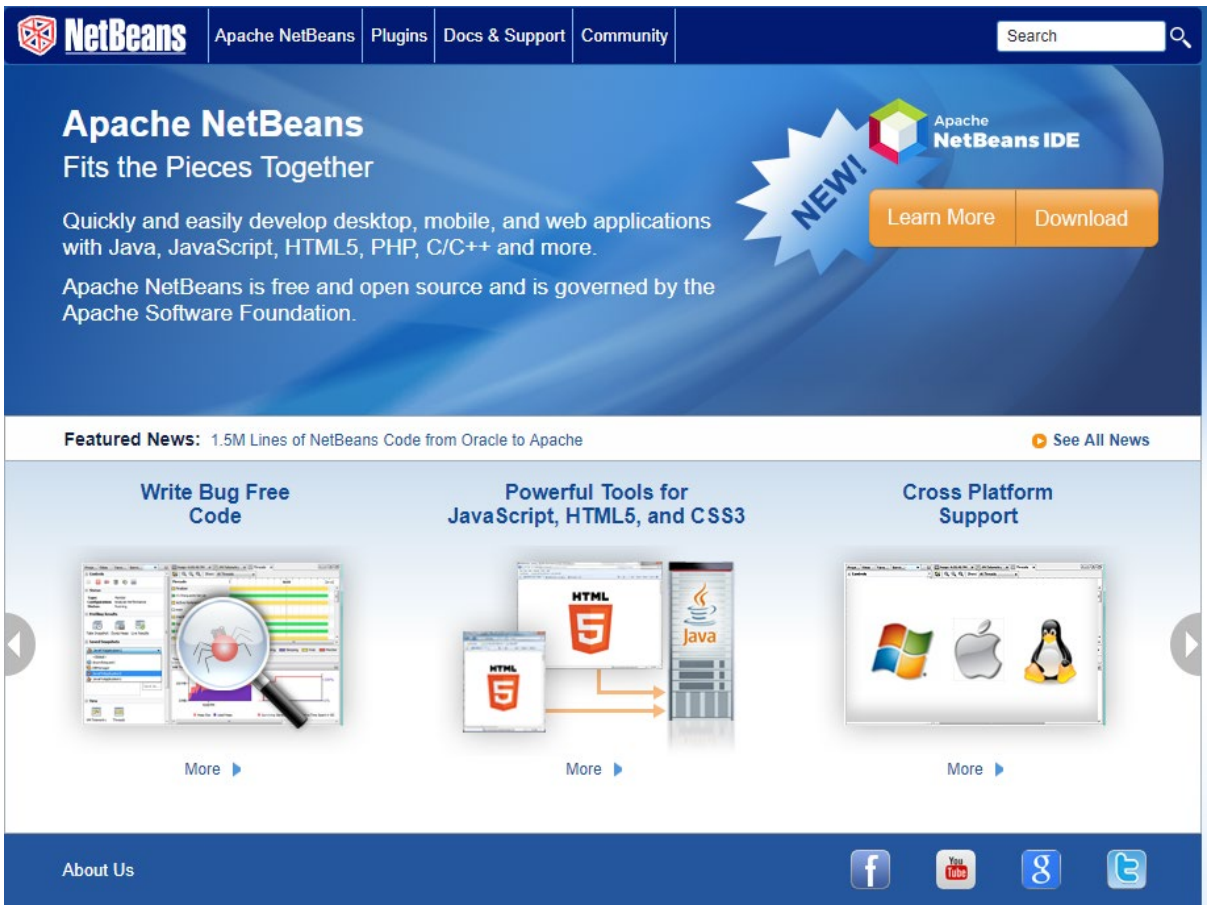
отладка кода xdebug

поддержка тестирования с PHPUnit (англ.)русск. и Selenium (англ.)русск.

поддержка PHP фреймворков Symfony (с версии 6.8) и Zend Framework (с версии 6.9)

поддержка PHP 5.3 (с версии 6.8)

поддержка Git начиная (с версии 7.1)


NetBeans
Apache NetBeans
Plugins
Docs & Support
Community
Search
Apache NetBeans
Fits the Pieces Together
Quickly and easily develop desktop, mobile, and web applications with Java, JavaScript, HTML5, PHP, C/C++ and more.
Apache NetBeans is free and open source and is governed by the Apache Software Foundation.
NEW!
Apache NetBeans IDE
Learn More
Download
Featured News: 1.5M Lines of NetBeans Code from Oracle to Apache
See All News
Write Bug Free Code
Powerful Tools for JavaScript, HTML5, and CSS3
Cross Platform Support
HTML
Java
More
More
More
About Us

Just released!

Apache NetBeans 11.1

Find out more

Apache NetBeans

Fits the Pieces Together

Development Environment, Tooling Platform and Application Framework.

Fast & Smart Editing

Apache NetBeans is much more than a text editor. It highlights source code **syntactically and semantically**, lets you easily **refactor code**, with a range of handy and powerful tools.

Java, JavaScript, PHP, HTML5, CSS, and More

Apache NetBeans provides editors, wizards, and templates to help you create applications in **Java**, **PHP** and many other languages.

Cross Platform

Apache NetBeans can be **installed** on all operating systems that support Java, i.e, Windows, Linux, Mac OSX and BSD. Write Once, Run Anywhere, applies to NetBeans too.

Join us

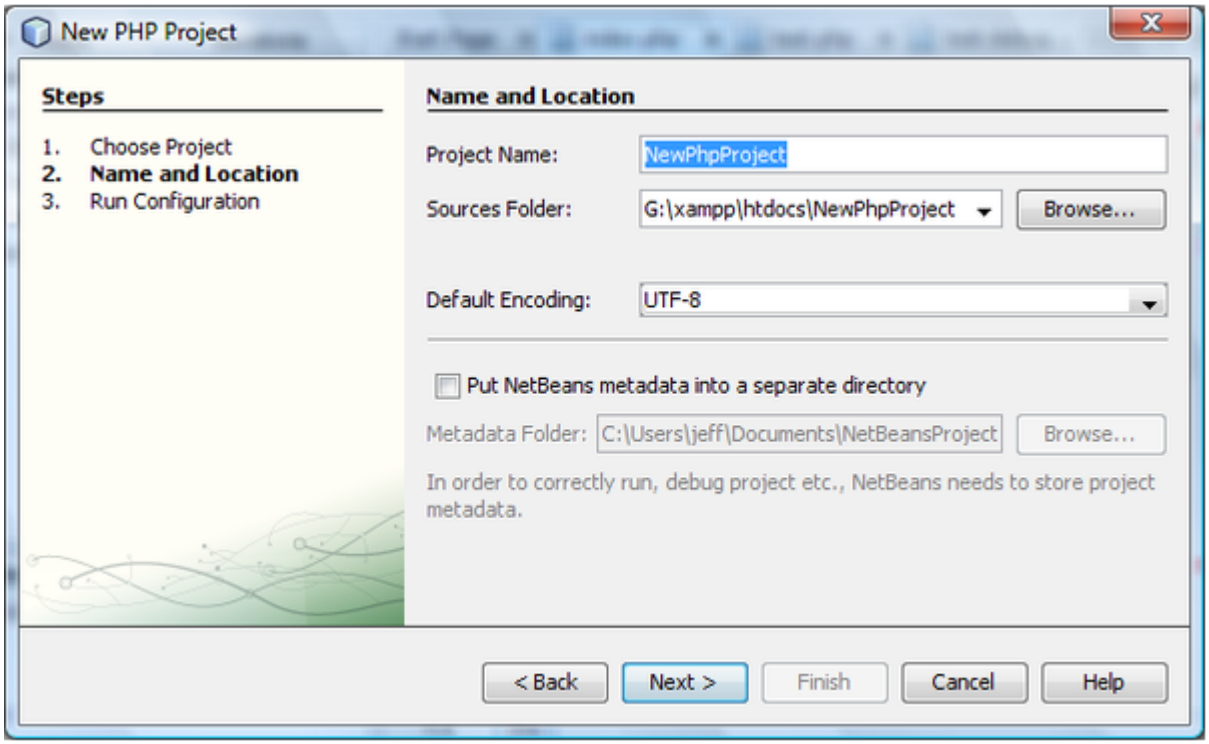
Subscribe to our **mailing lists**, or follow us in **Twitter**, **Slack**, **FaceBook** or **YouTube**.

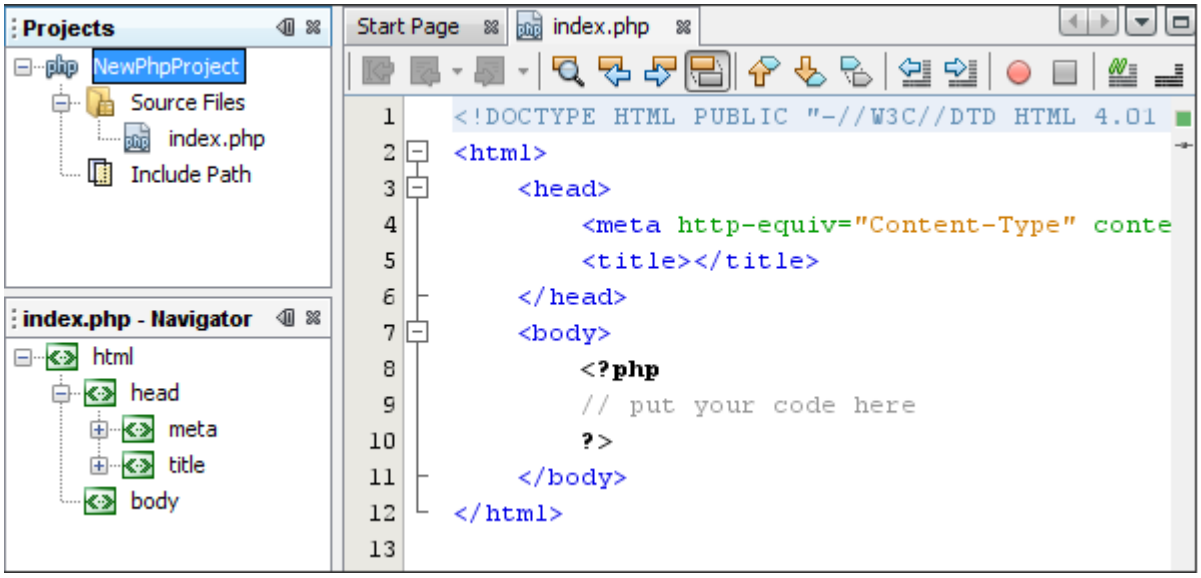
Participate

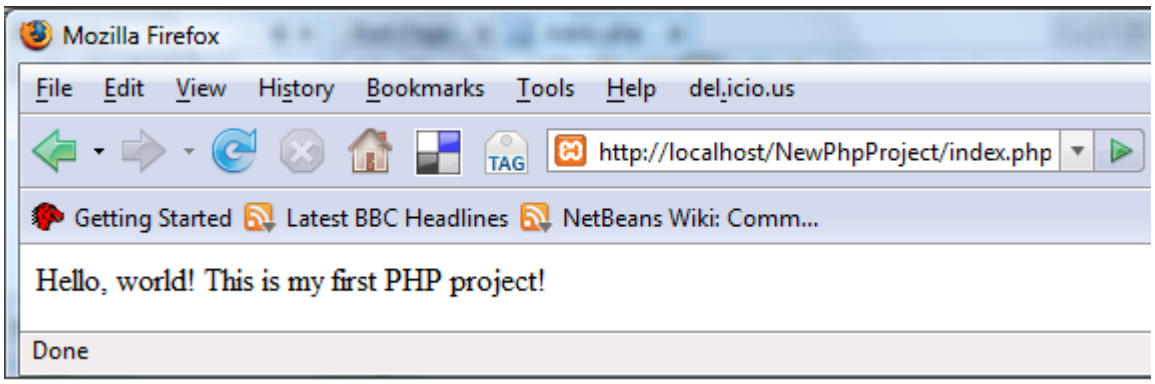
See how you can participate by **submitting pull requests**, or **filing issues**.

