Краткое руководство

PhpStorm - это интегрированная среда разработки для PHP-разработчиков, построенная на платформе **IntelliJ IDEA**.

Поддерживаемые языки

С помощью PhpStorm вы можете разрабатывать приложения на PHP 5.3, PHP 5.4, PHP 5.5, PHP **5.6**, PHP 7, PHP 7.1, PHP 7.2, PHP 7.3, PHP **7.4**, PHP **8.0**, PHP 8.1 и **PHP 8.2.** Подробнее см. в разделе Поддерживаемые версии PHP. Кроме того, он полностью поддерживает HTML5, CSS, JavaScript и XML: поддержка этих языков осуществляется с помощью плагинов, которые входят в состав IDE и включены по умолчанию. Поддержка других языков может быть добавлена с помощью плагинов: в диалоге **настроек** (Ctrl+Alt+S), перейдите в **раздел «Плагины»**, чтобы узнать больше, или настройте их во время первого запуска IDE.

Поддерживаемые платформы

PhpStorm — это кроссплатформенная IDE, работающая на **Windows**, **macOS** и **Linux**.

Требования к системе

Требование	Минимум	Рекомендованный
БАРАН	2 ГБ свободной оперативной памяти	8 ГБ общей оперативной памяти
ЦПУ	Любой современный процессор	Многоядерный процессор. PhpStorm поддерживает многопоточность для различных операций и процессов, что делает его тем быстрее, чем больше ядер процессора он может использовать.
Диске	3,5 ГБ	SSD-накопитель со свободным пространством не менее 5 ГБ
Разрешение монитора	1024×768	1920×1080 гг.
Операционная система	Официально выпущены 64-битные версии следующего:	Последняя 64-разрядная версия Windows, macOS или

Требование	Минимум	Рекомендованный
	• Microsoft Windows 10 1809 или более поздняя версия Windows Server 2019 или более поздней версии • macOS 10.15 или новее • Любой дистрибутив Linux, поддерживающий Gnome, KDE или Unity DE. PhpStorm недоступен для дистрибутивов Linux, которые не включают GLIBC 2.27 или более позднюю версию. Предварительные версии не поддерживаются.	Linux (например, Debian, Ubuntu или RHEL)

Вам не нужно устанавливать Java для запуска PhpStorm, так как JetBrains Runtime поставляется в комплекте с IDE (на основе JRE 17).

Настройка среды PHP

PhpStorm - это умная IDE, но сначала вам нужно сообщить ей, какую среду PHP мы используем, где хранятся компоненты PHP и как они настроены.

- Если вы работаете с локальным веб-сервером, выполните процедуру установки пакета AMP, чтобы подготовить среду.
- Если вы работаете в **контейнере** Docker, см. раздел Поддержка Docker в PhpStorm.
- Если вы используете Vagrant, см. раздел Поддержка Vagrant в PhpStorm.
- Если вы используете WSL, см. раздел Поддержка WSL в PhpStorm.

В этом **кратком руководстве** мы настроим предварительно настроенный проект в среде **Docker**.

Прежде чем вы начнете

1. Установите Docker для вашей операционной системы.
2. В PhpStorm в диалоговом окне «**Настройки**» (Ctrl+Alt+S), перейдите в раздел **Сборка, выполнение, развертывание** | Docker и выберите способ подключения к демону Docker.

   В зависимости от операционной системы выполните следующие действия.

   Виндоус

   macOS

   Линукс

. Выберите **сокет TCP**.

b. **Задайте для параметра URL-адрес API ядра** значение **tcp://**.

В разделе **Общие** параметров **Docker для Windows** включите **управляющую программу предоставления на tcp://localhost:2375 без TLS**.

c. Оставьте поле **Папка Сертификаты** пустым.
   Если вы используете Docker Toolbox, используйте следующие параметры конфигурации:
d. **Задайте для параметра URL-адрес API движка** значение **https://192.168.99.100:2376**
e. Задайте **для папки "Сертификаты" значение <your_home_directory>**.

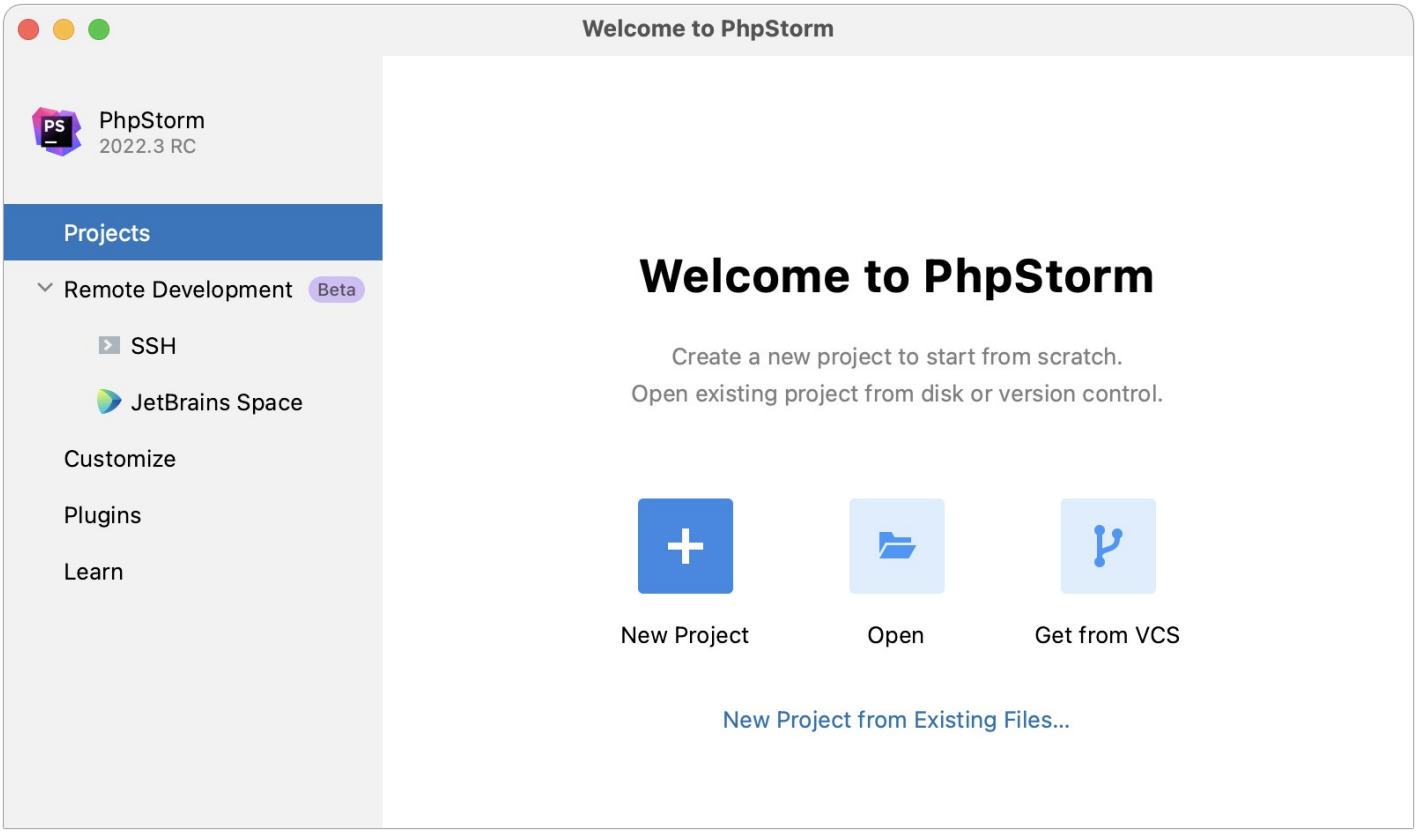
Шаг 1: Откройте проект в PhpStorm

Все, что вы делаете в PhpStorm, делается в контексте проекта. Он служит основой для помощи в написании кода, массового рефакторинга, согласованности стилей кодирования и так далее.

