

План анализа IDE Rider

- Общая характеристика продукта
 - История создания
 - Позиционирование
 - Характер решаемых задач
- Необходимое аппаратное и программное обеспечение
 - Минимальные аппаратные требования
 - Необходимое программное обеспечение для работы
 - Порядок приобретения продукта
 - Порядок установки продукта
 - Первоначальная настройка
 - Настройка окружения
- Основные функции
 - Навигация по проекту
 - Ориентация в исходном коде с помощью...
 - Navigate To
 - Авто- завершение и генерация кода
 - Краткое описание
 - Основные действия
 - Подробнее о генерации
 - Рефакторинг: редактирование и работа с кодом
 - Refactor This...
 - Основные рефакторинги
 - Исправление и глубокий анализ кода
 - Анализ кода
 - Проблемы
 - Проверка кода
 - Навигация и исправление
 - Форматирование кода
 - Отладка
 - Типичный алгоритм отладки
 - Диагностика кода
 - Плагины
 - Функции плагинов
 - Подборка полезных плагинов

Общая характеристика продукта

История создания

Rider был анонсирован в 2015 году компанией JetBrains. Был выпущен в 2017 году. В основе IDE лежит идея создания среды с широким набором рефакторинга кода и ускорения технологических процессов создания продукта. В основе Rider лежит другое решение компании - ReSharper, плагин для повышения продуктивности работы в Microsoft Visual Studio.

Позиционирование

Rider — это кросс-платформенная IDE, поддерживающая операционные системы Windows, macOS и Linux. Он обеспечивает расширенную поддержку редактирования и анализ кода для языков, используемых при разработке .NET: от C#, VB.NET и F# до синтаксиса ASP.NET Razor, JavaScript, TypeScript, XAML, XML, HTML, CSS, SCSS, JSON и SQL.

Характер решаемых задач

Программа распространяется по проприетарной лицензии, широко используется в крупных IT-компаниях, например Adobe. IDE Rider позволяет разрабатывать .NET, ASP.NET, .NET Core, Xamarin или Unity приложения. Может использоваться для разработки игр (Unity, Unreal Engine), работы в облаке (AWS, Azure), .NET, WEB, нативных приложений. Так же поддерживаются технологии Entity Framework, Docker.

Несмотря на большой набор функций, Rider — быстрая и отзывчивая IDE, которая делает упор на инспекцию и рефакторинг кода.

Необходимое аппаратное и программное обеспечение

Минимальные аппаратные требования

- Процессор: 2 GHz
- Память: 4 GB
- Диск: 2.5 GB

Необходимое программное обеспечение для работы среды

- Windows - 8.1, 10, 11, (.NET Framework 4.7.2 или более поздняя версия).
- macOS - 10.14+
- Linux - Debian 9+, Ubuntu 16.04+, CentOS 7+, Fedora 30+, Alpine 3.13+ или другие ОС с GLIBC 2.17 и новее.

Порядок приобретения продукта

- Пробная версия на первые 30 дней пользования.
- Бесплатно для студентов и преподавателей, проектов с открытым исходным кодом.
- Далее пользователям, ранее владевшим студенческой лицензией, предоставляется скидка 25% на приобретение новой персональной подписки на один из продуктов JetBrains или на All Products Pack.
- Скидка 50% предоставляется для стартапов, университетов и образовательных учреждений (для разработки коммерческих проектов), некоммерческих организаций.
- Частным лицам годовая лицензия предоставляется за €149.00 в первый год, €119.00 - во второй и €89.00 - начиная с третьего (все это цены без НДС).
- Организациям годовая лицензия предоставляется за €419.00 в первый год, €335.00 - во второй и €251.00 - начиная с третьего (все это цены без НДС).

Порядок установки продукта

- После приобретения лицензии, на странице установки, доступной из личного кабинета JetBrains, появится ссылка для скачивания последней версии продукта.
- Так же предлагается установить Toolbox App, в котором можно устанавливать и обновлять все доступные продукты компании.

Первоначальная настройка

- Доступен импорт настроек из других IDE, чтобы переход к Rider был максимально плавным - можно сохранить привычные сочетания клавиш.
- Так же сочетания клавиш можно настроить по шаблонам из Visual Studio, ReSharper (расширение для Visual Studio) и собственной схеме Rider.
- Предлагают выбрать схему подсветки синтаксиса: ReSharper, Visual Studio, Dracula.
- Rider по умолчанию имеет множество встроенных дополнений, которые служат для поддержки веб-разработки, работы с базами данных, разработки игр и так далее - при установке предлагается включить нужные модули.
- Для работы Rider необходимо установить дополнительное ПО, что так же предлагается сделать во время первоначальной настройке.