Learn

We are currently reviewing the tutorials. See how you can **help us** review the **Java** and **PHP** tutorials.


New PHP Project
Steps
1. Choose Project
2. Name and Location
3. Run Configuration
Name and Location
Project Name:
NewPhpProject
Sources Folder:
G:\xampp\htdocs\NewPhpProject
Browse...
Default Encoding:
UTF-8
Put NetBeans metadata into a separate directory
Metadata Folder:
C:\Users\jeff\Documents\NetBeansProject
Browse...
In order to correctly run, debug project etc., NetBeans needs to store project metadata.
< Back
Next >
Finish
Cancel
Help


Projects
NewPhpProject
Source Files
index.php
Include Path
index.php - Navigator
html
head
meta
title
body
Start Page
index.php
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
<html>
<head>
<meta http-equiv="Content-Type" conte
<title></title>
</head>
<body>
<?php
// put your code here
?>
</body>
</html>

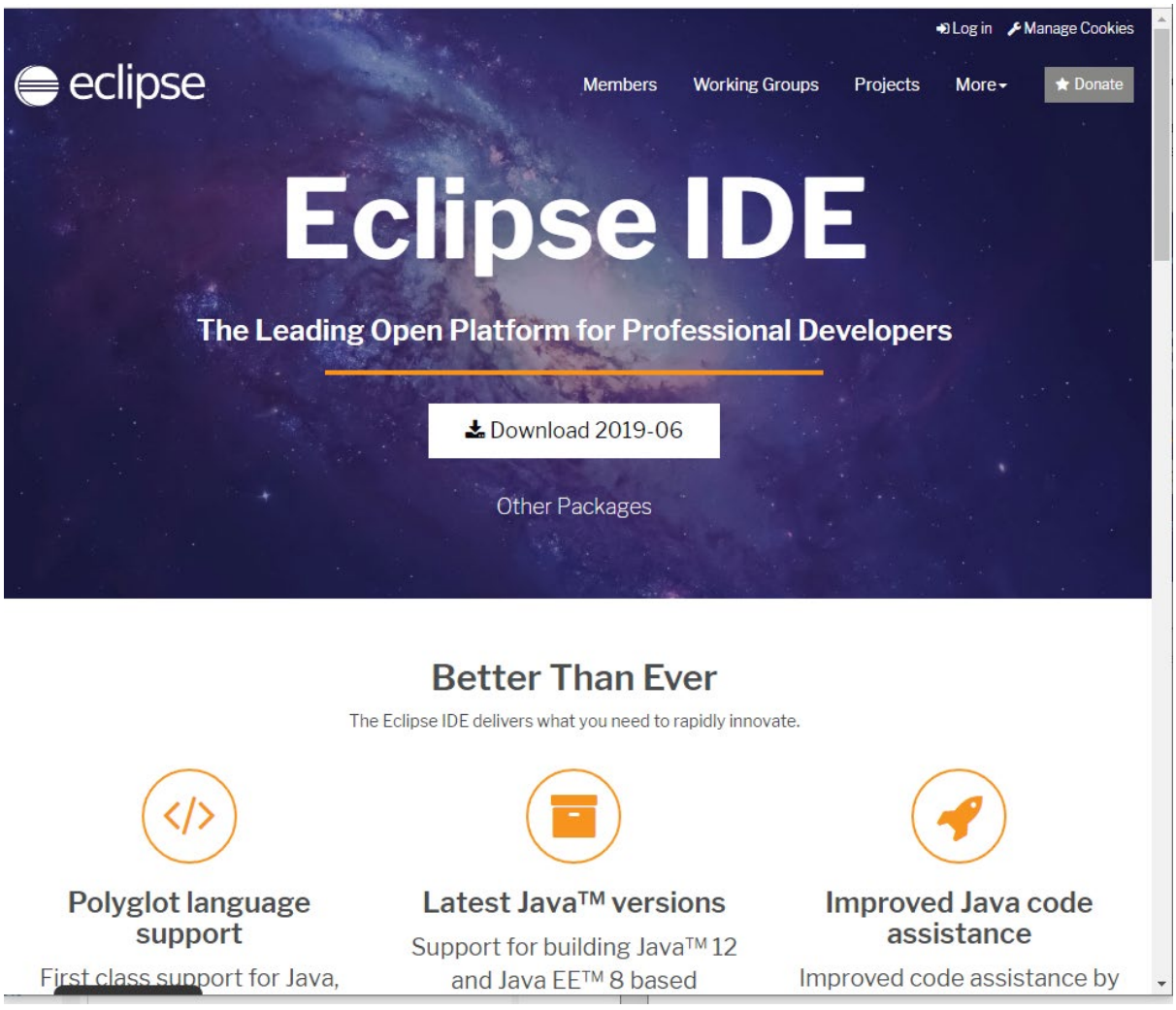

Mozilla Firefox
File Edit View History Bookmarks Tools Help del.icio.us
http://localhost/NewPhpProject/index.php
Getting Started
Latest BBC Headlines
NetBeans Wiki: Comm...
Hello, world! This is my first PHP project!
Done

Eclipse

Eclipse ([ɪˈklɪps], с англ. — «затмение»[2]) — свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation.

Наиболее известные приложения на основе Eclipse Platform — различные «Eclipse IDE» для разработки ПО на множестве языков (например, наиболее популярный «Java IDE», поддерживавшийся изначально, не полагается на какие-либо закрытые расширения, использует стандартный открытый API для доступа к Eclipse Platform)

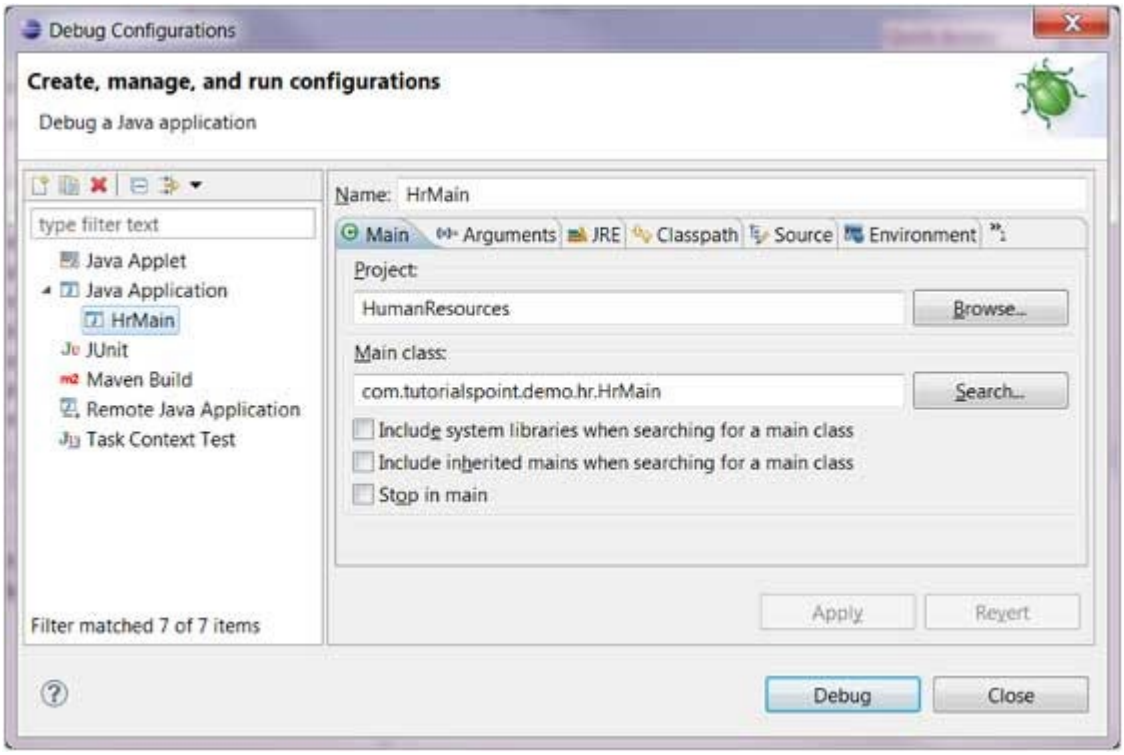
Первоначально Eclipse разрабатывалась фирмой IBM как преемник среды разработки IBM VisualAge, в качестве корпоративного стандарта IDE для разработки на разных языках под платформы IBM. По сведениям IBM, проектирование и разработка стоили 40 миллионов долларов.[3] Исходный код был полностью открыт и сделан доступным после того, как Eclipse был передан для дальнейшего развития независимому от IBM сообществу.



Eclipse написана на Java, потому является платформо-независимым продуктом, за исключением библиотеки SWT, которая разрабатывается для всех распространённых платформ (см. ниже). Библиотека SWT используется вместо стандартной для Java библиотеки Swing. Она полностью опирается на нижележащую платформу (операционную систему), что обеспечивает быстроту и натуральный внешний вид пользовательского интерфейса, но иногда вызывает на разных платформах проблемы совместимости и устойчивости приложений.

Для среды Eclipse существует целый ряд свободных и коммерческих модулей. Первоначально среда была разработана для языка Java, но в настоящее время существуют многочисленные расширения для поддержки и других языков:

Язык	**Модуль**
C/C++	CDT[38]
Fortran	Photran[39]
Perl	EPIC[40]
PHP	PDT[41]
JavaScript	JSEclipse[42]
Python	PyDev (Eclipse) (англ.)[43]
Ruby	RDT[44]
1C V8	1C:Enterprise DT[45]



Отладка и тестирование WEB – приложений

Ручное тестирование

Ручное тестирование **пользовательского интерфейса** проводится тестировщиком-оператором, который руководствуется в своей работе описанием тестовых примеров в виде набора *сценариев*. Каждый *сценарий* включает в себя перечисление последовательности действий, которые должен выполнить оператор и описание важных для анализа результатов тестирования ответных реакций системы, отражаемых в пользовательском интерфейсе. Типичная форма записи *сценария* для проведения *ручного тестирования* – таблица, в которой в одной колонке описаны действия (шаги *сценария*), в другой – ожидаемая реакция системы, а третья предназначена для записи того, совпала ли ожидаемая реакция системы с реальной и перечисления несовпадений.

В табл. 19.1 приведен пример *сценария* для *ручного тестирования пользовательского интерфейса* [7].

аблица 19.1. Пример *сценария* для *ручного тестирования пользовательского интерфейса*

№ п/п	Действие	Реакция системы	Результат
	Делкните на значке "Система" и выберите пункт меню "Администрирование системы".	оявится окно ввода логина и пароля	**ерно**
	ведите в появившееся окно ввода имя пользователя "admin" и пароль "admin". Затем нажмите кнопку "OK".	оявится окно "Администрирование системы". В верхнем правом углу должно быть выведено имя вошедшего пользователя admin.	**еверно** Окно имеет название "Управление системой"

Ручное тестирование пользовательского интерфейса удобно тем, что контроль корректности интерфейса проводится человеком, т.е. основным "потребителем" данной части программной системы. К тому же при чисто косметических изменениях в интерфейсах системы, не отраженных в требованиях (например, при перемещении кнопок управления на 10 пикселей влево) анализ успешности прохождения теста будет выполняться не по формальным признакам, а согласно человеческому восприятию.

При этом *ручное тестирование* имеет и существенный недостаток – для его проведения требуются значительные человеческие и временные ресурсы. Особенно сильно этот недостаток проявляется при проведении *регрессионного тестирования* и вообще любого повторного тестирования – на каждой итерации повторного *тестирования пользовательского интерфейса* требуется участие тестировщика-оператора. В связи с этим в последнее десятилетие получили распространение средства автоматизации *тестирования пользовательского интерфейса*, снижающие нагрузку на тестировщика-оператора.