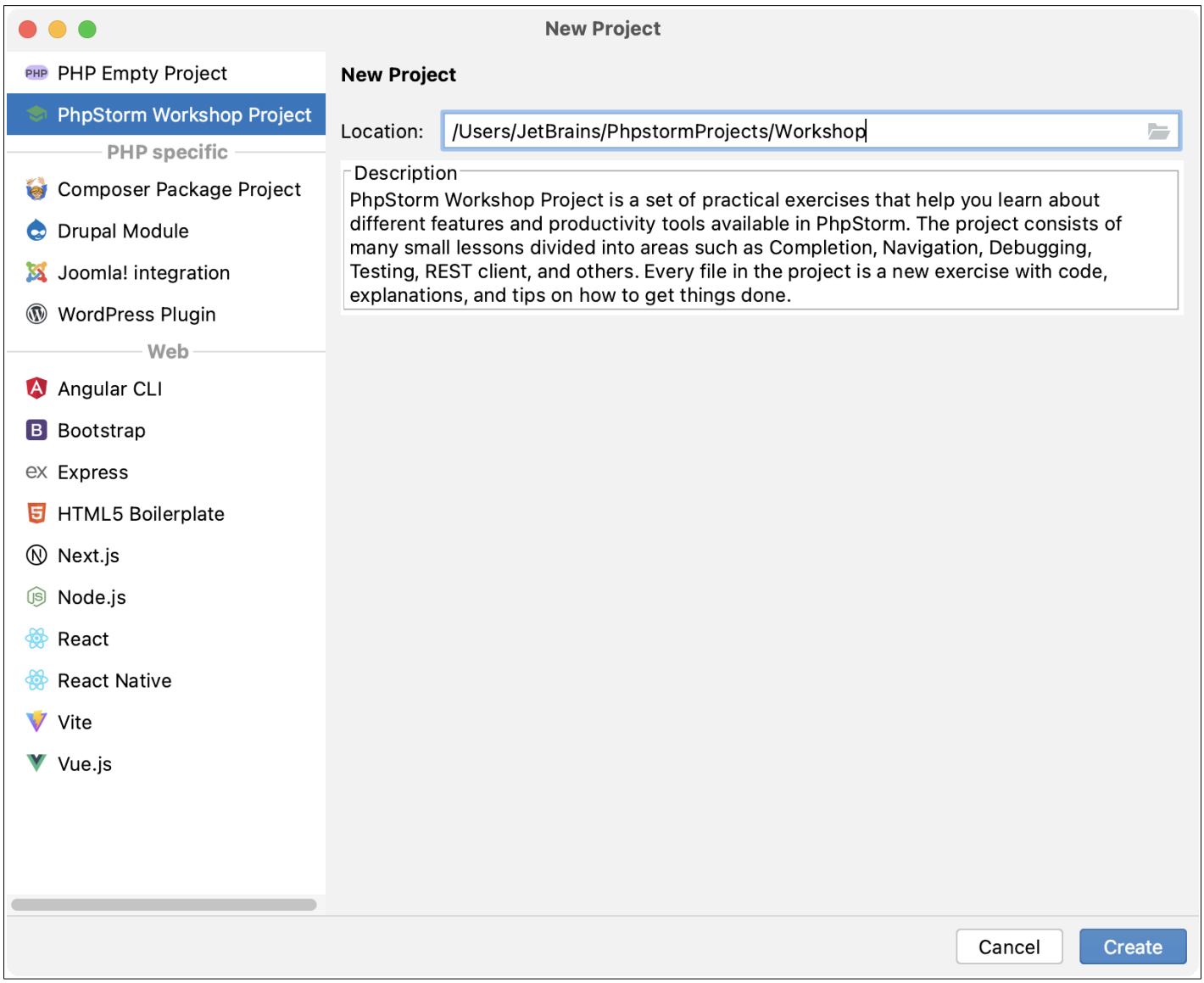
В этом **кратком руководстве** мы предоставляем проект мастерской, который уже содержит файл **docker-compose.yml** с предварительно настроенной средой. Проект доступен на GitHub по адресу https://github.com/JetBrains/phpstorm-workshop.

Запуск проекта Workshop в PhpStorm

1. На экране приветствия PhpStorm нажмите кнопку **Создать проект**.



2. **Выберите PhpStorm Workshop Project** из списка слева, укажите местоположение проекта в поле «**Местоположение**» и нажмите «**Создать**».



3. Откройте **docker-compose.yml** в редакторе и обновите переменную XDEBUG_CONFIG значением в зависимости от вашей операционной системы. Это необходимо для веб-отладки.

   Виндоус
   macOS
   Линукс

   Используйте значение host.docker.internal, которое относится к удаленному узлу, то есть к компьютеру, на котором запущен Docker. Он автоматически разрешится во внутренний адрес хоста, что позволит вам подключиться к нему из контейнера.
   Соответствующая часть файла **docker-compose.yml** должна выглядеть следующим образом:

   ```
   XDEBUG_CONFIG: remote_host=host.docker.internal
   ```

4. В том же файле **docker-compose.yml** раскомментируйте соответствующую строку для службы sftp в зависимости от вашей операционной системы. Это необходимо для правильной работы развертывания.
5. Щелкните ▶▶ в желобе редактора, чтобы запустить все необходимые контейнеры Docker.