Настройка окружения

- Огромное количество тем и стилей подсветки синтаксиса на любой вкус.
- Рабочие области, такие как проводник, терминалы и тому подобные, можно свободно перемещать и клонировать.

Основные функции

Навигация по проекту

Ориентация в исходном коде с помощью...

- Сигнатур и объявлений.
- Переменных и типов.
- Методов.
- Иерархии наследования.
- Ошибок.

А также с помощью интерфейса IDE: действий, настроек и окон с инструментами. Интерфейс Rider гибко настраивается, расположение вкладок, окон инструментов можно менять по собственному усмотрению - удалять, дублировать и перемещать.

Navigate To

Навигация по коду в Rider реализована через общее окно поиска **Navigate To**. Проще всего его вызвать двойным нажатием **Shift**. Область поиска **Navigate To** зависит от положения, в котором находится курсор. В зависимости от типа элемента, на которой указывает курсор, в диалогов окне будет предложено произвести поиск:

- По вашему собственному коду и подключенным библиотекам по имени.
- По структурным элементам выделенного: перейти к объявлению, объявлению типа, базовому типу в иерархии наследования, к фактическому использованию объявленных типов в коде, перейти на наследникам и производным типа и так далее.

В целом, **Navigate To** может реализовывать поиск в любой области видимости, в том числе извлекать исходный код используемых библиотек с официальных сайтов разработчиков. И положение курсора позволяет интеллектуально настраивать фильтры поиска в диалоговом окне.

Навигация по проекту происходит с помощью окна **Explorer**. Также функционал навигации может выполнять **TODO**, отдельная вкладка проекта, в которой можно создавать трекеры к элементам кода, к которым необходимо вернуться позже, либо же ошибки в которых необходимо устранить. TODO позволяет как выставлять приоритет, так и формировать гибкое описание действий, которые требует таргетированный участок.

Некоторые горячие клавиши

Поиск по тексту:

Сочетание	Операция
Ctrl+F / Ctrl+H	Поиск и замена кода по проекту
Ctrl+Shift+F / Ctrl+Shift+H	Поиск и замена кода по выбранным фильтрам видимости
Ctrl+F3	Поиск слова вне зависимости от его структурного значения в проекте

Поиск по имени:

Сочетание	Операция
Ctrl+T / двойной Shift	Navigate To
Ctrl+Alt+Shift+T	Поиск по символам (Типы, методы, поля...)
Ctrl+Shift+T	Поиск файла

Навигация по символам:

Сочетание	Операция
F12	Перейти к объявлению
Ctrl+Shift+F11	Перейти к объявлению типа
Ctrl+F12	Найти фактическую реализацию типа, место использования
Alt+Home	Перейти к базовому типу по иерархии наследования
Alt+End	Перейти от типа к его производным вниз по иерархии наследования

Навигация по коду при редактировании:

Сочетание	Операция
Ctrl+Left / Ctrl+Right	Перейти к следующему или предыдущему слову
Home / End	Перенести курсор к первому или последнему слову в строке

[Другие хоткеи навигации по коду.](#)

Авто- завершение и генерация кода

Краткое описание

Интеллектуальный редактор кода предоставляет несколько видов автодополнения, шаблоны для написания типовых конструкций и постфиксные шаблоны, позволяет редактировать код в нескольких местах одновременно и быстро перемещаться по иерархии наследования с помощью иконок на полях. Rider импортирует недостающие пространства имен, вставляет парные скобки, подсвечивает границы блоков кода. Для перехода к рефакторингам, генерации кода, командам навигации и контекстным действиям, нужно нажать всего пару клавиш.

Основные действия

Список предложений автоматического завершения появляется, как только вы начинаете вводить новый идентификатор. Список предложений завершения появляется, когда вы нажимаете **Ctrl+Space**.

Две другие команды завершения, завершение с сопоставлением типов **Ctrl+Alt+Space** и второе базовое завершение **Alt+Shift+Space**, применяют расширенные алгоритмы для добавления дополнительных элементов в список предложений.

По умолчанию все функции завершения JetBrains Rider поддерживают **CamelHumps**, то есть вы можете ввести начальные буквы частей составного имени, и имя появится в списке предложений.

Когда вы используете автозавершение кода поверх существующих элементов кода, вы можете либо вставить выбранное предложение завершения перед существующим элементом, нажав **Enter**, либо заменить существующий идентификатор выбранным предложением, нажав **Tab**. При необходимости вы можете изменить сочетания клавиш по умолчанию на странице **Keymap** в настройках JetBrains Rider **Ctrl+Alt+S**.