Существуют две взаимодополняющие технологии *отладки*.

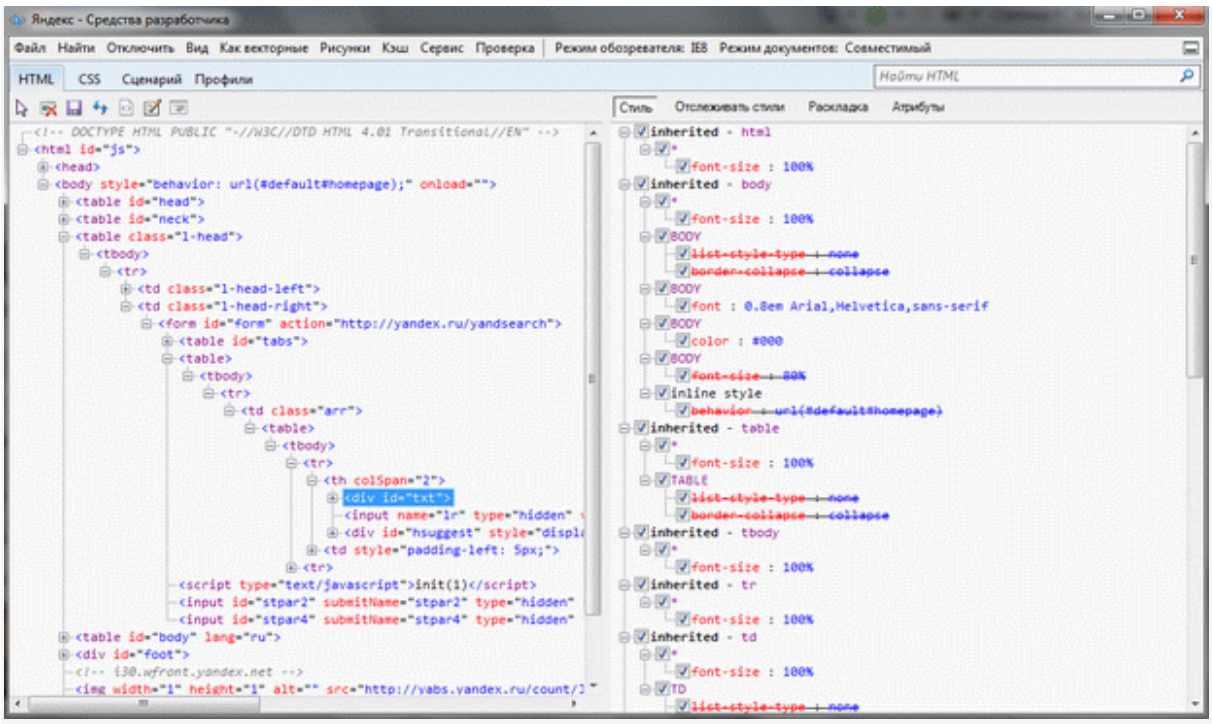
- Использование отладчиков – программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.
- Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода – на экран, принтер, громкоговоритель или в файл. Вывод отладочных сведений в файл называется журналированием.

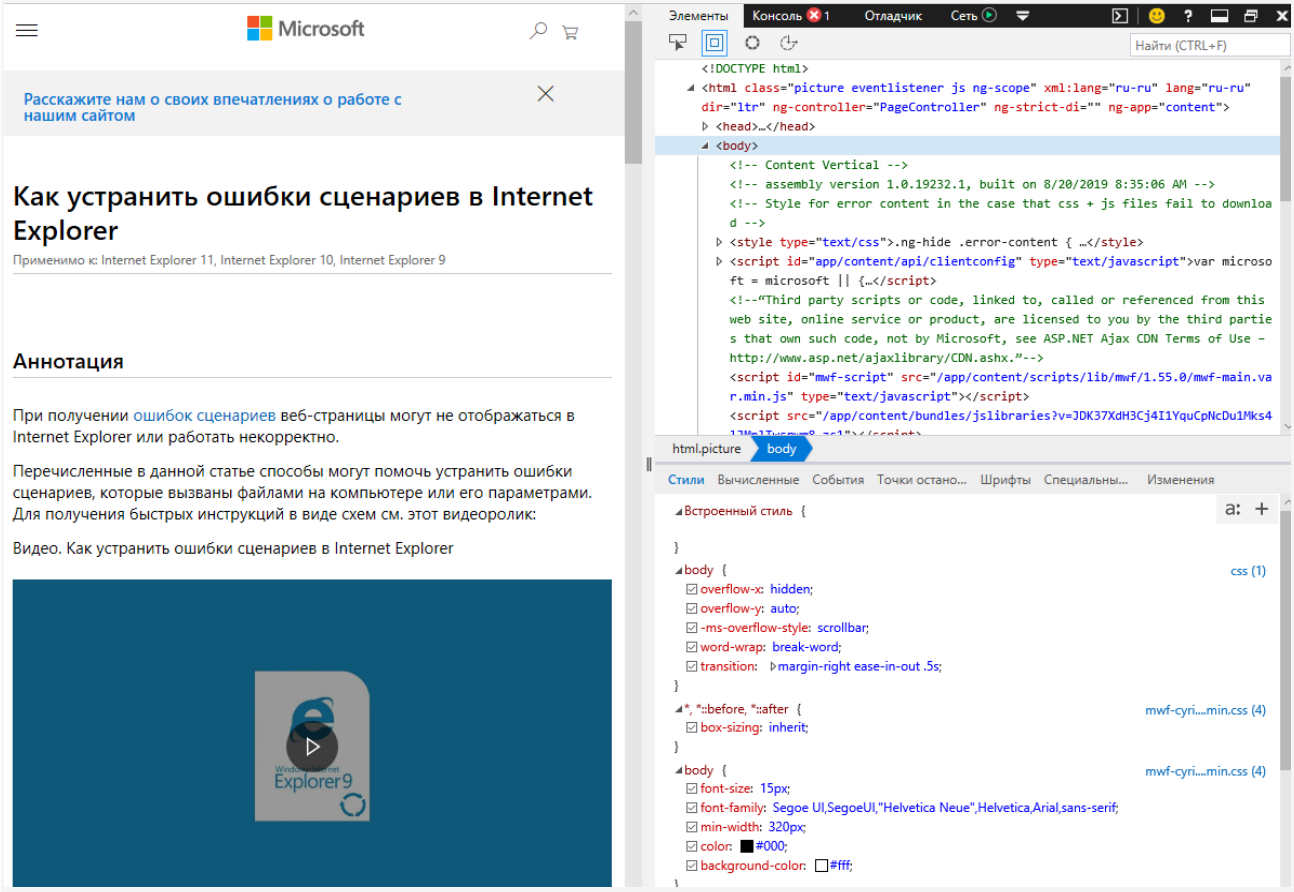
Отладка и профилирование в Internet Explorer 8

Рассмотрим, какие же возможности предоставляет IE8 -IE11, и сравним их с реализацией в других популярных веб-браузерах [29].

Инструменты для разработчика – Developer Tools – требуются для быстрой проверки кода страницы на его валидность (соответствие веб-стандартам, отсутствие ошибок в синтаксисе), а также для отладки различных сценариев (в основном, из-за распространения AJAX в современном сайтостроительстве это отладка JavaScript) и отображения стилей (CSS, отдельные цвета, шрифты и графика). При этом важно, чтобы данные инструменты были встроены в сам браузер, а не представляли собой отдельное приложение. В IE длительное время не было подобного инструмента, пока в 2007 году не появился отдельный тулбар для разработчиков (IE Developer Toolbar). С его помощью можно было разбирать код страницы (DOM), "на лету" проверять корректность HTML, CSS, RSS, включать отображение линеек масштаба для последующей верстки страниц и многое другое. Кроме этого тулбара, под IE были выпущены еще ряд инструментов – инспектор кода IEWatch, отладчики DebugBar и Fiddler, средства для работы со скриптовыми технологиями Ajaxview и Visual Web Developer Express. Однако в большинстве своем это были решения от сторонних разработчиков браузера, а фирменная панель IE Developer Toolbar иногда замедляла быстродействие браузера.

Запуск встроенного в IE8 инструмента веб-разработки осуществляется или через меню (Сервис – Средства разработчика), или с помощью горячей клавиши F12. В отдельном окне откроется двухпанельный редактор. В левой части будет отображаться DOM-структура сайта (секции head, body), а справа – расширенное содержание выбранного фрагмента кода. Developer Tools поддерживает работу по инспекции и отладкеHTML, CSS, JavaScript, а также просмотр cookies и кэша браузера (

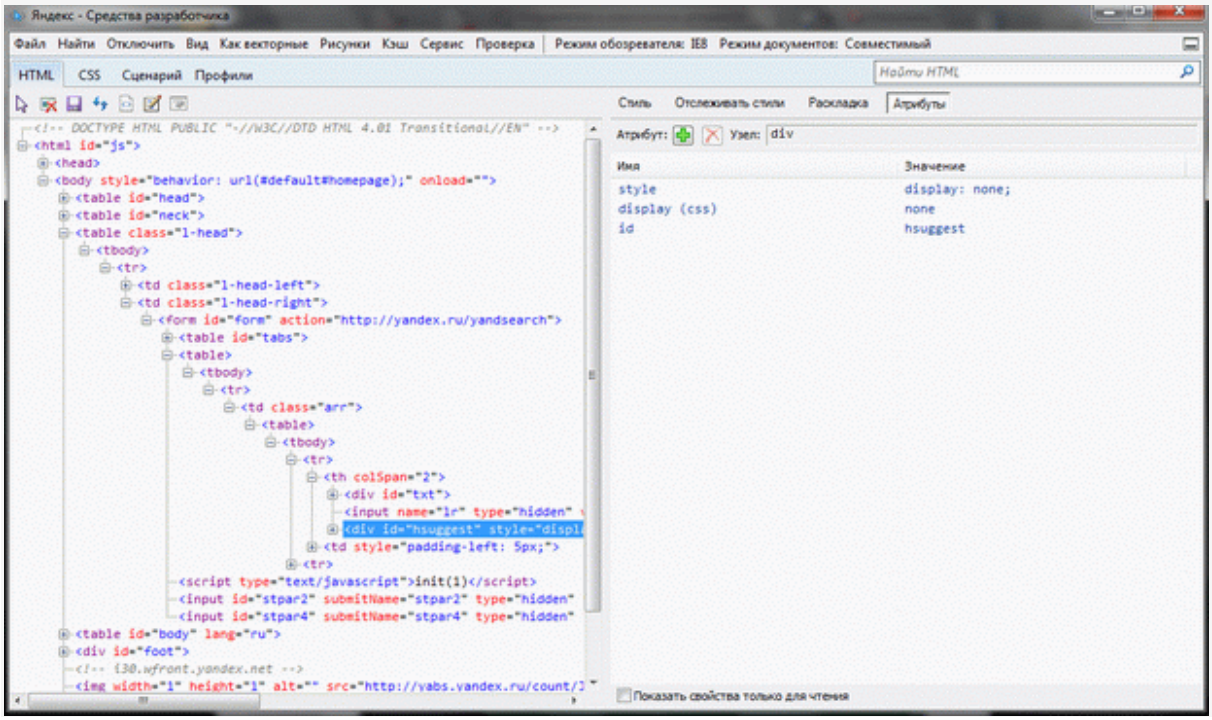




С помощью средств разработки можно проверять код HTML и каскадные таблицы стилей сайта в том виде, в котором они обрабатываются браузером Internet Explorer, а не в исходном виде. Это особенно удобно в случае с динамическими сайтами, сложными сайтами и сайтами, где применяются платформы ASP или PHP.