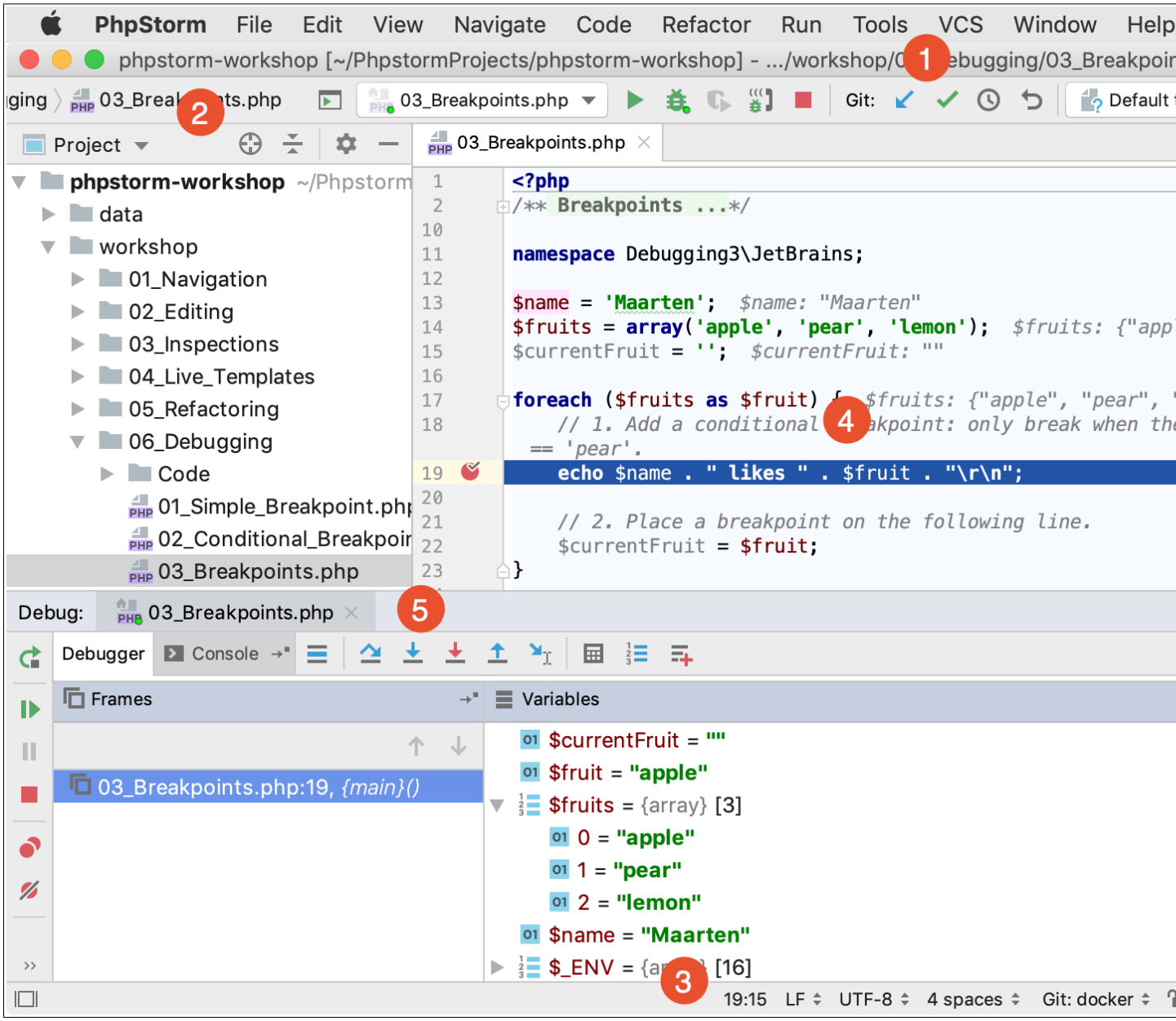
docker-compose.yml

```
version: '2'
services:
  debug:
    image: phpstorm/php-71-apache-xdebug
    ports:
      - "8081:80"
```

Кроме того, откройте встроенный терминал PhpStorm (Alt+F12) и выполните команду docker-compose up.

Шаг 2: Изучите пользовательский интерфейс

Главное окно PhpStorm разделено на несколько логических областей:



1. **Меню** и **панели инструментов**, которые помогают нам выполнять различные команды.
2. **Навигационная панель** для навигации по проекту.
3. **Строка состояния** с различной информацией обо всем PhpStorm, текущем проекте или файле в редакторе, предупреждениями и сообщениями об ошибках.

4. **Редактор**, в котором вы пишете свой код. Он имеет вкладки для удобной навигации между открытыми файлами.
5. **Многочисленные окна инструментов**, которые выполняют различные функции: помогают исследовать и перемещаться по проектам и файловым структурам, просматривать результаты поиска и проверки, запускать, отлаживать и тестировать приложения, работать в интерактивных консолях и многое другое.

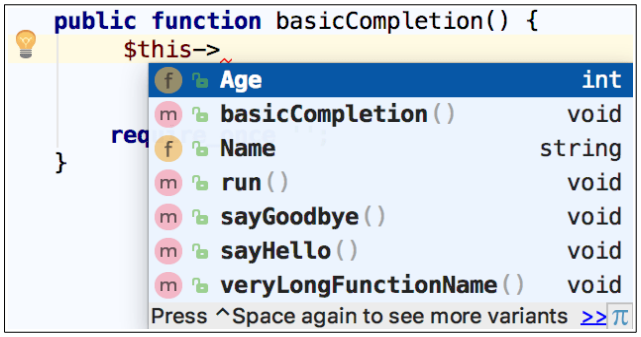
Шаг 3: Код с умной помощью

PhpStorm позаботится о рутине, чтобы вы могли сосредоточиться на важном. Используйте многочисленные возможности кодирования PhpStorm для создания безошибочных приложений, не теряя драгоценного времени. В базовом проекте вы можете интерактивно опробовать эти функции на ряде файлов PHP с примерами кода и рекомендациями.

Автозавершение кода

Автозавершение кода значительно экономит время независимо от типа файла, с которым вы работаете. В PhpStorm есть два типа автозавершения кода: базовое и сопоставление типов.

Базовая комплектация Ctrl+Space Отображает параметры автозавершения кода для текущего контекста и выделяет элементы текущего типа полужирным шрифтом:



Чтобы отобразить больше вариантов, нажмите Ctrl+Space снова.

Чтобы попробовать **базовое автозавершение кода**, откройте **workshop** в своем проекте и следуйте инструкциям в комментариях.

Интеллектуальное завершение сопоставления типов Ctrl+Shift+Space анализирует контекст, в котором вы работаете в данный момент, и предлагает более точные предложения на основе этого анализа, фильтруя список функций и переменных в соответствии с типом выражения.

Чтобы попробовать **смарт-завершение кода**, откройте **workshop** в своем проекте и следуйте инструкциям в комментариях.

Действия намерения

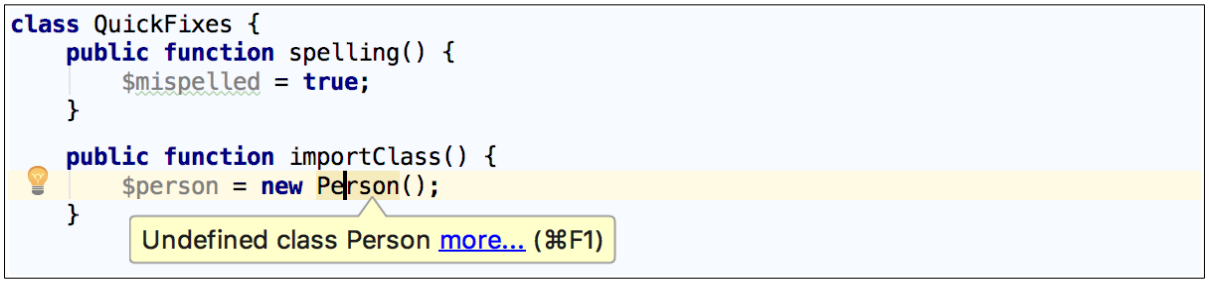
PhpStorm следит за тем, что вы делаете в настоящее время, и делает умные предложения, называемые действиями намерения, чтобы

сэкономить больше вашего времени. Действия намерения позволяют автоматически вносить изменения в **правильный** код (в отличие от проверок кода, которые предоставляют быстрые исправления для кода, который **может быть неправильным**).
Ваш код ссылается на файл, который не существует? Не проблема с PhpStorm. Давить Alt+Enter и выберите **Создать файл <имя файла>**:

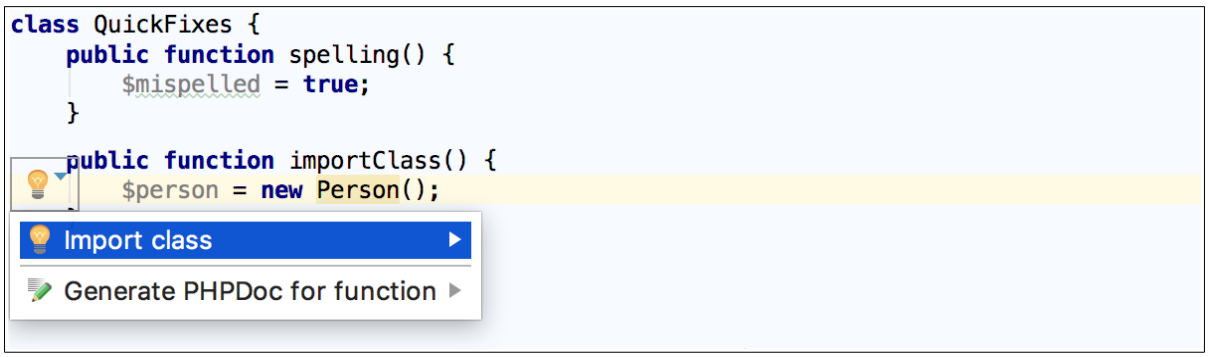
Чтобы просмотреть полный список доступных действий с намерением, в диалоговом окне «**Настройки**» (Ctrl+Alt+S) перейти в **Редактор | Намерения**.
Чтобы попробовать различные **действия с намерением**, откройте **workshop** в своем проекте и следуйте инструкциям в комментариях.

Шаг 4: Держите свой код аккуратным

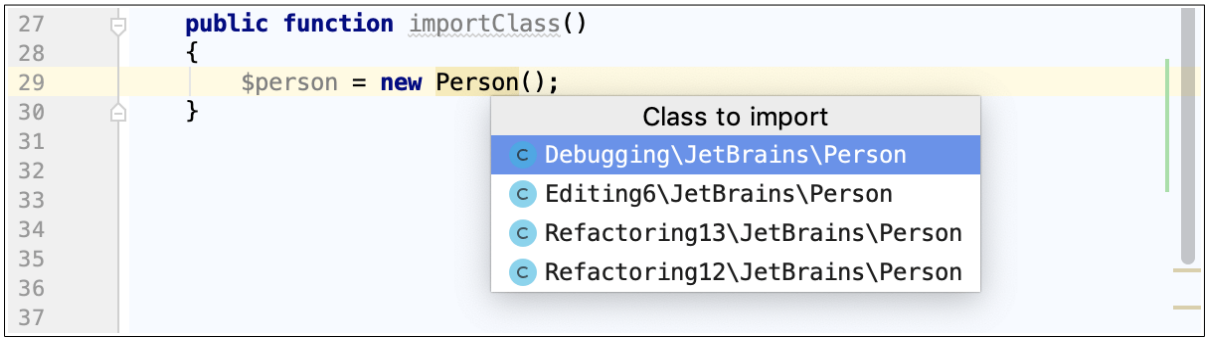
PhpStorm следит за вашим кодом и старается поддерживать его точность и чистоту. Он обнаруживает потенциальные ошибки и проблемы и предлагает быстрые решения для них.
Каждый раз, когда PhpStorm находит неиспользуемый код, бесконечный цикл, скрытую верхнюю область, оператор присваивания = в условном выражении и многие другие вещи, которые, вероятно, требуют вашего внимания, вы увидите лампочку. Нажмите на нее или нажмите Alt+Enter, чтобы применить исправление.
Вы забыли заявление об **использовании**?



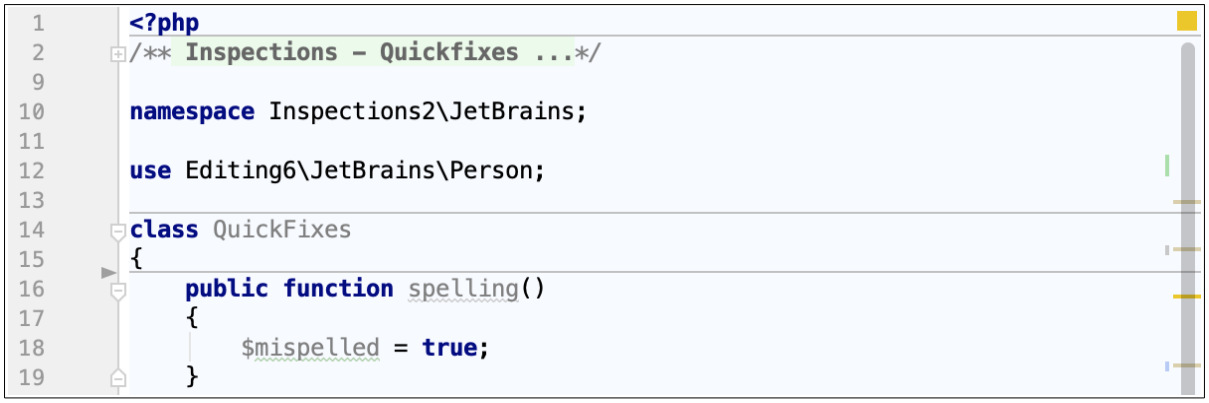
Давить Alt+Enter и нажмите **кнопку Импорт класса**:



В списке выберите класс для импорта:



PhpStorm импортирует выбранный класс и добавляет оператор **use**:

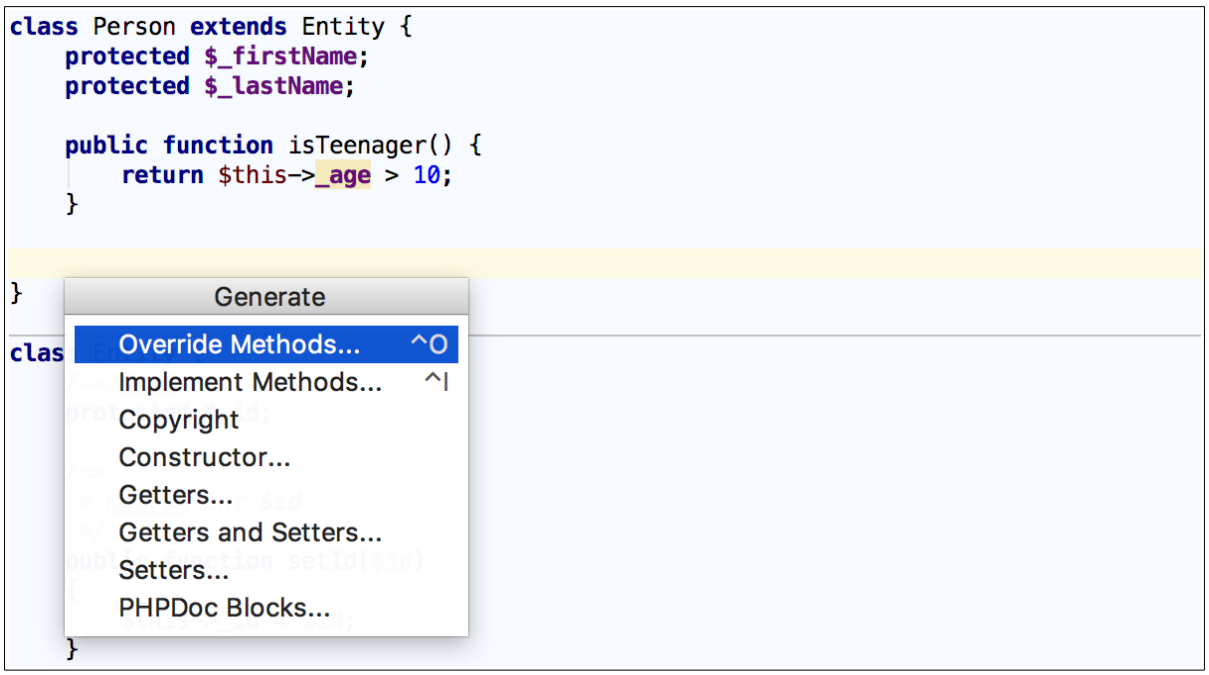


Попробуйте больше живых примеров из **workshop**.

