

Эллеонора Керри

41 уровень



02.12.2019



62441



19

Java. План действий

Статья из группы Java Developer

44944 участника

Вы в группе

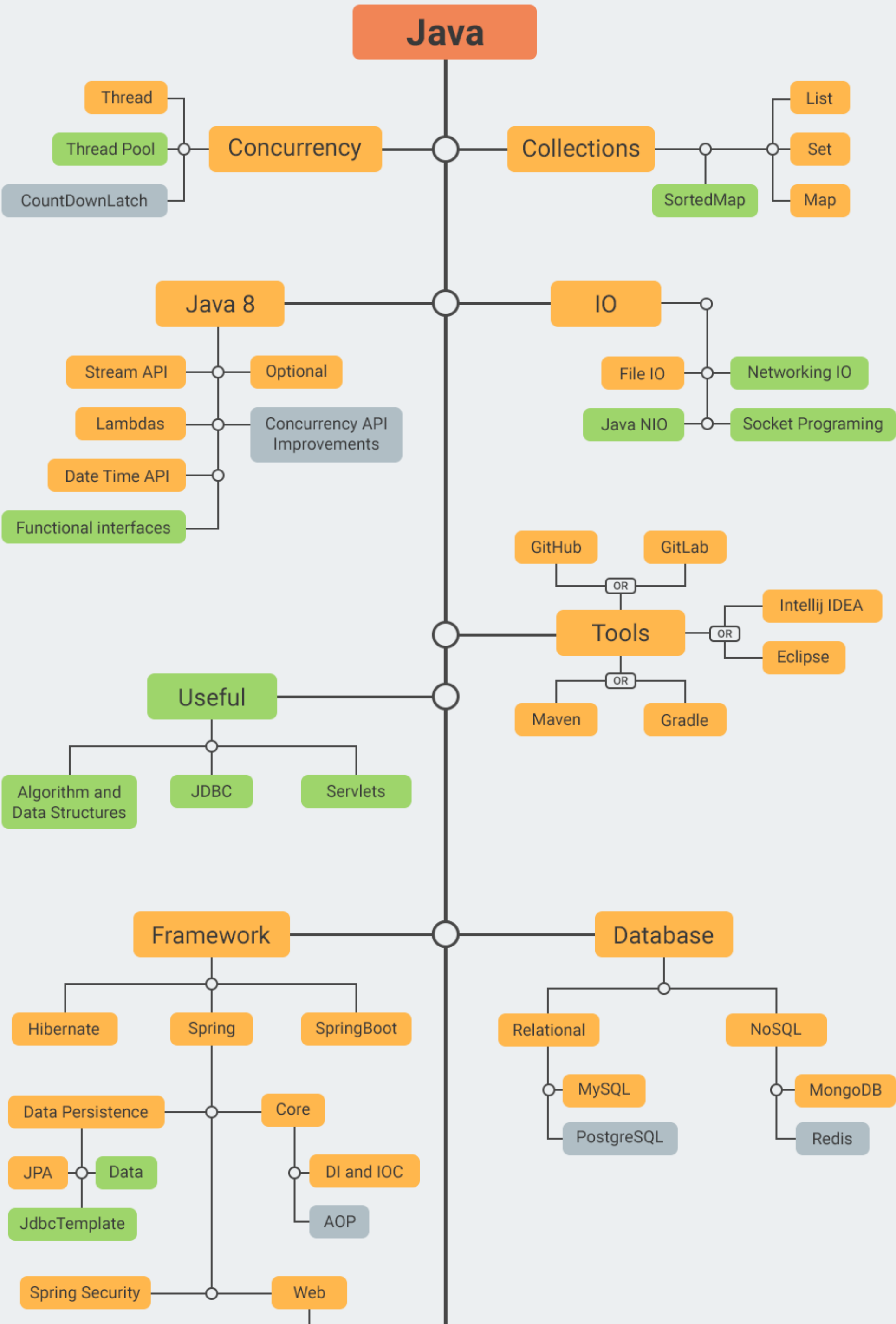
План развития Java - разработчика

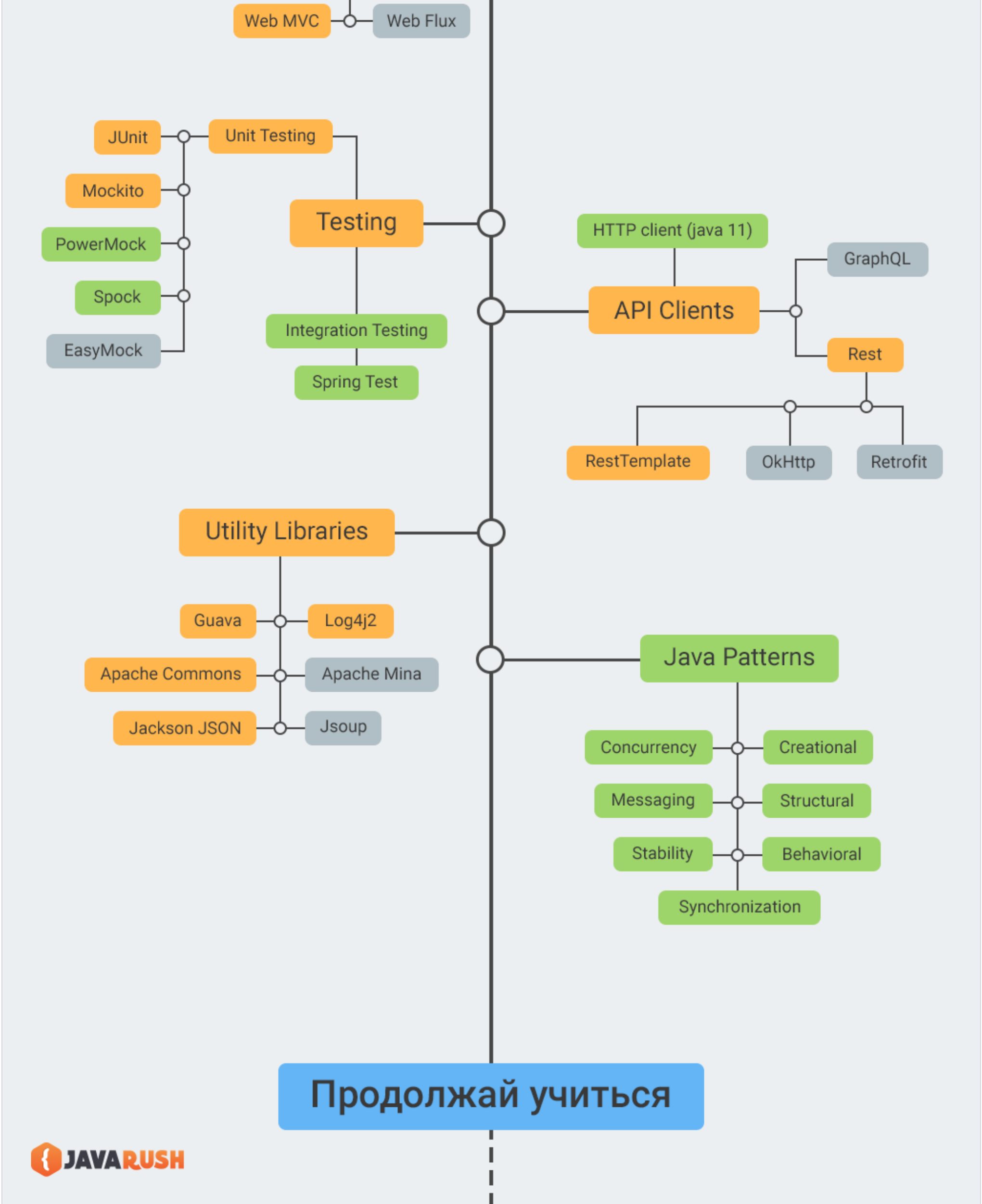
На следующий год после курса JavaRush

Обязательно

Будет плюсом

На будущее





Содержание:

- 0. [Раздел Zero — Java Core](#)
- 1. [Инструменты](#)
 - [IDE или интегрированная среда разработки](#)
 - [Инструменты для автоматической сборки](#)
 - [Системы контроля версий и сервисы онлайн-хостинга](#)
- 2. [JDK API](#)
 - [Java Collections Framework](#)
 - [Java Concurrency API](#)
 - [Java I/O API](#)