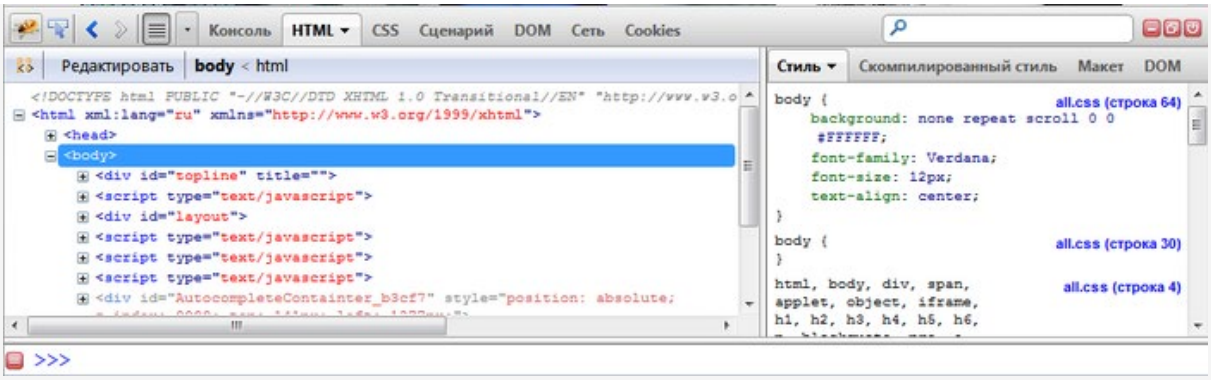
В основной области содержимого используется дерево документо-объектной модели (DOM) сайта, соответствующее структуре, содержащейся в памяти браузера Internet Explorer для отображения сайта. Для перемещения по дереву можно использовать мышь или клавиатуру. Самый быстрый способ найти узел, соответствующий определенному элементу страницы, – использовать функцию "Выбрать элемент щелчком". Используйте эту функцию, чтобы выбрать элемент на странице, и в средствах разработки будет автоматически выбран нужный узел дерева. Также можно использовать поле поиска.

После загрузки DOM-структуры сайта пользователь увидит древовидный список структуры тегов на левой панели. Справа же отображаются связанные с ними, соответственно, стили и атрибуты CSS. Например, при выборе в странице тега можно увидеть связанные с таблицей классы CSS и отследить вызываемые параметры цветов, шрифтов, полей, отступов и так далее. Здесь же в разделе "Раскладка" (Layout) будет показано расположение этого элемента на площади страницы с указанием размеров в пикселях и пропорций. Естественно, его можно изменять непосредственно на этой вкладке. В атрибутах отображается список всех зависимостей HTML-кода на странице; к ним можно добавлять собственные из выпадающего списка



Mozilla Firefox

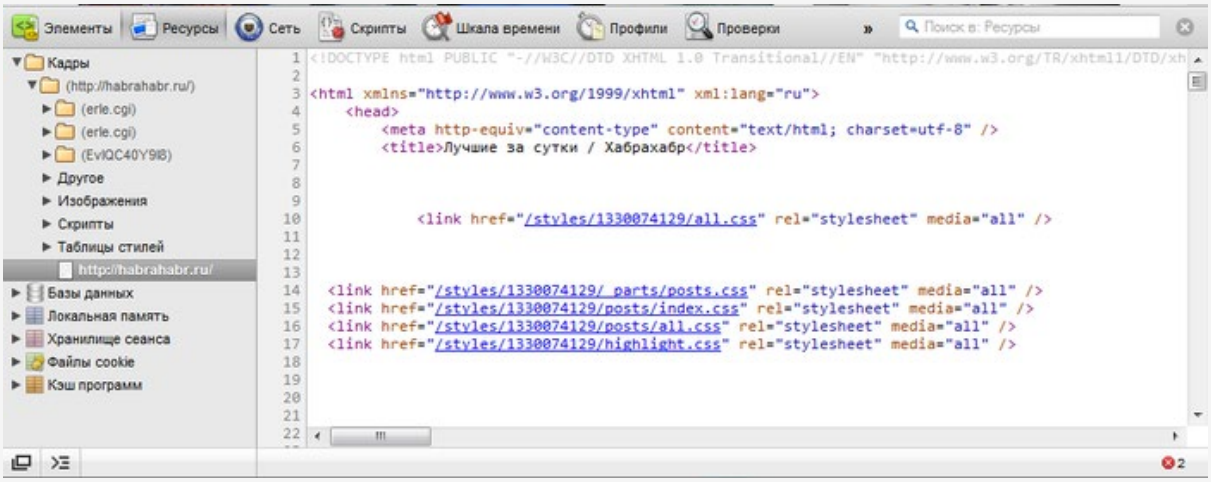
Наверно, именно Firefox можно назвать пионером отладки клиентского кода. Долгое время его указывали как браузер, наиболее подходящий для разработки, а все благодаря расширению Firebug, которое содержит, наверно, все нужные возможности, кроме валидации HTML кода.



Так же, начиная с версии 4, появилась встроенная Веб-консоль, которая реализует часть функций вкладки «Консоль» и «Сеть» Firebug'a, а так же некоторые возможности по отладке CSS.

Google Chrome и Safari

Эти браузеры, основанные на WebKit, обладают встроенным инструментом разработки Web Inspector, который очень хорошо развит и реализует практически те же функции, что и Firebug. При этом, надо отдать ему должное, он не замедляет работы браузера, что водится за «старшим братом».



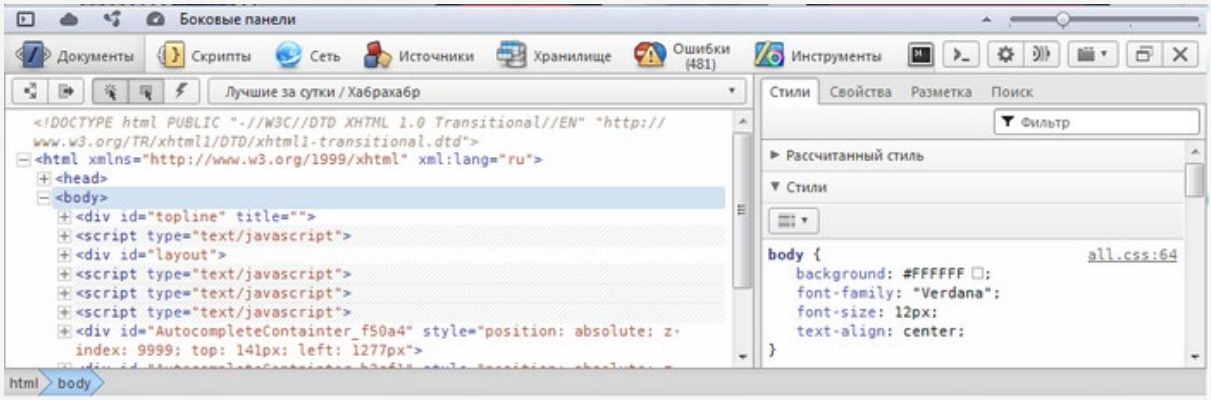
В Chrome он может быть вызван по нажатию клавиш Ctrl+Shift+I или просто по F12. В Safari он хорошо спрятан, и для его использования нужно включить возможности разработки в настройках браузера. Позже инструменты разработчика станут доступными из пункта «Разработка» главного меню или по сочетанию клавиш Ctrl+Alt+I.

Для валидации HTML кода так же нужно устанавливать сторонние расширения. К примеру, для Chrome, это может быть Validity. Для Safari пока не удалось подобрать ничего подходящего.

Opera

Opera так же имеет встроенный инструмент для разработчиков, который называется «Opera

Dragonfly», и может быть вызван в любой момент по сочетанию клавиш Ctrl+Shift+I. Он похож на то, что нам представляет WebKit, и имеет подобные возможности и плюсы, хотя, на мой лично взгляд, менее удобен.



В Опере есть простое средство отправить страницу на валидацию в validator.w3.org. В контекстном меню области отображения сайта есть пункт «Соблюдены ли веб-стандарты», который как раз и отвечает за это.

С расширениями для валидации HTML в каталоге Оперы напряженная ситуация. По этой причине у расширения Validator нет альтернатив.

Средства отладки. Программные, внутрисхемные симуляторы, отладочные платы

Программные симуляторы.

Назначение.

Программный симулятор пред-назначен для имитации на инструментальном (персональном) компьютере ра-боты контроллера при выполнении разработанной программы управления и представляет собой программно–логическую модель проектируемого устройства. Симуляция, или тестирование программы на компьютере, предоставляет разра-ботчику широкие возможности для контроля ее исполнения, выявления алгорит-мических неточностей и требуемых ресурсов проектируемой системы. Для ими-тации внешних условий и ситуаций используются файлы входных воздействий, которые задают последовательность сигналов, поступающих на моделируемое устройство. Входные сигналы имитируют воздействие на устройство и позволяют при выполнении программы контролировать его состояние в различные моменты времени.

На основе тестирования модели ведется разработка реального устройства уп-равления. Программные симуляторы используется на начальной стадии проекти-рования реального устройства. Полноэкранный графический интерфейс пользователя современных симуляторов позволяет:

● быстро проверить правильность алгоритма выполнения программы и отдель-ных операций;

● вести наблюдение за исполнением любого фрагмента программы и реакцией микроконтроллера на различные события;

● исследовать ситуации, трудно воспроизводимые на реальной аппаратуре. Особенность программных симуляторов: исполнение программ, загружен-ных в симулятор, происходит в масштабе времени, отличном от реального.

Основные функции. Основными функциями программных симуляторов являются:

● загрузка файлов программ управления в широко распространенных форматах,

● запуск отлаживаемой программы в пошаговом или непрерывном режимах, в прямом и обратном направлениях с заданием условных и безусловных конт-рольных точек останова. При отладке программы по шагам остановка в конт-рольной точке эквивалентна останову системы тактирования МК с прекраще-нием работы всех периферийных модулей. Дальнейший прогон программы возобновляется с того состояния периферийных модулей, в котором они были приостановлены. Именно такой режим работы наиболее удобен для на-блюдения за процессом выполнения контроллером управляющей программы;

● контроль и модификация состояния ресурсов симулируемого микроконтрол-лера, содержимого ячеек памяти и регистров; встроенных периферийных устройств (таймеров, портов, АЦП, систем прерываний и др.). В процессе вы-полнения программы на экране компьютера отображается текущее состояние модели;

● высокоуровневая символьная отладка программ при условии использования компиляторов, поставляющих необходимую отладочную информацию. Такую возможность предоставляют высокоуровневые симуляторы–отладчики (High–Level Debuggers);

● вывод статистической информации по результатам прогона отлаживаемой программы. Эта информация необходима для оптимизации структуры программы.

Дополнительные возможности. К ним следует отнести:

● использование специальных файлов входных воздействий, которые позволяют задать симулятору различные комбинации входных потоков данных и формы сигналов. Входные сигналы изменяют состояние устройства в различные мо-менты времени и позволяют контролировать выполнение операций в течение определенного числа машинных циклов. Для понимания работы устройства эти файлы обычно пишутся в текстовом формате, поэтому на их создание затрачивается много времени;

● интерфейс внешней среды, позволяющий пользователю создавать и гибко использовать модель внешней среды микроконтроллера, функционирующую и взаимодействующую с отлаживаемой программой по заданному алгоритму;

● встроенную интегрированную среду разработки, позволяющую не только уп-равлять процессом симуляции, но и поддерживать весь процесс отладки.

Основными характеристиками, определяющими качество и функциональ-ность симулятора, являются:

● перечень поддерживаемых микроконтроллеров и встроенных узлов периферии, а также компиляторов и форматов;

● скорость, детальность и точность симуляции.