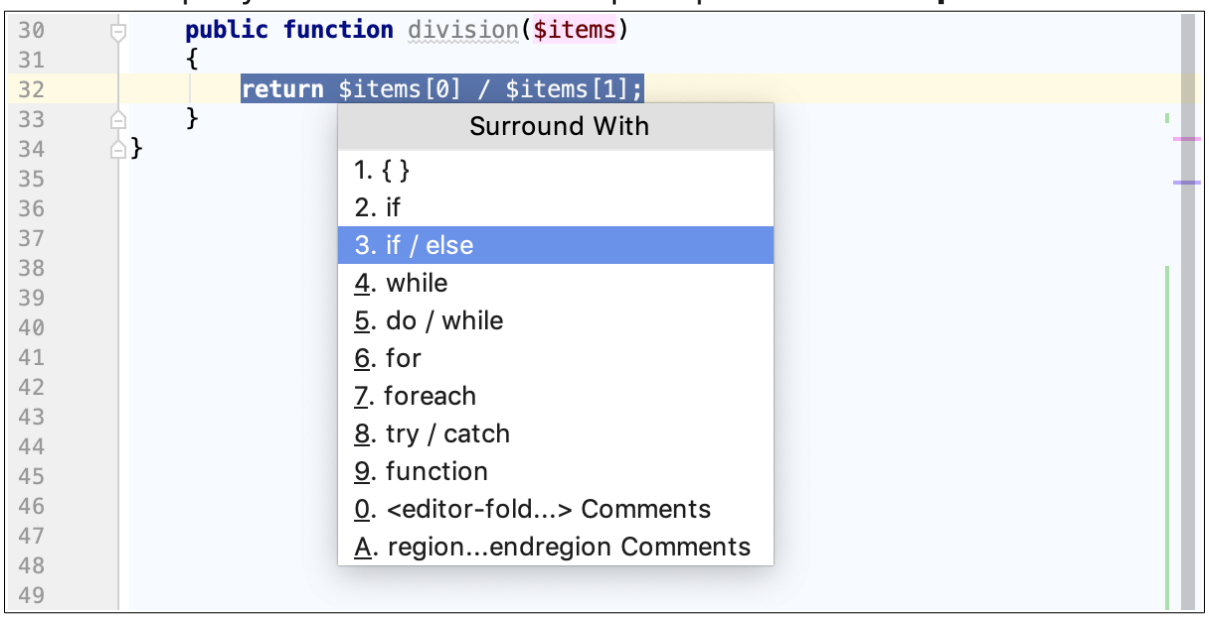
Чтобы просмотреть полный список доступных проверок, в диалоговом окне «**Настройки**» (Ctrl+Alt+S) перейти в **Редактор | Проверки**. Вы можете отключить некоторые из них или включить другие, а также настроить серьезность каждой проверки. Вы сами решаете, следует ли считать это ошибкой или просто предупреждением.

Шаг 5: Сгенерируйте код

Написание кода может быть намного проще и быстрее с опциями генерации кода, доступными в PhpStorm. **Кодекс | Сгенерировать** меню (или просто Alt+Insert) поможет вам с генерацией конструкторов, геттеров/сеттеров, комментариев PHPDoc, а также предложит переопределение/реализацию некоторых методов Ctrl+O / Ctrl+I. Попробуйте больше живых примеров в **workshop**.



Используйте **динамические шаблоны** (выберите **Код | Вставьте живой шаблон** или нажмите Ctrl+J) для создания целых конструкций кода. Чтобы просмотреть список доступных готовых к использованию динамических шаблонов, в диалоговом окне **Параметры** (Ctrl+Alt+S) перейти в **Редактор | Живые шаблоны**.
Если вы видите, что вам нужно что-то важное для вашего развития, расширьте этот набор шаблонов своими собственными. Попробуйте живые примеры из workshop/04_Live_Templates/**01_Code_Expansion**.
Кроме того, подумайте о том, чтобы окружить свой код полными конструкциями. Выберите **код | Объемное звучание** с помощью или нажмите Ctrl+Alt+T, затем выберите нужный оператор во всплывающем меню. Попробуйте больше живых примеров в **workshop**.

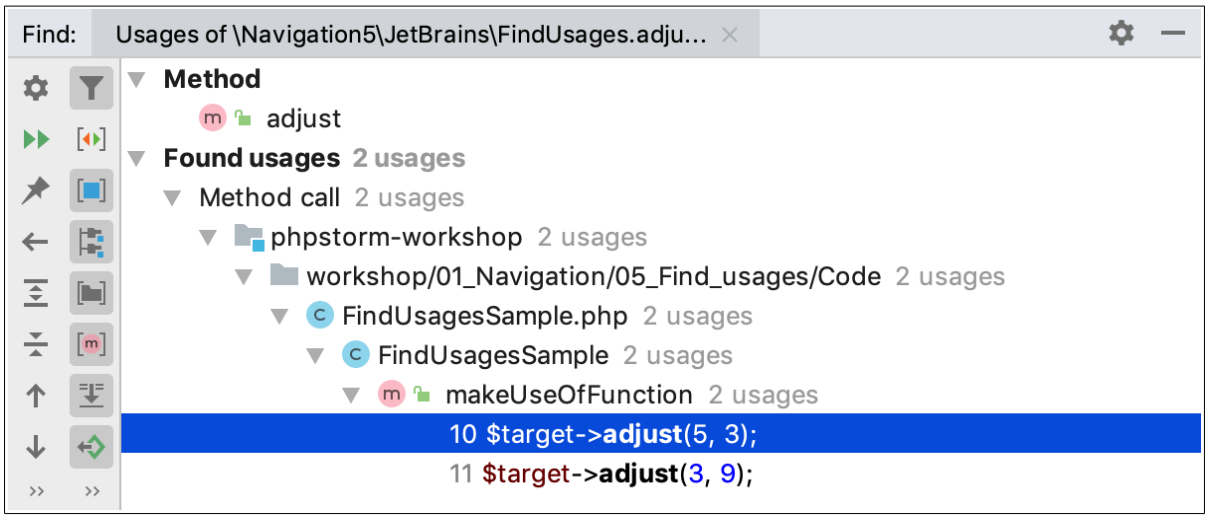


Шаг 6: Найдите свой путь

Когда ваш проект большой, или когда вам приходится работать с кодом, написанным кем-то другим, очень важно иметь возможность быстро найти то, что вы ищете, и покопаться в коде. Вот почему PhpStorm поставляется с набором функций навигации, которые помогут вам разобраться в коде.