При выборе вызова метода из списка завершения JetBrains Rider по умолчанию автоматически вставляет пару круглых скобок **()** и устанавливает между ними курсор.

Когда список завершения открыт, вы можете нажать **Ctrl+Down** или **Ctrl+Up**, чтобы закрыть его и переместить курсор вниз или вверх в редакторе.

JetBrains Rider может использовать модели машинного обучения, чтобы ранжировать наиболее подходящие элементы выше в списках предложений, по умолчанию эта функция выключена.

Сочетание	Операция
Ctrl+Space	Базовое завершение
Ctrl+Alt+Space	Типовое завершение
Alt+Shift+Space	Второе базовое завершение
Ctrl+Shift+Enter	Закончить текущее выражение

[Больше можно узнать в документации.](#)

Подробнее о генерации

Один из самых простых способов генерировать код с помощью JetBrains Rider — использовать завершение кода. Все, что вам нужно сделать, это ввести несколько символов.

JetBrains Rider позволяет использовать классы, методы, переменные, свойства, поля и т. д. до их объявления. Когда JetBrains Rider обнаруживает необъявленный символ, он предлагает одно или несколько быстрых исправлений **Alt+Enter** для создания объявления, а затем корректирует объявление в соответствии с контекстом использования.

С помощью JetBrains Rider вы можете быстро создавать члены различных типов. Чтобы просмотреть список доступных параметров генерации кода для текущего типа, нажмите **Alt+Insert**.

Сочетание	Операция
Alt+Insert	Сгенерировать...

[Больше можно узнать в документации.](#)

Рефакторинг: редактирование и работа с кодом

Refactor This...

Сочетание	Операция
Ctrl+Shift+R	Refactor This...

Эта команда является ярлыком для всех рефакторингов JetBrains Rider, доступных в текущем контексте. Вы можете вызвать эту команду практически из любого места:

- При объявлении или использовании символа в редакторе.
- На выделении в редакторе.
- На текущем элементе или выделении в обозревателе решений.

Основные рефакторинги

Сочетание	Операция
Ctrl+R, R	Переименовать...
Ctrl+R, O	Переместить...
Ctrl+R, M	Извлечь метод...
Ctrl+R, V	Объявить переменную...

Переименовать...

Этот рефакторинг позволяет изменить имя любого символа или проекта в вашем решении. Все ссылки на символ и его использования будут обновляться автоматически.

Переместить...

Рефакторинги в этой группе помогают перемещать сущности разных типов в другие типы, пространства имен, файлы и папки в рамках вашего решения.

Извлечь метод...

Этот рефакторинг позволяет вам создать новый метод или локальную функцию, основанные на выделенном фрагменте кода. JetBrains Rider анализирует выделенные выражения и находит переменные, которые могут быть конвертированы в параметры метода или представлять возвращаемое значение.

Объявить переменную...

Этот рефакторинг позволяет создать новую локальную переменную или константу на основе выбранного выражения, инициализировать ее выражением и, наконец, заменить все вхождения выражения в методе ссылками на вновь введенную переменную.




Исправление и глубокий анализ кода

Анализ кода

JetBrains Rider обеспечивает статический анализ кода, применяя больше 2500 статистических паттернов для C#, XAML, XML, Razor, JavaScript, TypeScript, HTML и других языков.

По умолчанию IDE начинает анализировать код в момент открытия редактора и продолжает это делать автоматически в процессе всего написания кода. Результат автоматического анализа - проблемы, выявленные в текущем проекте и отсортированные по уровню серьезности. Инспекцию кода можно запустить вручную, проверяя отдельные файлы проекта или даже отдельные участки.

Проблемы

-  Ошибка - проблемы с кодом, которые либо препятствуют компиляции, либо приводят к ошибкам во время выполнения.
-  Предупреждение - проблемы кода, не препятствующие компиляции, но вызывающие предупреждения компилятора или снижающие эффективность. Например, избыточное приведение типов или неиспользуемые аргументы и типы.
-  Предложение - проблемы, не влияющие на работоспособность программы, однако нарушающие принятые паттерны написания кода в конкретном языке или продукте.
-  Подсказка - рекомендация по использованию возможностей языка или IDE, которые бы упростили задачу, решаемую в конкретной области вашего кода.

Проверка кода

Ознакомиться с окном проверки ошибок можно с помощью **Alt+6**. В то же время настроить проверку ошибок можно в диалоговом окне **Ctrl+Alt+S**, выбрав интересующие категории сканирования:

- Проблемы с качеством кода.
- Проблемы читаемости кода.
- Проблемы избыточности и неиспользуемого кода.
- Проблемы использования устаревших конструкций языка.
- Проблемы, связанные со стилем кода.
- Проблемы области видимости кода.
- Проблемы, вызванные предупреждением компилятора.
- Проблемы синтаксиса кода и проверки правописания.
- Проблемы форматирования кода.
- Проблемы, выявленные подключаемыми плагинами для выбранного языка или компилятора.