- [Устройство класса Object](#)
3. [Новые возможности Java 8](#)
 4. [SQL, базы данных, JDBC](#)
 5. [Фреймворки](#)
 - [Spring Framework](#)
 - [Hibernate](#)
 - [Spring MVC](#)
 - [Spring Boot](#)
 6. [Библиотеки и фреймворки для тестирования](#)
 7. [Сервисные библиотеки](#)
 - [Библиотеки для логирования](#)
 - [Библиотеки для JSON](#)
 - [Google Guava](#)
 - [Apache Commons](#)
 8. [API-клиенты](#)
 9. [Паттерны проектирования](#)
 10. [Дополнительные знания](#)
 - [Алгоритмы и структуры данных](#)
 - [Сервлеты](#)
 - [HTML и CSS](#)
 - [XML](#)
 - [JavaScript](#)

Что должен знать потенциальный Java Junior, чтобы получить первую работу или хотя бы претендовать на должность Trainee в хорошей компании? Какие инструменты помогут Java-программисту выйти на новый уровень? Какие технологии изучить, а какие — оставить на потом?

Стандартного ответа на эти вопросы нет, как не существует и единого плана действий, который подошёл бы абсолютно всем. Одни компании стремятся к развитию, постоянно внедряют новые технологии и тестируют возможности новых версий языка, другие упорно держатся за старые. Есть и “серединные” варианты, и, пожалуй, таких больше всего.

Тем не менее, мы составили план действий или Roadmap для начинающего Java-разработчика. В стремлении сделать его максимально простым, мы указали только те технологии и темы, которые необходимы *подавляющему большинству* “джавистов”. Следует помнить, что не всё нужно изучать досконально (кое-что из приведённого удастся освоить только в процессе работы в команде), но иметь общее представление о них — не помешает.

0. Раздел Zero — Java Core

Нулевой раздел мы вставили в статью на всякий случай — вдруг сюда заглянет человек, который только планирует изучать Java и не знает, с чего начать. Java Core — то, что даже новичок должен знать очень хорошо. То есть знать базовые вещи, ориентироваться, что предлагает язык для решения той или иной задачи и в простых случаях уметь применить эти знания. Попрактиковаться в Java Core можно на JavaRush, и если вы ещё этого не сделали — приглашаем пройти [курс](#)! Ну а всем остальным напомним главные вехи Java Core:

- Основные конструкции, операторы и типы данных Java
- ООП и его реализация в Java
- Исключения
- Java Collections
- Дженерики
- Многопоточность

1. Инструменты

IDE или интегрированная среда разработки

Главный инструмент современного разработчика — это IDE. Сегодня на рынке их очень много, однако в профессиональной Java-разработке обычно фигурируют только два имени. Это построенная на плагинах бесплатная [Eclipse](#), которая много лет подряд удерживала пальму первенства, и [IntelliJ IDEA](#), в последние годы активно вытесняющая Eclipse, и это невзирая на то, что подписка на нужную профессионалам Ultimate-версию стоит денег. Напомним, в курсе JavaRush мы пользуемся бесплатной редакцией Community IntelliJ IDEA, у которой есть определённые функциональные ограничения по сравнению с Ultimate.

Фраза “я знаю IDE” означает, что вы ориентируетесь в основных возможностях среды разработки, умеете компилировать, запускать, отлаживать и тестировать файлы, рефакторить код. Неплохим подспорьем в ускорении работы будет освоение горячих клавиш. Не поленитесь, уделите несколько часов разбору возможностей IDE, о которых вы не знали, и начинайте использовать их на практике. И не пренебрегайте отладкой, это очень полезное умение. Все эти действия помогут существенно улучшить скорость и качество вашей работы.

- [IntelliJ IDEA и Debug: не дайвинг, но снорклинг](#)