Основной поиск

Чтобы узнать, где используется тот или иной символ в вашем проекте, PhpStorm предлагает полномасштабный поиск через Find Usages Alt+F7:



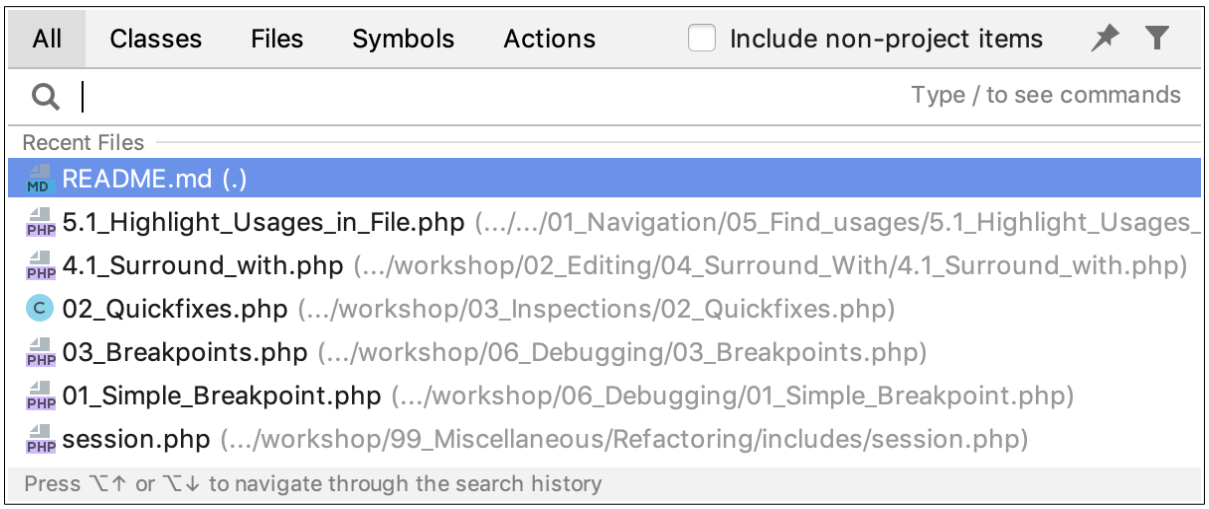
Вы также можете искать только в текущем файле Ctrl+F или в каталоге, любой произвольной области или во всем проекте Ctrl+Shift+F.

Чтобы попробовать **базовый поиск**, откройте workshop/01_Navigation/**05_Find_usages**

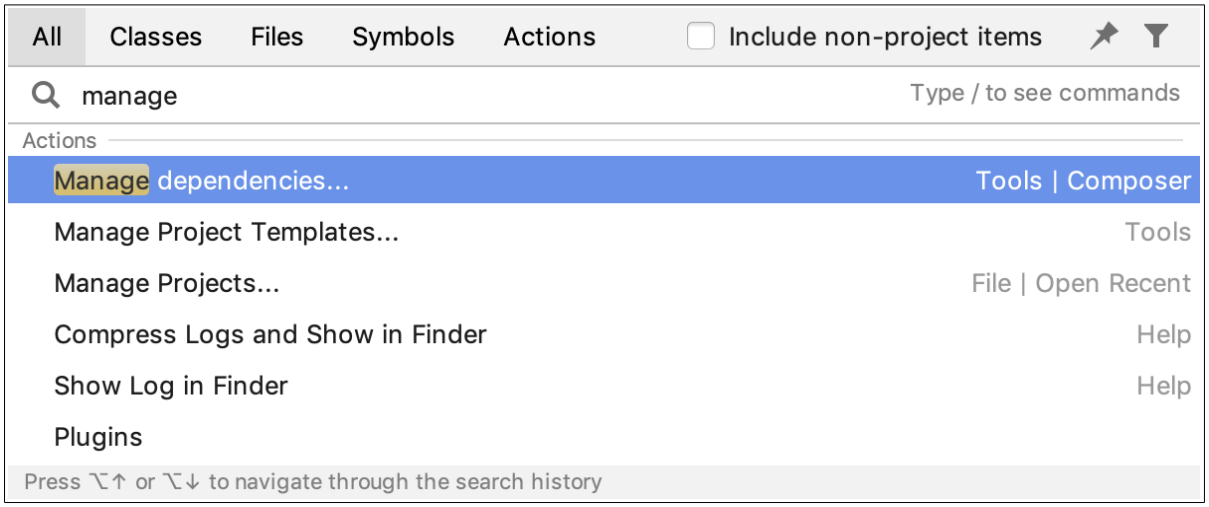
Навигация по проектам

Вы можете многое рассказать, просто взглянув на свою файловую структуру с ее импортом или иерархией вызовов, и, возможно, использовать ее для навигации по коду с помощью мощных навигационных действий.

- Search Everywhere позволяет одновременно искать классы, файлы, символы и действия меню, а также переходить к ним. Чтобы запустить **Search Everywhere**, нажмите на значок увеличительного стекла в правом верхнем углу окна PhpStorm или нажмите Shift дважды. Если вы не укажете какой-либо шаблон поиска, PhpStorm покажет список последних файлов, в которых вы можете перейти к соответствующему элементу, щелкнув по нему:



Чтобы найти что-то конкретное, введите шаблон поиска. Обратите внимание, что кроме символов, **Search Everywhere** может находить действия:



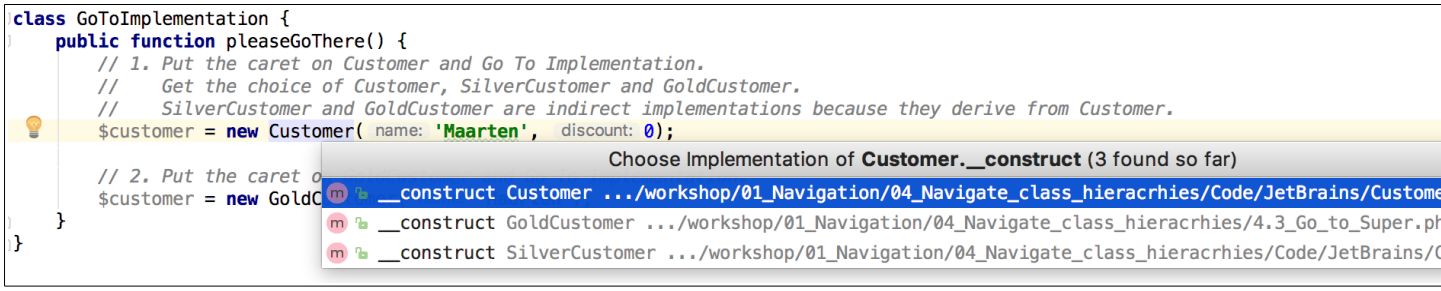
В режиме **Search Everywhere** PhpStorm даже принимает **CamelHumps** и распознает средние матчи.

Чтобы попробовать различные способы использования **Search Everywhere**, откройте **workshop** в своем проекте и следуйте инструкциям.

Конечно, навигация к **классу** Ctrl+N, **файл** Ctrl+Shift+N или **символ** Ctrl+Alt+Shift+N по его названию также в вашем распоряжении, см. Поиск везде.