К достоинствам программной симуляции следует отнести:

● возможность детального исследования и анализа алгоритма работы проекти-руемого устройства, многократного воспроизведения рабочих ситуаций бла-годаря вмешательству в процесс исполнения тестируемой программы;

● отсутствие в ограничении ресурсов МК при отладке программы управления;

● использование при симуляции исходного текста программы, а не файлов, сгенерированных компилятором (объектного кода) значительно упрощает контроль выполнения программы;

● сравнительно низкая стоимость;

● расширение воспроизводимости путем использования файлов входных воз-действий.

Недостатки программной симуляции:

● невозможность использования реальных источников входных сигналов и фор-мирования реальных выходных сигналов управления. Разработка файла вход-ных воздействий может потребовать много времени и больших затрат;

● использование виртуального микроконтроллера не позволяет получить пол-ностью достоверных результатов и требует дополнительной отладки реально-го устройства.

Итог

Таким образом, программный симулятор можно использовать как виртуальный микроконтроллер, алгоритмы работы которого полностью совпадают с алгоритмами работы реального контроллера, при этом: имеются широкие дополнительные возможности по вмешательству в процесс исполнения тестируемой программы управления; отсутствуют ограничения в использовании ресурсов, связанных с обслужива-нием процесса отладки.

Этап моделирования на программном симуляторе проводится на самой ран-ней стадии работы над проектом, когда аппаратная часть находится еще в стадии проектирования.

Схемные эмуляторы

Назначение.

Схемный (внутрисхемный) эмулятор (ln–Circuit Emulator — ICE) представляет собой программно–аппаратный комплекс, предназначенный для отладки программного обеспечения и аппаратной части разрабатываемого устройства в реальном времени. При эмуляции связь с инст-рументальным компьютером и управление режимами отладки возложены на за-мещающий контроллер, входящий в состав схемного эмулятора, благодаря чему:

● при отладке разрабатываемой программы отсутствуют ограничения в исполь-зовании внутренних ресурсов моделируемого микроконтроллера;

● функционирование отлаживаемой системы становится понятной, контролиру-емой, произвольно управляемой и модифицируемой по требованиям разработчика. Обеспечивается визуальное наблюдение за работой системы на эк-ране дисплея и изменение ее режимов путем подачи соответствующих управ-ляющих сигналов и модификации содержимого регистров и памяти;

● можно получить параметры входных и выходных сигналов, совпадающие с аналогичными параметрами целевого МК на плате конечного изделия. Схемные эмуляторы используются на этапе комплексной отладки всей системы:

● для проведения отладки в непрерывном режиме и между точками останова;

● для выполнения отлаживаемой программы управления с использованием замещающего микроконтроллера, полностью идентичного (или близкого по параметрам) к целевому микроконтроллеру;

● для получения параметров входных и выходных сигналов МК отладочной пла-ты, совпадающих с аналогичными параметрами целевого МК на плате конеч-ного изделия. Функциональные возможности.

Схемный эмулятор позволяет:

● загружать отлаживаемую программу в память системы и выполнять ее в по-шаговом или непрерывном режиме, останавливать программу при выполне-нии заданных условий;

● выводить на монитор состояния и содержимое всех регистров и памяти и при необходимости модифицировать их;

● вести символьную отладку, т. е. одновременно контролировать ход выполнения программы и наблюдать соответствие между исходным текстом программы и состоянием всех ресурсов эмулируемого микроконтроллера. Символьная отладка позволяет оперировать с более приемлемыми для человека символь-ными именами, не требуя запоминания машинных кодов команд и адресов;

● выполнять трассировку, т. е. вести для последующего анализа запись внеш-них сигналов и (или) состояния внутренних регистров микроконтроллера;
● получать по результатам прогона отлаживаемой программы статистическую информацию (например, количество обращений к различным участкам программы и время, затраченное на их выполнение). Наличие в программной оболочке эмулятора встроенного редактора, встроенного менеджера проектов и системы управления облегчает отладку, избавляет разработчика от множества рутинных действий.

При этом предоставляется возможность управлять не только процессом эмуляции, но и поддерживать в целом процесс отладки, начиная от написания исходного текста программы до ее компиляции и отладки, обеспечивается простое и быстрое взаимодействие с другими инструментальными средствами (программным отладчиком–симулятором и программатором).
Схемные эмуляторы с набором перечисленных выше функций называют отладочными комплексами, или системами развития (Development System). Наряду с такими сложными и дорогими моделя-ми выпускаются упрощенные варианты схемных эмуляторов, реализованные на одной печатной плате. Они позволяют выполнять отладку систем малой и сред-ней сложности, имеют на порядок более низкую стоимость, поэтому находят до-статочно широкое практическое применение.

Интегрированные среды разработки

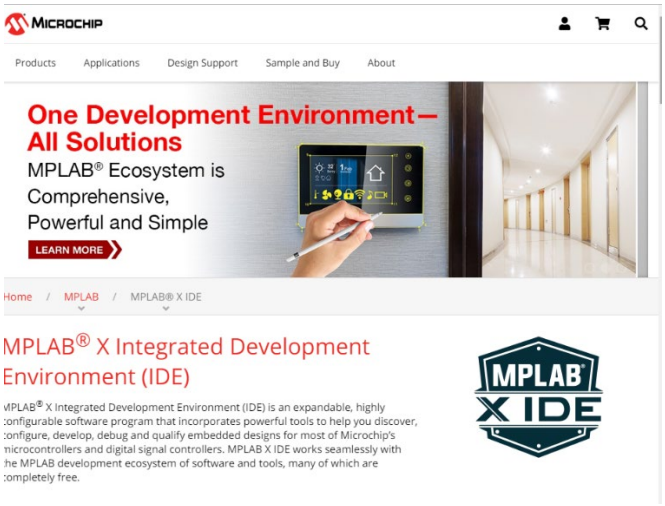
MPLAB

MPLAB — интегрированная среда разработки, представляющая собой набор программных продуктов, предназначенная для облегчения процесса создания, редактирования и отладки программ для микроконтроллеров семейства PIC, производимых компанией Microchip Technology. Среда разработки состоит из отдельных приложений, связанных друг с другом, и включает в себя компилятор с языка ассемблер, текстовый редактор, программный симулятор и средства работы над проектами, также среда позволяет использовать компилятор с языка C.

MPLAB X IDE

MPLAB X — это версия среды разработки MPLAB, написанная на базе платформы NetBeans. MPLAB X отличается расширенной функциональностью по сравнению с MPLAB IDE v8.X за счёт дополнительных функций платформы NetBeans, таких, как встроенная система управления версиями и поддержка плагинов, написанных сторонними разработчиками. Также данная версия, помимо Windows, поддерживает операционные системы Mac OS X и

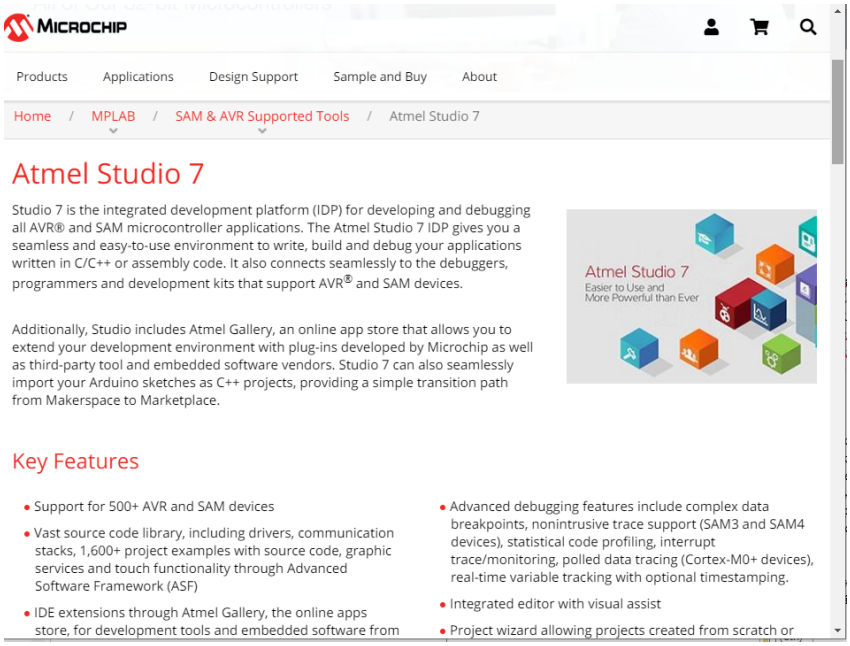
Linux. Среда также поддерживает компилятор SDCC с открытым исходным кодом, доступный для трёх основных семейств операционных систем: Mac OS, Windows и Linux.



Atmel Studio (ранее AVR Studio) — основанная на Visual Studio бесплатная проприетарная интегрированная среда разработки (IDE) для разработки приложений для 8- и 32-битных микроконтроллеров семейства AVR и 32-битных микроконтроллеров семейства ARM от компании Atmel, работающая в операционных системах Windows NT/2000/XP/Vista/7/8/10. Atmel Studio содержит компилятор GNU C/C++ и эмулятор, позволяющий отладить выполнение программы без загрузки в микроконтроллер.

Ранее среда разработки носила название AVR Studio, но начиная с версии 6.0, вышедшей в 2012 году, в неё была добавлена поддержка разработки для микроконтроллеров архитектуры ARM, также выпускаемых фирмой Atmel, и среда разработки получила новое название Atmel Studio. Текущая версия (Atmel Studio 7) поддерживает все выпускаемые на сегодняшний день фирмой Atmel микроконтроллеры архитектур AVR, AVR32 и ARM и средства разработки.

Atmel Studio содержит в себе менеджер проектов, редактор исходного кода, инструменты виртуальной симуляции и внутрисхемной отладки, позволяет писать программы на ассемблере или на C/C++.



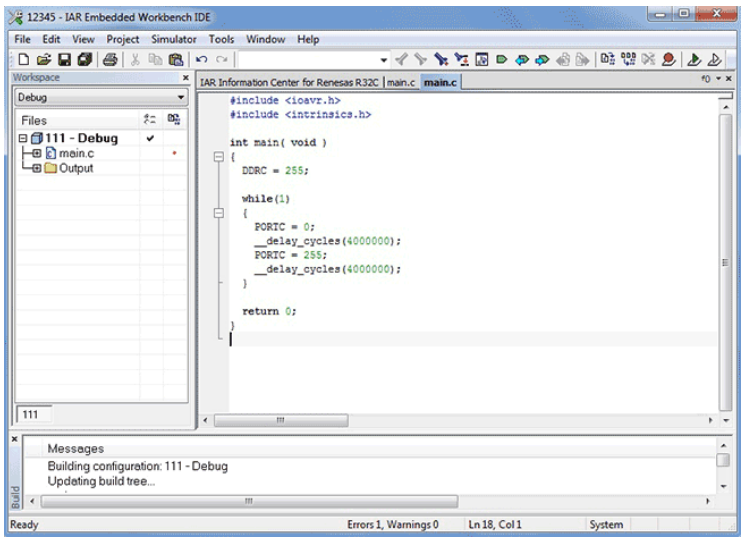
Code Composer Studio

Code Composer Studio — интегрированная среда разработки для создания кода для DSP и/или ARM процессоров семейства TMS320, и других процессоров, таких как MSP430, выпускаемых Texas Instruments. Code Composer Studio включает операционную систему реального времени DSP/BIOS, позднее (начиная с релиза 6.3) получившую название SYS/BIOS. Также в состав продукта входят симуляторы и поддержка JTAG-ориентированной отладки.

Текущая версия 5.3 основана на стоковой (без модификаций) версии Eclipse IDE проекта Helios[1]. Использование стоковой версии позволяет команде разработчиков IDE обновлять версию Eclipse без скрупулёзной подводки всех своих изменений под новую версию.