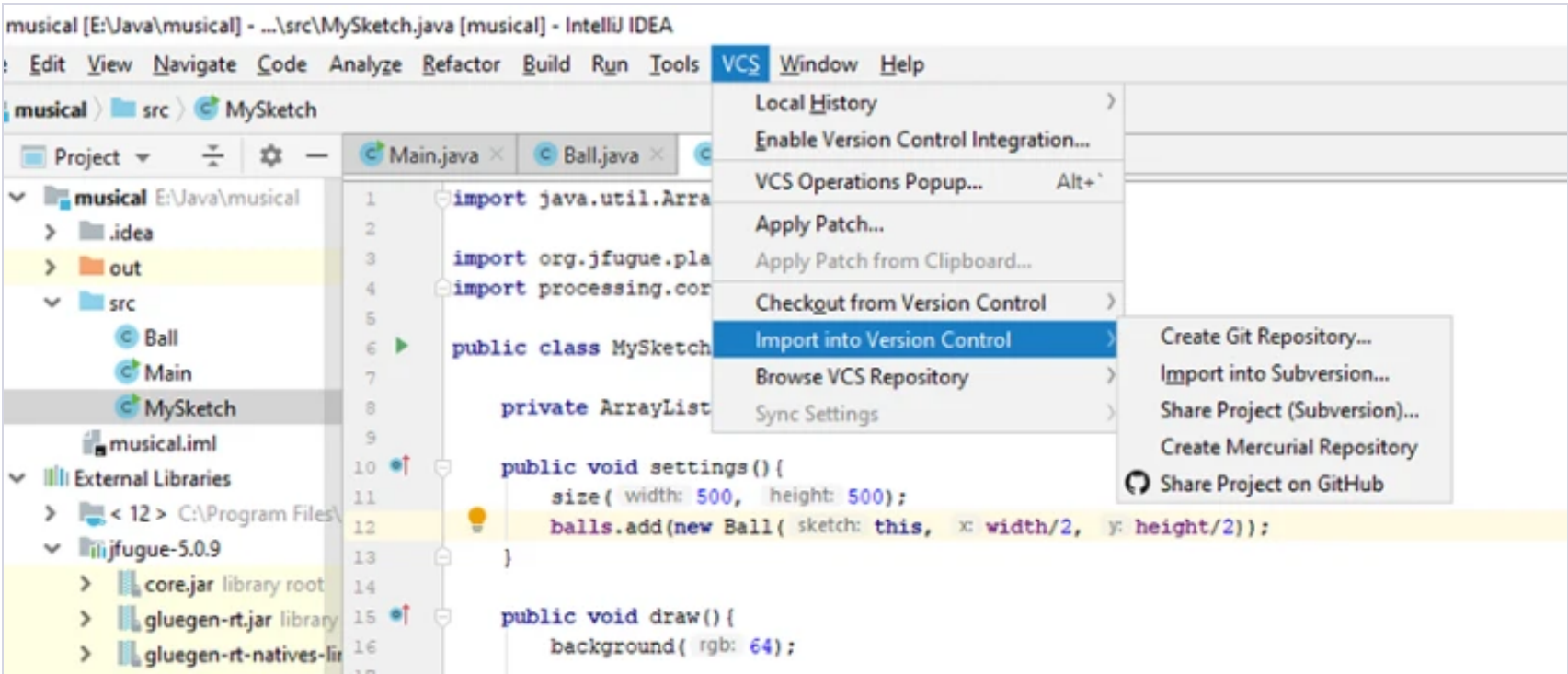
Инструменты для автоматической сборки

Сегодня в Java-проектах чаще всего используют такие инструменты как Maven и Gradle. Не обязательно изучать их досконально, но будет полезно разобраться, чем они отличаются друг от друга, на чём основаны, что такое task’и (в Gradle) и фазы с целями в Maven. Будет достаточно почитать о системах и развернуть на них пару-тройку небольших проектов. Сделать это довольно просто, а с подробностями разберётесь уже в условиях реальной работы.

Системы контроля версий и сервисы онлайн-хостинга

Система контроля версий — это то, что помогает программистам работать в команде над общим проектом, не “поломав” его, синхронизировать разрозненные куски кода, сделанные разными людьми, откатывать неудачные обновления и добавлять новые. Наиболее распространены две системы контроля версий. Одна из них — распределённая и называется Git, вторая — централизованная, по имени SVN (она же — Subversion). На сегодняшний день стандартом де-факто является Git. Работать с этой системой удобнее и легче, она поддерживается всеми IDE (впрочем, как и SVN). Работу с Git можно опробовать быстро и просто, благо, в сети на эту тему — масса информации. Например, доступный на русском интерактивный учебник [GitHowTo](#) (проходится очень быстро).

Начинающему разработчику очень важно освоить сервисы онлайн-хостинга для систем контроля версий. Чаще всего они базируются на Git и их называют Git-платформами (хотя некоторые из них умеют работать с разными системами контроля версий). Самая популярная из них — GitHub. Также довольно распространены BitBucket и GitLab. Эти системы помогают хранить и извлекать код, а ещё — делать то, что умеет Git, только не через командную строку, а через интерфейс. Также GitHub позволяет проверять код, предлагать решения проблем непосредственно на сайте. Там же можно найти какой-нибудь чужой открытый проект и попробовать предложить свои решения по его улучшению. По сути, GitHub — своеобразная социальная сеть для разработчиков. Так что если вы ещё этого не сделали, обязательно создайте аккаунт на GitHub и размещайте там свои проекты. Также почитайте о GitLab и BitBucket, а если будет время — можно и опробовать, у них есть бесплатные версии. К слову, все эти платформы полностью интегрированы с современными IDE.