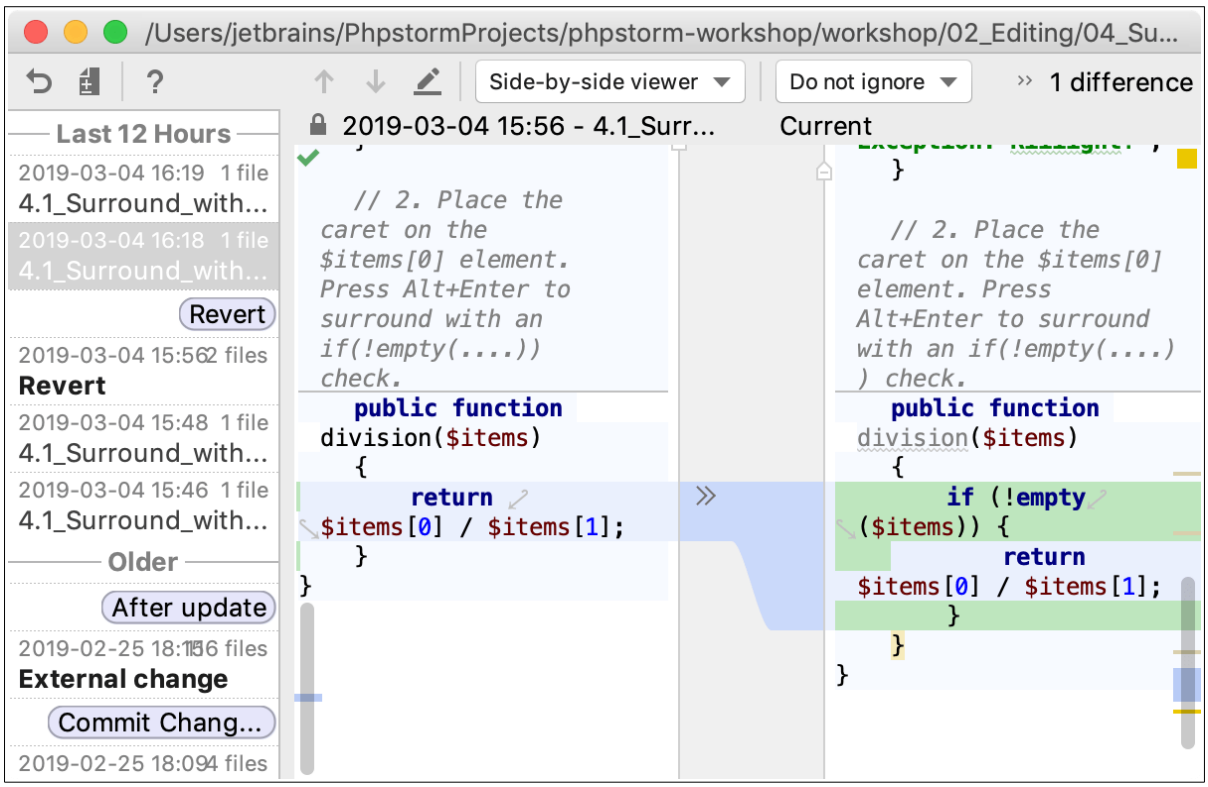
- Перейти к декларации (Ctrl+B, Ctrl+Click) приводит вас к месту, где определенный символ впервые объявлен. Этот тип навигации работает из любого места в исходном коде, даже из другого класса или комментария. Чтобы попробовать перейти **к декларации**, откройте **workshop** в своем проекте и следуйте инструкциям в комментариях.
- Перейти к реализации Ctrl+Alt+B подводит вас к реализации того или иного класса. Если реализаций несколько, PhpStorm показывает их в списке, где вы можете выбрать соответствующую для перехода. Чтобы

попробовать перейти **к реализации**, откройте **workshop** в своем проекте и следуйте инструкциям в комментариях.



Навигация по временной шкале

PhpStorm автоматически отслеживает изменения, которые вы вносите в исходный код, результаты рефакторинга и так далее в локальной истории. В отличие от традиционных систем контроля версий, **локальная история** всегда включена. Чтобы просмотреть файл или папку, выберите «**Файл**» | **Краеведение** | **Показать историю** из главного меню. Здесь вы можете просмотреть изменения, отменить их () или создать патч ( ):



Чтобы попробовать использовать **Local History** самостоятельно, откройте **workshop** в своем проекте и следуйте инструкциям в комментариях.

Шаг 7: Отладка приложения

Делает ли ваше приложение именно то, для чего оно предназначено? Если это не так, вам придется выполнить некоторую отладку, чтобы выяснить, что вызывает проблему. К счастью, в нашем примере проекта уже установлен и настроен Xdebug.

Настройка интерпретатора PHP

В нашем проекте среда Docker уже содержит интерпретатор PHP, и вам нужно только сообщить PhpStorm, где он находится.

1. В диалоговом окне «**Параметры**» (Ctrl+Alt+S), перейти на страницу PHP.
2. **Выберите PHP 7.1 с Xdebug** из списка **CLI Interpreter**.

Установка точек останова

Отладка начинается с установки точек останова, в которых выполнение программы будет приостановлено, чтобы можно было изучить данные программы. Просто щелкните желоб линии, в которой вы хотите увидеть точку останова:

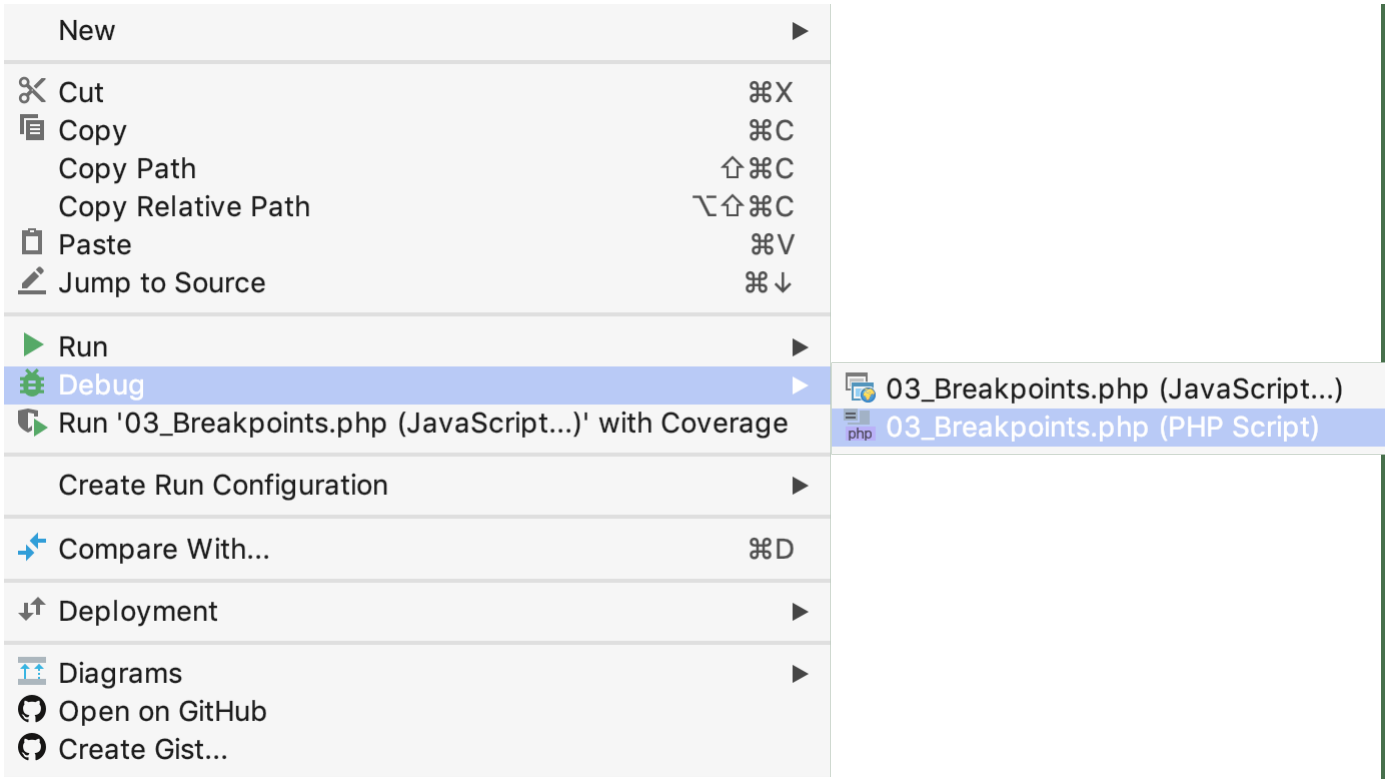
```php
$name = 'John';
$fruits = array('apple', 'pear', 'lemon');
$currentFruit = '';

foreach ($fruits as $fruit) {
    echo $name . " likes " . $fruit . "\r\n";
}
```