IAR Embedded Workbench

Многофункциональная среда разработки приложений на языках C, C++ и ассемблере для целого ряда микроконтроллеров от различных производителей.

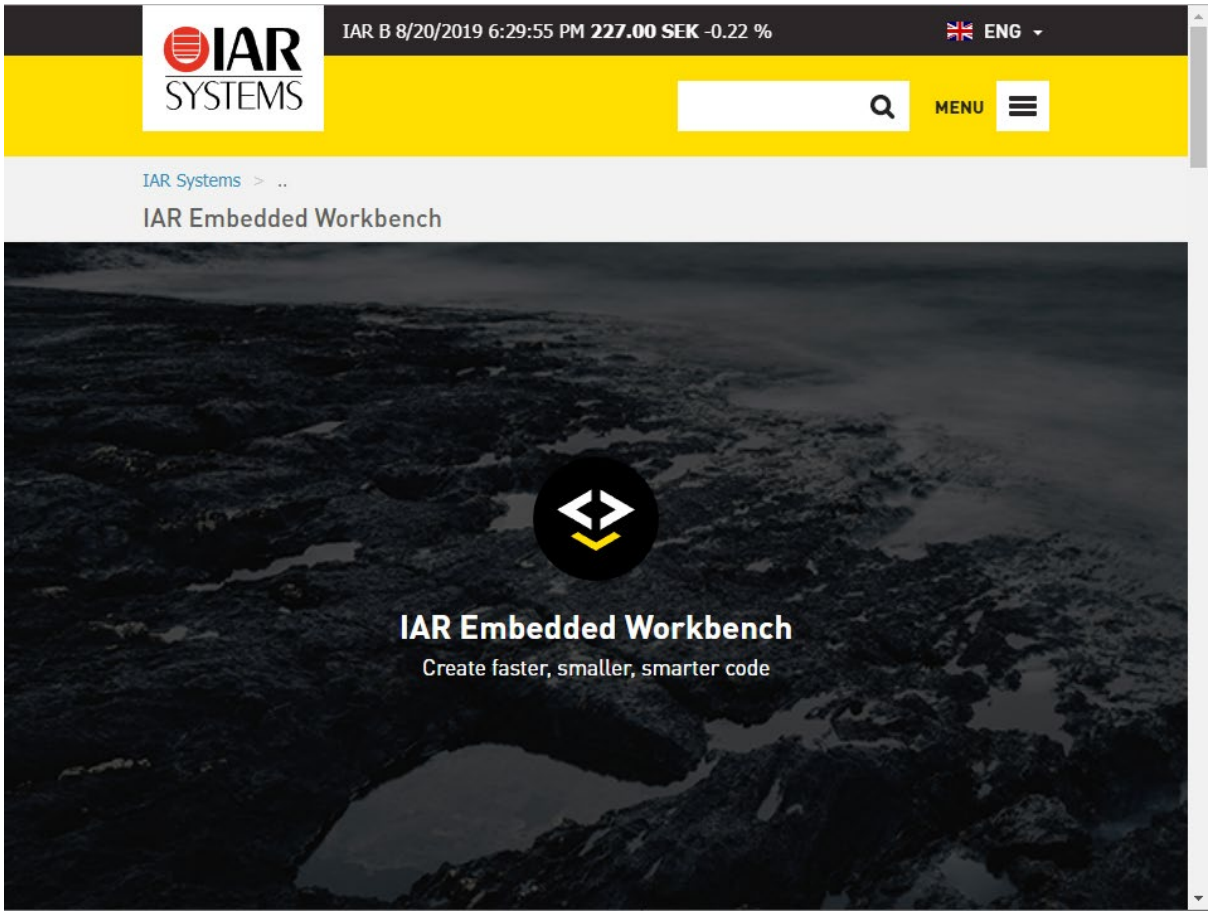


Рабочее окно программы IAR

Основные преимущества пакета – дружественный пользовательский интерфейс и непревзойденная оптимизация генерируемого кода. Кроме этого реализована поддержка различных операционных систем реального времени и JTAG -адаптеров сторонних компаний.

В настоящее время IAR Embedded Workbench поддерживает работу с 8-, 16-, 32-разрядными микроконтроллерами от Atmel, ARM, NEC, Infineon, Analog Devices, Cypress, Microchip Technologies, Micronas, Dallas Semiconductor/Maxim, Ember, Luminary, NXP, OKI, Samsung, National Semiconductor, Texas Instruments, STMicroelectronics, Freescale, TI/Chipcon, Silicon Labs и Renesas. Для каждой платформы существует своя среда разработки, в частности ARM микроконтроллерам соответствует версия пакета IAR Embedded Workbench for ARM.

IAR Embedded Workbench была разработана IAR Systems, более двадцати лет являющейся одной из ведущих компанией по созданию C/C++ компиляторов для встраиваемых микроконтроллерных устройств и систем. Ее штаб-квартира находится в старинном городе Уппсала (Швеция). В настоящее время программы IAR Systems используют по всему миру более сотни тысяч разработчиков, производителей телекоммуникационного и промышленного оборудования, медицинской и компьютерной техники, среди которых такие гиганты, как Apple Computer, Cisco Systems, Motorola, Hewlett-Packard и Siemens.



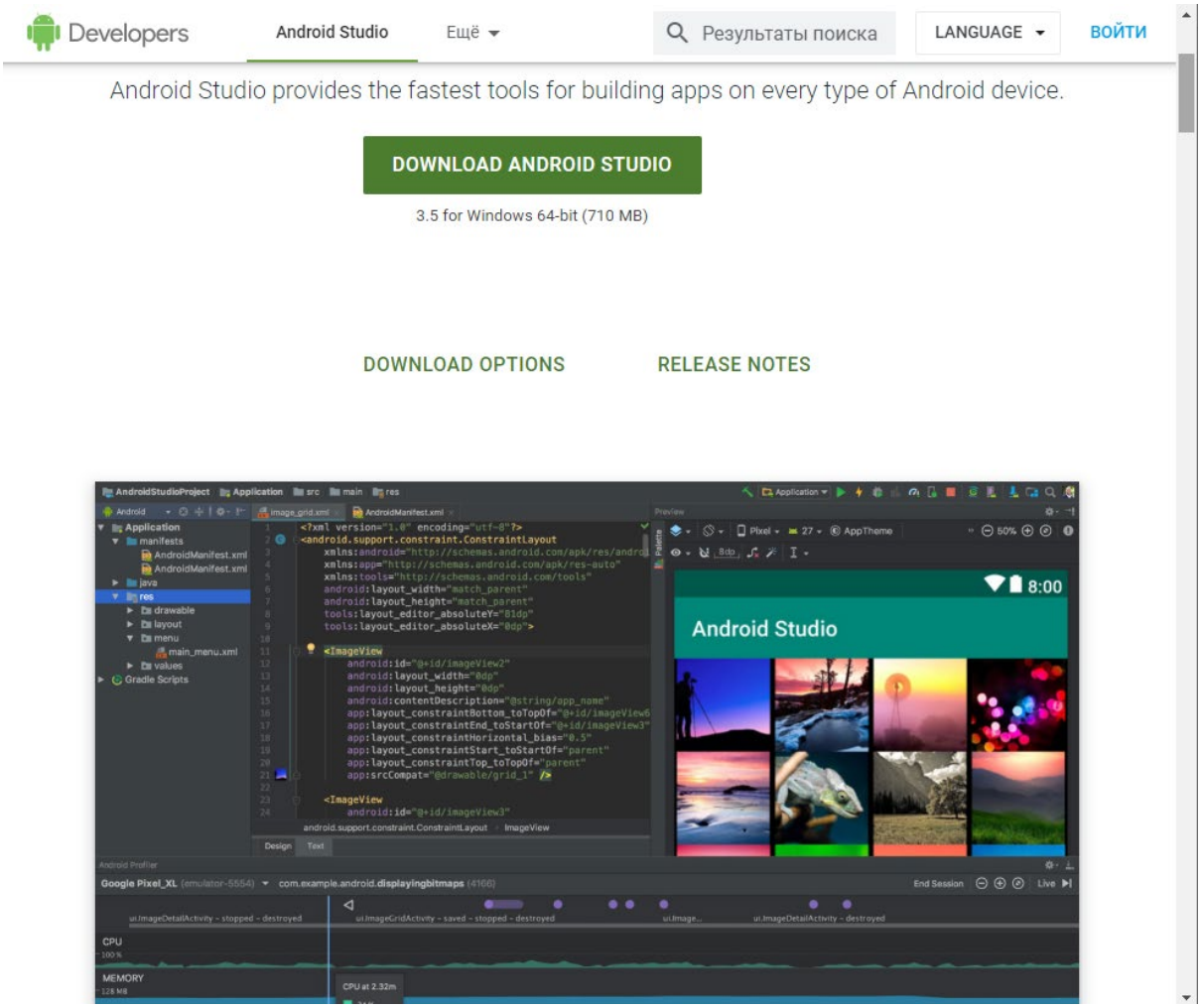
IAR Embedded Workbench является коммерческим продуктом, его стоимость составляет около 3000 долларов за одну пользовательскую лицензию. Однако, в качестве дополнения к полнофункциональной версии, существует бесплатный вариант среды программирования с единственным ограничением на размер выходного кода до 4 или 8 КБ в зависимости от модели контроллера. Этот вариант подойдет для первого знакомства с программой, а также написания небольших приложений. Можно найти и взломанную версию, но для ее нормальной работы придется отключать выход в интернет.

Android Studio — это интегрированная среда разработки (IDE) для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O.

IDE находилась в свободном доступе начиная с версии 0.1, опубликованной в мае 2013, а затем перешла в стадию бета-тестирования, начиная с версии 0.8, которая была выпущена в июне 2014 года. Первая стабильная версия 1.0 была выпущена в декабре 2014 года, тогда же прекратилась поддержка плагина Android Development Tools (ADT) для Eclipse.

Android Studio — среда разработки под популярную операционную систему Андроид. Программное обеспечение вышло в 2013 году и развивается по сегодня. В каждой новой версии Android Studio разработчик добавляет увеличивает функционал, оптимизирует процессы и другое.

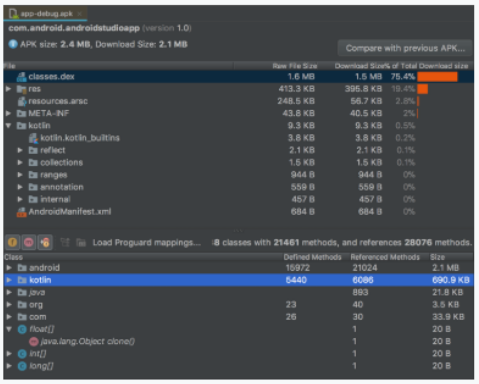
В комплекте с IDE идет эмулятор, проверяющий корректную работу уже написанных утилит, приложений на разных конфигурациях.



FEATURE

APK Analyzer

Find opportunities to reduce your Android app size by inspecting the contents of your app APK file, even if it wasn't built with Android Studio. Inspect the manifest file, resources, and DEX files. Compare two APKs to see how your app size changed between app versions.