2. JDK API

В этом разделе выделены те JDK API, которые современному Java-разработчику нужно знать достаточно уверенно. Программисту не мешает время от времени заглядывать в исходный код этих библиотек, ориентироваться в них, понимать, когда их нужно применять и почему. Побочный эффект: если вы хорошо ориентируетесь в этих API, скорее всего вам будет гораздо проще успешно пройти собеседование.

Java Collections Framework

Java Collection Framework — один из важнейших API языка Java, его должен знать каждый разработчик. Он представляет собой иерархию интерфейсов и реализаций стандартных структур данных в Java, таких как список, связанный список, множество, стек, очередь, хэш-таблица и многих других.

Разработчик должен хорошо разбираться в классах ArrayList, HashMap, HashSet, LinkedHashSet, TreeSet и прочих, знать об их свойствах. В частности, нужно понимать, какие затраты по времени и памяти у той или иной коллекции идут на стандартные операции (индекс, поиск, вставка, удаление) и исходя из этого правильно применять их в своих проектах.

Коллекции в Java реализованы очень хорошо, но если есть необходимость разработчик может предложить собственную реализацию. Программист, который отлично разбирается в коллекциях, может расширить или переопределить логику в уже написанных классах или же реализовать всё с нуля.

Java Concurrency API

Java изначально разработана с поддержкой параллельного программирования, а начиная с версии 5.0, язык включает высокоуровневые API для параллельных потоков. Так что грамотный Java-разработчик просто обязан хорошо разбираться в многопоточности и иметь представление об основных API из пакетов [java.util.concurrent.*](#).

Как минимум, нужно знать и чётко осознавать, что такое Thread, Runnable, блокировка объектов и синхронизация. Обязательно разберитесь в концепциях deadlock, livelock, состоянии гонки и что со всем этим делать.

Чтобы чувствовать себя уверенно, изучите синхронизаторы из java.util.concurrent.*, например, Semaphore, CyclicBarrier, CountdownLatch, Phaser, Exchanger<V>, CompleteableFuture и так далее. А ещё — интерфейсы Callable и Future.

Java I/O API

Начинающие разработчики частенько игнорируют углублённое изучение [Java I/O](#) и [Java Non-blocking I/O](#). А зря: эти Java API облегчают работу с потоками и регулярно используются в реальных приложениях. Особенно такие классы, как File, InputStream, OutputStream, Reader и Writer из пакета java.io, который является ядром Java IO API.

Java Non-blocking I/O (java.nio) представляет собой коллекцию прикладных программных интерфейсов, предназначенных для реализации высокопроизводительных операций ввода-вывода. К ним относятся, в частности, ByteBuffer, FileChannel и Selector и другие. Потрудитесь разобраться с этими API, не пожалеете.

- [Ввод-вывод в Java. Классы FileInputStream, FileOutputStream, BufferedInputStream](#)
- [Считывание с клавиатуры — ридеры](#)

Устройство класса Object

Понимаешь суперкласс Object — в каком-то смысле становишься “носителем языка Java”, гораздо лучше осведомлённым о структуре ООП и многих процессах. Класс java.lang.Object, лежит на самой вершине иерархии классов. Помимо лучшего понимания всего происходящего, знание методов класса существенно облегчит прохождение собеседований — интервьюеры просто обожают проверять кандидатов на прочность с помощью класса Object и его объектов.

- [Class Object docs](#)

Научитесь программировать с нуля с JavaRush:

1200 задач, автопроверка решения и стиля кода

[НАЧАТЬ ОБУЧЕНИЕ](#)

3. Новые возможности Java 8

Невзирая на то, что после выхода Java 8 прошло время, и уже появились и другие номерные обновления, именно восьмая версия стала знаковой. В ней появились важные новшества, которые упрощают, а в каком-то смысле и меняют подходы к программированию на Java. Вы должны разобраться, как использовать лямбда-выражения, а также со Stream API в Java 8 и новыми API даты и времени.