Чтобы поиграть с **точками останова** самостоятельно, откройте **workshop** в своем проекте и следуйте инструкциям в комментариях.

Начать отладку

В окне инструментов **Проект** выберите **workshop** и выберите в контекстном меню **Debug | 03_Breakpoints.php**:



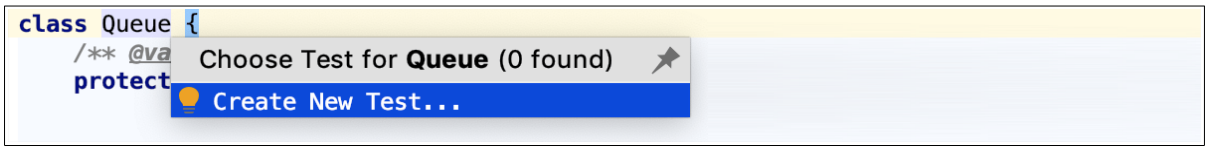
PhpStorm запускает сеанс отладки и открывает окно инструмента отладки, где вы можете пошагово выполнить приостановленную программу, просмотреть значения переменных, вычислить выражения и многое другое. Попробуйте больше живых примеров из **workshop**. Альтернативные сценарии отладки см. в разделе Отладка с нулевой конфигурацией.

Шаг 8: Проверьте свой код

PhpStorm интегрируется с самыми популярными тестовыми фреймворками PHP, такими как PHPUnit, Behat, PHPSpec и Codeception.

Написание тестов

Перейдите к классу для тестирования и нажмите Ctrl+Shift+T. Во всплывающем списке выберите «**Создать новый тест...**»:

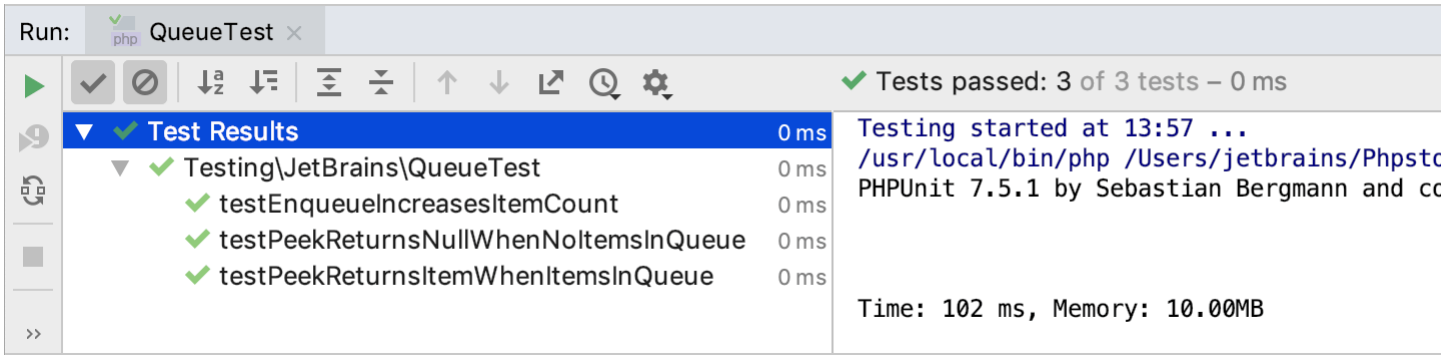


В открывшемся диалоговом окне все поля заполнены, поэтому просто нажмите **кнопку OK**. PhpStorm генерирует тестовый класс **<class_to_test>Test** и открывает его в редакторе. Напишите тесты по мере необходимости.

Чтобы попробовать писать тесты самостоятельно с помощью кодирования PhpStorm, откройте **workshop** и следуйте инструкциям в комментариях.

Запустите тест

Откройте файл с тестами в редакторе или щелкните его в окне инструментов **Проект** и выберите **Выполнить <class_to_test>Тест**. Это запустит конфигурацию запуска **PHPUnit**, которую PhpStorm сгенерировал для вас автоматически. Кроме того, можно создать конфигурацию запуска (**Run | Edit Configurations**) типа **Behat**, **PHPSpec** или **Codeception**, выберите его из списка на панели инструментов и нажмите кнопку ▶. PhpStorm показывает результаты на вкладке Test Runner в окне инструментов «Выполнить».



Шаг 9: Храните исходный код в разделе «Контроль версий»

Если вы храните свой исходный код под контролем версий, вы будете рады узнать, что PhpStorm интегрируется со многими популярными системами контроля версий: Git (или GitHub), Mercurial, Perforce и Subversion. Чтобы указать учетные данные и параметры, относящиеся к конкретной системе контроля версий, в диалоговом окне «**Параметры**» (Ctrl+Alt+S) перейдите в раздел Контроль версий.

Меню **Git** даст вам подсказку о том, какие команды доступны. Например, вы можете увидеть внесенные вами изменения, зафиксировать их, создать списки изменений и многое другое во всплывающем представлении Git (**Git | Показать журнал Git** или Alt+9).

Шаг 10: Вот и все! Вперед и развивайтесь с удовольствием!
Мы надеемся, что этот краткий обзор основных функций PhpStorm даст вам быстрый старт. Есть много важных функций, которые делают жизнь разработчика проще и веселее, а их исходный код аккуратнее и чище. Сделайте эти первые несколько шагов сейчас, а затем копните глубже, когда почувствуете, что пришло время. Наслаждайтесь PhpStorm! С любыми вопросами посетите наш дискуссионный форум PhpStorm, твиттер и блог, где вы можете найти новости, обновления, а также полезные советы и рекомендации. Кроме того, не стесняйтесь сообщать о любых проблемах в нашу службу поддержки) или в систему отслеживания проблем PhpStorm.