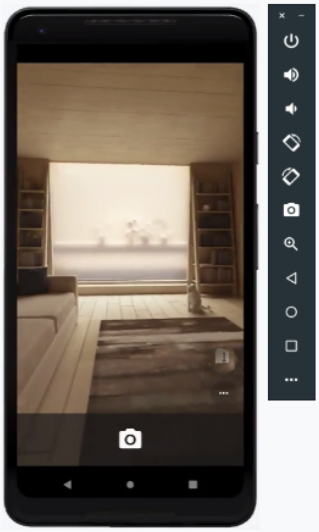


MORE ABOUT THE APK ANALYZER

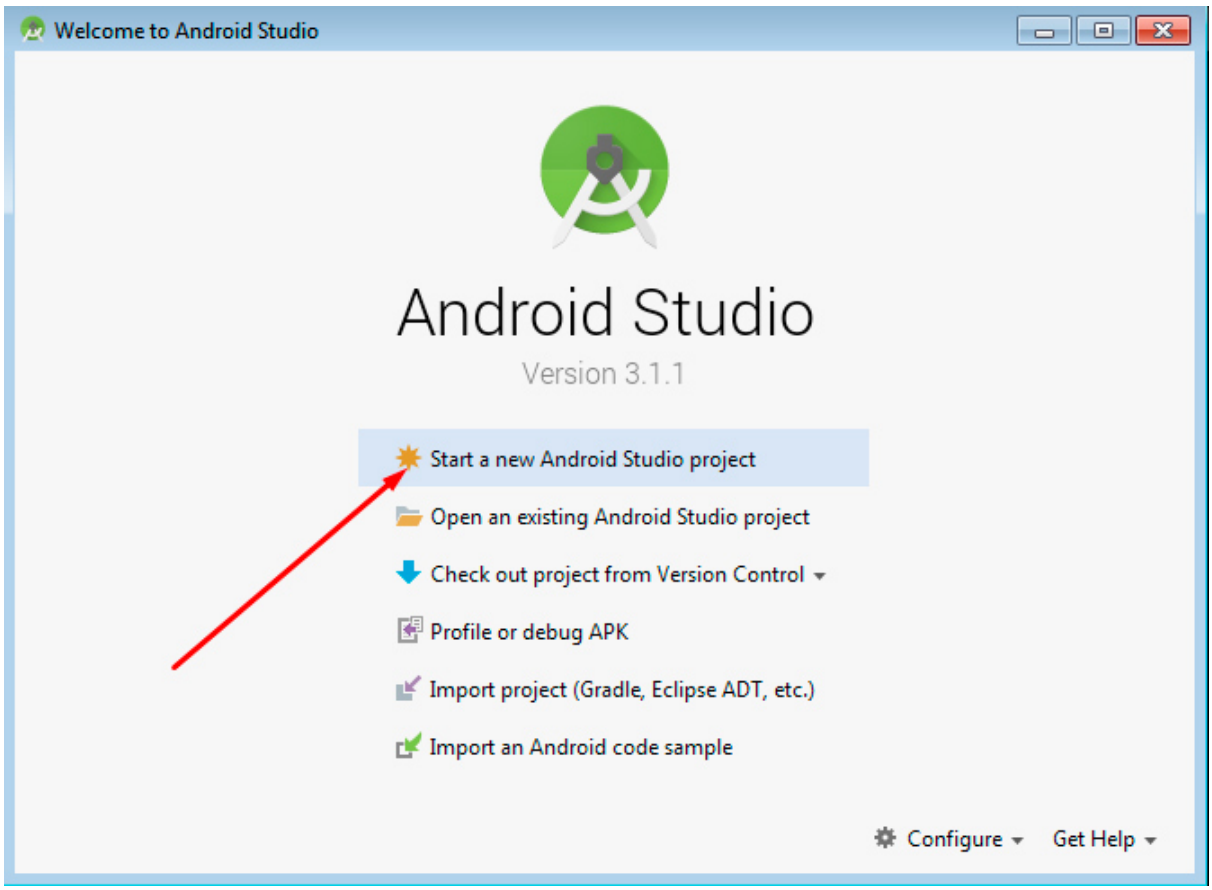
FEATURE

Fast emulator

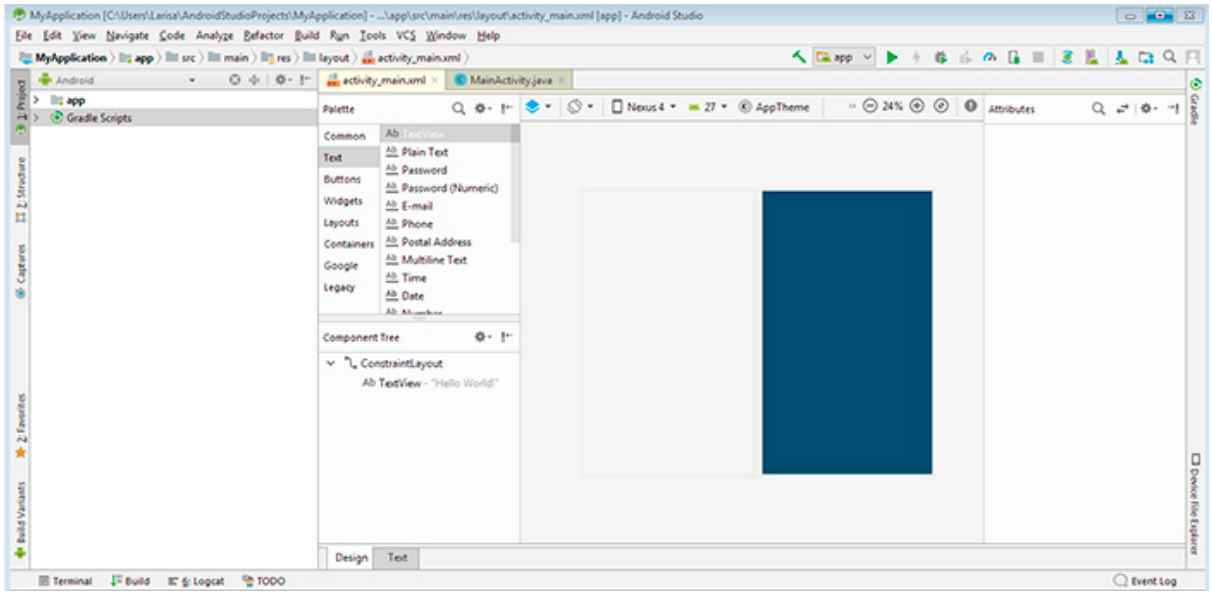
Install and run your apps faster than with a physical device and simulate different configurations and features, including ARCore, Google's platform for building augmented reality experiences.



MORE ABOUT THE EMULATOR



IDE Android Studio никак внешне не отличить от других именитых аналогов. Рабочая область очень схожа с программой Visual от Microsoftt. Среда разработки выполнена в классическом виде. Программистам, не имевшим ранее дело с Android, необходимо привыкнуть к тому, как создаются приложения под нее. Все игры и утилиты — это набор идеально синергирующих между собой компонентов, а не цельный программный код.



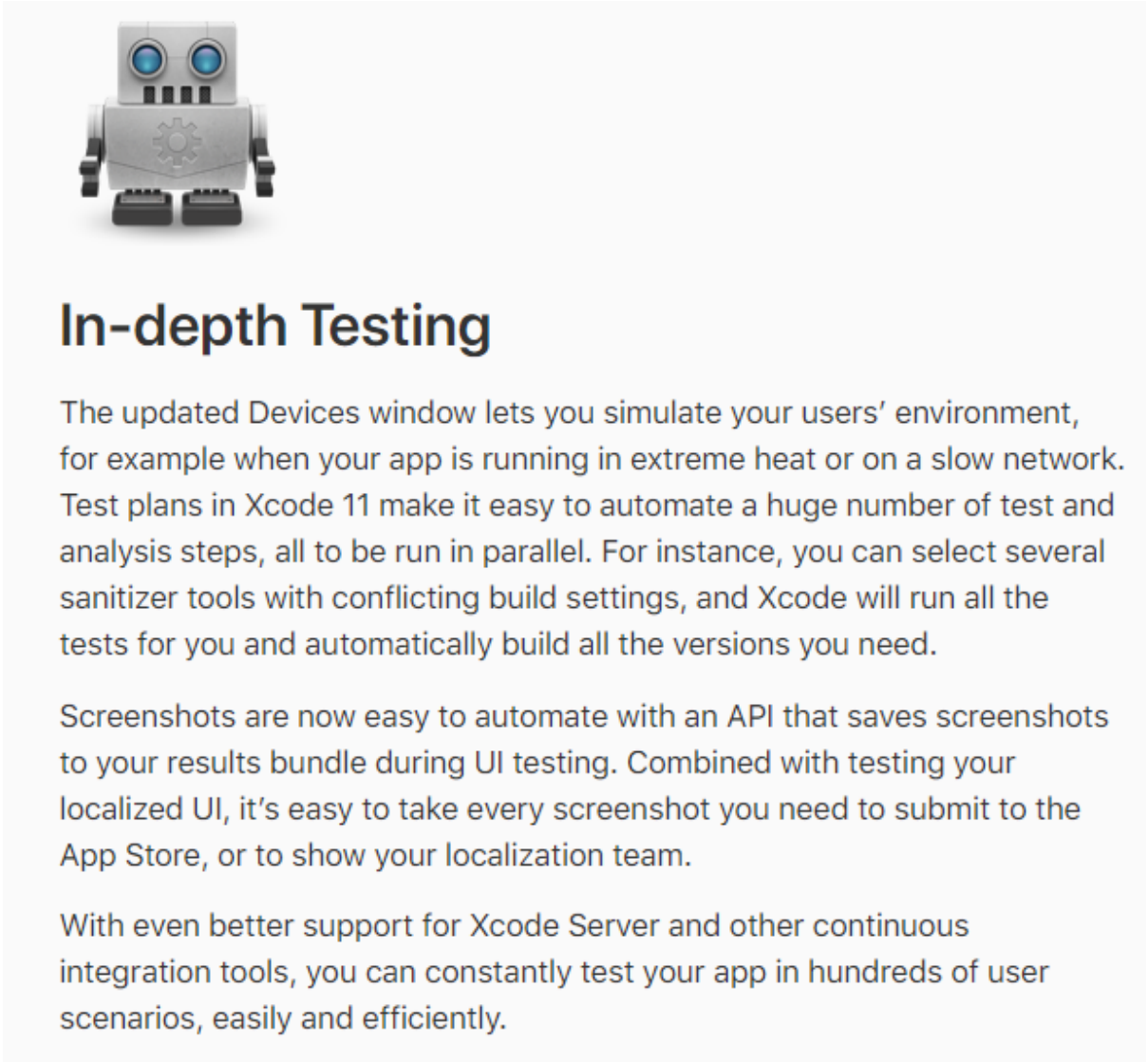
Xcode

Xcode — интегрированная среда разработки (IDE) программного обеспечения для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple. Первая версия выпущена в 2001 году. Стабильные версии распространяются бесплатно через Mac App Store. Зарегистрированные разработчики также имеют доступ к бета-сборкам через сайт Apple Developer.

Xcode включает в себя большую часть документации разработчика от Apple и Interface Builder — приложение, использующееся для создания графических интерфейсов.

Пакет Xcode включает в себя изменённую версию свободного набора компиляторов GNU Compiler Collection и поддерживает языки C, C++, Objective-C, Objective-C++ (англ.)русск., Swift, Java, AppleScript, Python и Ruby с различными моделями программирования, включая (но не ограничиваясь) Cocoa, Carbon и Java. Сторонними разработчиками реализована поддержка GNU Pascal[1], Free Pascal[2], Ada[3], C#[4], Perl[5], Haskell[6] и D[7]. Пакет Xcode использует GDB в качестве back-end’а для своего отладчика.

В августе 2006 Apple объявила о том, что DTrace, фреймворк динамической трассировки от Sun Microsystems, выпущенный как часть OpenSolaris, будет интегрирован в Xcode под названием Xray. Позже Xray был переименован в Instruments.



Углубленное тестирование

Окно обновленные устройства позволяет моделировать среду пользователей, например, когда приложение работает в условиях высокой температуры или в медленной сети. Планы тестирования в Xcode 11 позволяют легко автоматизировать огромное количество этапов тестирования и анализа, которые должны выполняться параллельно. Например, вы можете выбрать несколько инструментов дезинфекции с конфликтующими настройками сборки, и Xcode выполнит все тесты для вас и автоматически построит все необходимые версии.