Навигация и исправление

Навигация по выявленным проблемам может осуществляться с помощью окна проверки ошибок **Alt+6**, там же может быть выполнено автоматическое решение ошибки. Перейдя к интересующему вас месту проблемы, можно воспользоваться Quick-fixes сочетанием клавиш **Alt+Enter**, который предложит список возможных исправлений.

Форматирование кода

Стиль кода, который включает в себя стандарты именования, правила форматирования, макет файла, стиль заголовка файла, порядок модификаторов и так далее, настраивается в **Code Style Configuration**. В нем можно как использовать правила

форматирования от JetBrains, так и создать собственные правила для конкретного проекта или профиля форматирования, используя маски и примеры исправления.

Используя статистический анализ кода и проверку, Rider может отмечать участки, в которых не соблюдены требования **Coding conventions** или присутствует избыточность, проблемы с читаемостью. Кроме того, проверка позволяет определить проблемы синтаксиса, не влияющие на работоспособность программы.

Исправить выявленные проблемы можно с помощью Quick-fixes **Alt+Enter** или же специальным инструментом Code cleanup **Ctrl+E**, с. также можно вызвать инструмент форматирования всего файла **Ctrl+Alt+Enter**.

Отладка

Типичный алгоритм отладки

1. Определить конфигурацию отладки. Она может быть **Temporary** и создаваться каждый раз при запуске или отладке проекта, а также **Permanent** и настраиваться отдельно для конкретных задач одноплатформенной отладки модулей.
2. Установка **Breakpoints** **F9**. Точки остановки могут иметь разный статус, от активного до условного, их гибкая настройка позволяет более эффективно изолировать ошибку и отслеживать значения параметров в зависимости от заданных условий хода отладки. Точки остановки могут запускать процесс трассировки и отслеживать текущее состояние и ресурсы программы.
3. Запуск программы в режиме отладки с помощью **Alt+F5**. Переход к следующей точке остановки осуществляется с помощью **F5**.
4. Навигация в пошаговой отладке осуществляется с помощью:

Сочетание	Операция
F11	Шаг (Step Into)
Shift+F7	Умный шаг (Smart Step Into)
Shift+F11	Выйти из текущего оператора (Step Out)
F10	Выполнить текущий оператор и остановиться перед следующим (Step Over)
Alt+Shift+F8	Принудительно выполнить текущий оператор, игнорируя точки остановки (Force Step Over)
Ctrl+F10	Выполнение программы до места, в котором стоит курсор (Run to Cursor)
Ctrl+Alt+F9	Выполнение программы до места, в котором стоит курсор, игнорируя точки остановки (Run to Cursor Non-Stop)
Ctrl+Shift+F10	Начать отладку с указателя курсора (Skip to Cursor)

5. Отредактировать код в режиме отладки. Благодаря **Hot Reload** изменения, внесенные во время отладки, автоматически синхронизируются с текущей конфигурацией и применяются в момент, когда вы используете одну из пошаговых команд или **F5**.
6. Для остановки отладки следует нажать **Shift+F5**

Диагностика кода

С помощью плагина **NUnit** в режиме отладки возможна отладка юнит-тестами в .NET. В других языках и фреймворках могут использоваться **MSTest**, **xUnit**, **bUnit** и так далее. Во время отладки также может быть полезен встроенный в IDE плагин **dotTrace** для профилактики производительности программы - анализировать можно как отдельный метод, так и общее время выполнения программы. Встроенный **dotMemory** позволяет отслеживать утечку памяти и количество её выделения для отдельных функций или программы в целом. Плагины для разработки .NET встроены в Rider, аналогичные присутствуют в **JetBrains Marketplace** для других языков и фреймворков.

Плагины

Функции плагинов

- Интеграция с системами контроля версий, системами отслеживания ошибок, серверами управления сборкой и другими инструментами.
- Поддержка для различных языков и фреймворков.
- Подсказки по шорткатам, предварительный просмотр в реальном времени и т. д.
- Упражнения по программированию, которые помогут вам выучить новый язык программирования.

Подборка полезных плагинов

1. Key Promoter X - поможет выучить все комбинации клавиш, каждый раз показывая уведомление, что действие можно было выполнить с помощью хоткея.
2. Rainbow brackets - раскрасит пары скобок в разные цвета, чтобы не запутаться в их нагромождении.
3. Nyan Progress Bar - превратит шкалу загрузки в что-то менее скучное.