Скриншоты теперь легко автоматизировать с помощью API, который сохраняет скриншоты в ваш пакет результатов во время тестирования пользовательского интерфейса. В сочетании с тестированием локализованного пользовательского интерфейса легко сделать каждый снимок экрана, который вам нужно отправить в магазин приложений, или показать свою команду локализации.

Благодаря еще лучшей поддержке Xcode Server и других инструментов непрерывной интеграции вы можете постоянно тестировать свое приложение в сотнях пользовательских сценариев, легко и эффективно.

AppSpector: платформа отладки iOS и Android приложений

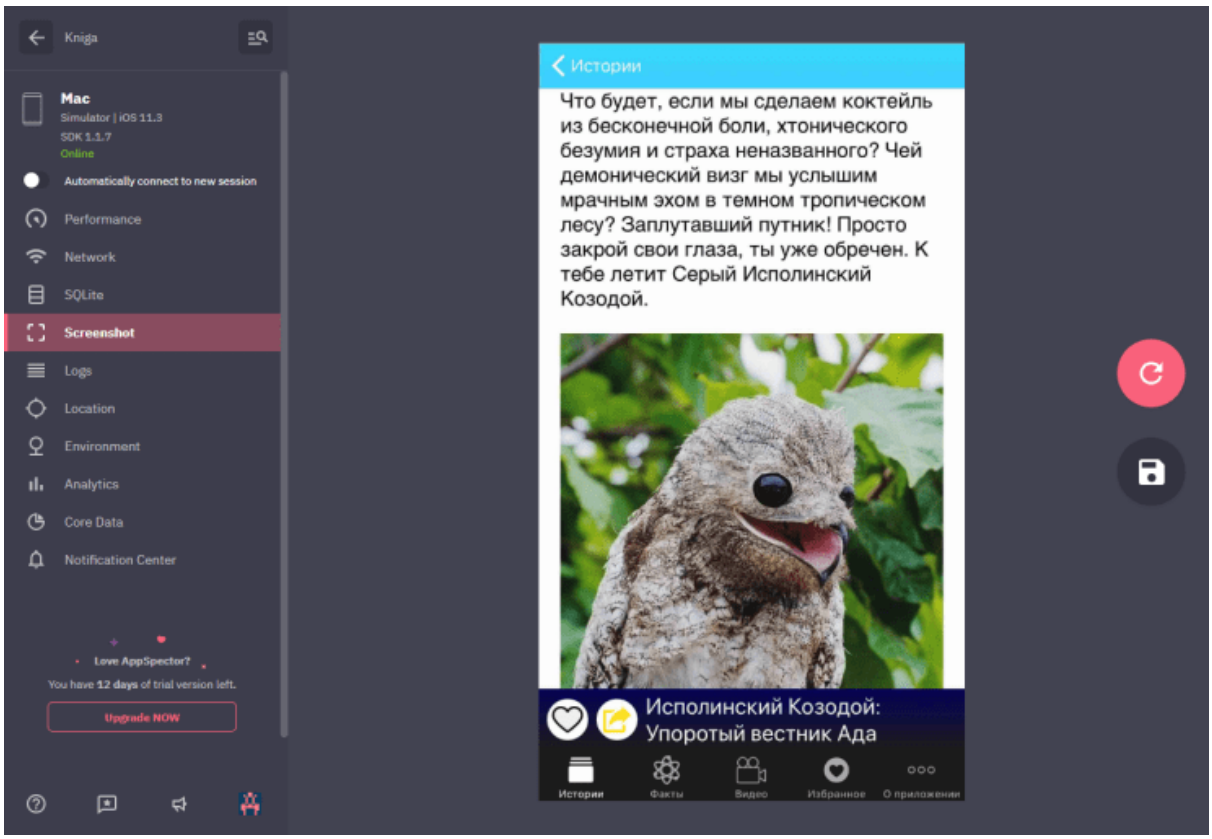
Отладка мобильных приложений всегда была делом непростым. Да, у Xcode и Android Studio есть свои встроенные инструменты для этого, но в них многого не хватает и чтобы получить полную картину происходящего с приложением зачастую приходиться прибегать к сторонним инструментам.

AppSpector – внешний SaaS-сервис для отладки iOS и Android приложений. Встроив SDK отладчика в свое приложение на стадии разработки, вы сможете в веб-панели:

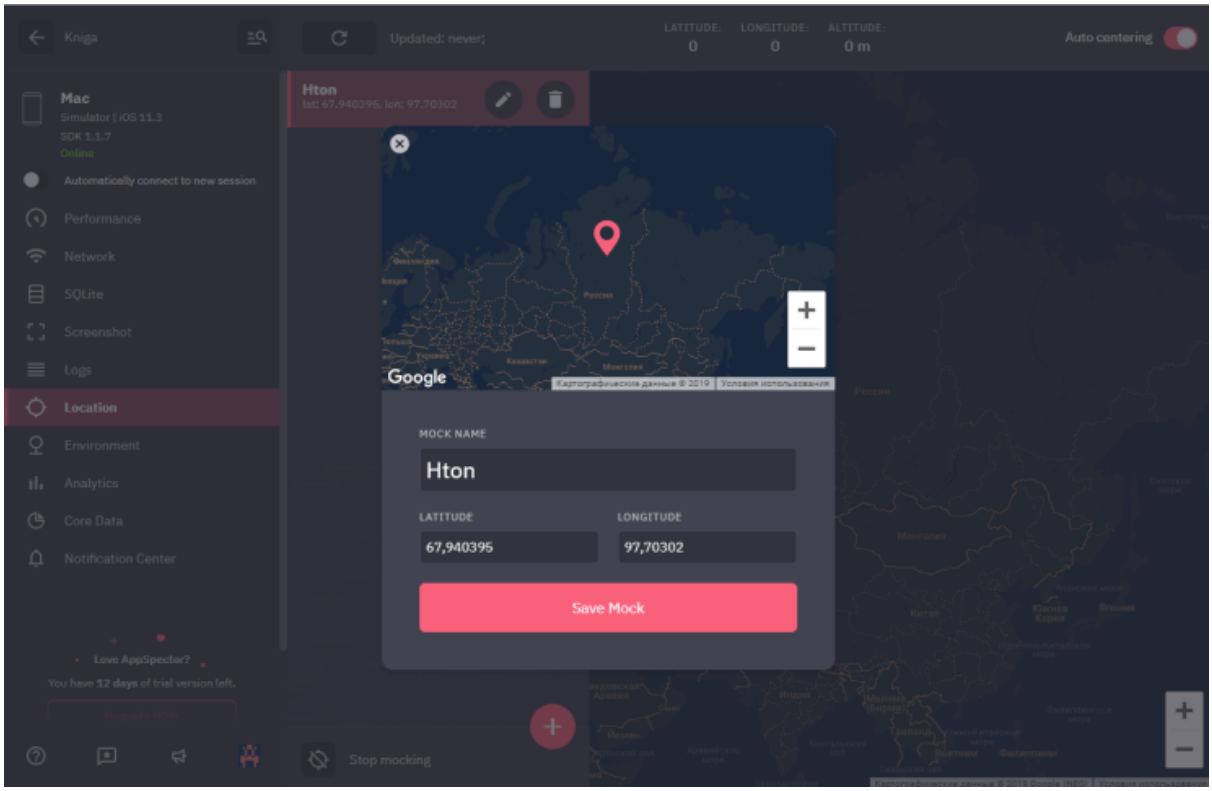
- Отслеживать быстродействие – загрузку процессора, потребление памяти, потребление трафика, объем данных на диске, частоту кадров в секунду.
- Отслеживать сетевые HTTP-запросы – адреса, их размер, продолжительность выполнения – с возможностью фильтрации.
- Изменять местоположение.
- Получить доступ к базам данных SQLite и Core Data на устройстве – к таблицам, данным, используемым View и триггерам, схемам, запросам и т.п. – и даже делать запросы прямо из сервиса.
- Получать скриншоты в реальном времени.
- Получать логи с устройства.
- Смотреть переменные окружения
- Отслеживать уведомления Notification Center.

Все это в реальном времени, для реальных устройств и симуляторов. Сервис автоматически подключается к запущенным приложениям и пишет для них данные.

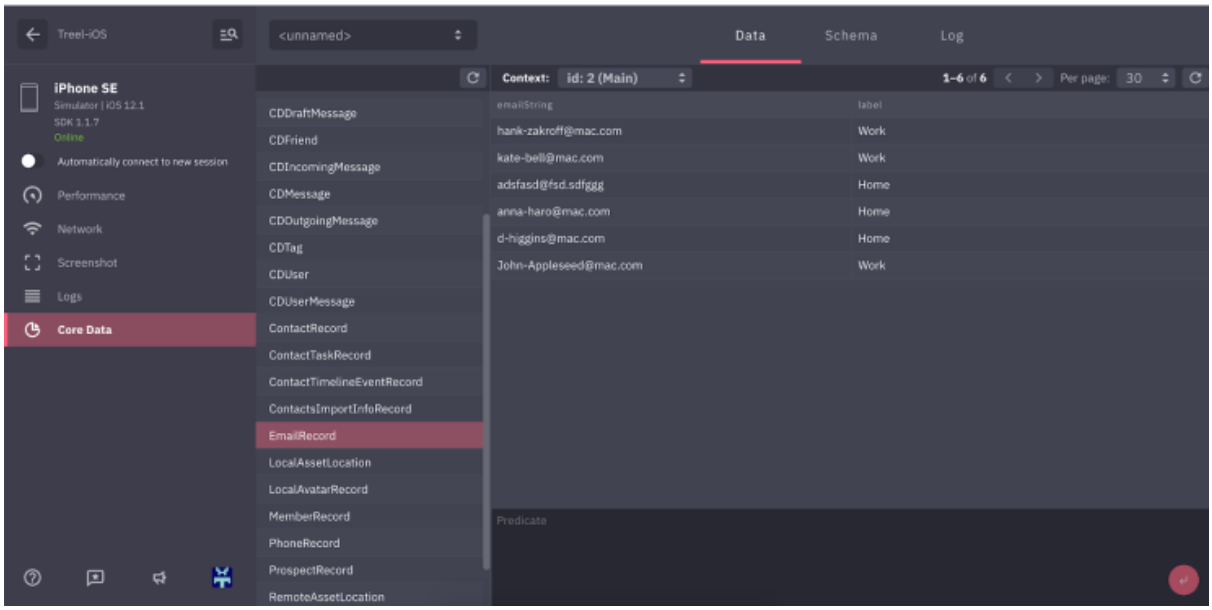
Мгновенный скриншот с симулятора на Mac сразу в Windows (и сразу правильного размера – можно заливать в App Store, очень удобно):



Мокинг местоположения – вы можете задать произвольную точку, в которой окажется ваше приложение:



Доступ к CoreData и SQLite базам данных в реальном времени.



Все это с обновлением в реальном времени.

Прекрасный логотип AppSpector – резиновый утенок, знакомый многим программистам.

Метод утёнка — психологический метод решения задачи, делегирующий её мысленному помощнику. Суть метода заключается в том, что тестируемый ставит на рабочем столе игрушечного утёнка (или представляет его мысленно, на самом деле уточка — это условно, предмет может быть любым), и когда у него возникает вопрос, на который трудно ответить, то он задаёт его игрушке, как живому человеку, словно она действительно может ответить. Считается, что правильная формулировка вопроса содержит как минимум половину ответа, а также это даёт толчок мыслям, направляя их в нужное русло, переводя “поток сознания”, фактически – “кашу в голове”, в вид формальных терминов.