- [Java 8 docs](#)
- [Java 8 Tutorial](#)
- [Популярно о лямбда-выражениях с примерами и задачами](#)

4. SQL, базы данных, JDBC

Мало какой Java-разработчик не сталкивается в своей работе с SQL-запросами и базами данных. Поэтому важно понимать, что такое SQL и реляционные базы данных, как они работают, уметь писать простые запросы на join двух таблиц. Для тренировки можно попробовать поработать с одной из СУБД, например, [PostgreSQL](#) или [MySQL](#). Также было бы неплохо получить начальные знания о нереляционных базах данных, о подходах noSQL и поверхностно ознакомиться с документоориентированной СУБД [MongoDB](#).

Для работы с базами данных в чистой Java можно использовать стандарт **JDBC** вместе с одноименным API. Он реализован в виде пакета java.sql, который входит в JDK. В чистом виде сегодня им пользуются редко, однако его часто можно встретить в старых приложениях для поддержки, да и в основе более современных и общепринятых средств зачастую лежит именно этот стандарт.

- [Java JDBC API](#)
- [JDBC или с чего всё начинается](#)
- [Часть 1. Введение в SQL](#)

5. Фреймворки

Среди требований к Junior Java Developer сегодня всё чаще можно встретить “знание Spring, Hibernate, Spring Boot”. Изучить эти технологии самостоятельно — задача очень непростая, но, тем не менее, это возможно, особенно на поверхностном уровне. Более глубокое понимание придёт во время работы. Итак.

[Spring Framework](#)

Практически каждое приложение, создаваемое на Java в наши дни использует Spring Framework. Этот мощный фреймворк предоставляет определённую систему координат, костяк, с помощью которого строится приложение. Spring-приложение гораздо проще тестировать и поддерживать. А всё благодаря внедрению зависимостей.

- [Spring для ленивых. Основы, базовые концепции и примеры с кодом](#)

[Hibernate](#)

Ещё один важнейший для Java-разработчиков фреймворк — Hibernate. Он реализует спецификацию JPA (Java Persistence API), с помощью которой решаются задачи объектно-реляционного отображения (ORM). Большинство Java-приложений взаимодействуют с базами данных, и, если речь идёт о реляционных базах данных, работать с ними без Hibernate неудобно. Этот фреймворк предоставляет разработчикам ряд важных функций, в частности, кэширование и транзакции из коробки, что в свою очередь позволяет сосредоточить усилия на разработке логики приложения и освобождает программиста от множества низкоуровневых задач при работе с реляционными базами данных. Это существенно повышает производительность разработчиков.

- [Ваше первое приложение на Hibernate](#)

[Spring MVC](#)

Этот фреймворк обеспечивает разработку приложения согласно паттерну Model — View — Controller, применяя слабо связанные готовые компоненты. Изучите этот паттерн (о паттернах проектирования речь идёт чуть ниже) и логику работы Spring MVC. На практике его используют довольно часто.

[Spring Boot](#)

При надлежащем умении Spring упрощает создание приложения Java. В свою очередь Spring Boot упрощает создание Java-приложения на основе Spring. Spring Boot позволяет вам легко создавать полноценные Spring-приложения уровня Enterprise,

которые можно запустить с минимальными усилиями: автоконфигурация устраняет большинство проблем, связанных с настройкой Spring-приложений.

- [Знакомство с Maven, Spring, MySQL, Hibernate и первое CRUD-приложение](#)

6. Библиотеки и фреймворки для тестирования

Некоторые будущие разработчики уверены, что тестирование кода — забота вовсе не их, а специальных людей, которые так и называются — тестировщики. На практике дело обстоит не совсем так. Тестирование, особенно модульное (его чаще всего называют unit-тестированием), — очень важный навык для каждого программиста. Мало того, только что приступившим к исполнению обязанностей новичкам довольно часто поручают покрыть unit-тестами чей-то код. Так что настоятельно рекомендуем изучить библиотеку JUnit и взрастить привычку писать unit-тесты для своего кода. Также обратите внимание на фреймворк Mockito, который можно использовать совместно с JUnit для создания фиктивных классов-зависимостей.

- [JUnit docs](#)
- [О JUnit, часть 1](#)

7. Сервисные библиотеки

В Java есть огромное количество сервисных библиотек, которые помогают решить практически любые задачи, стоящие перед разработчиком. Изучить их все невозможно, да и смысла особого в этом нет. А вот ориентироваться в них — отличная идея. Здесь выделим лишь несколько из тех, которые очень часто используются на практике.

Библиотеки для логирования

В первую очередь можно упомянуть [log4j](#) и [Slf4j](#). Эти библиотеки разработаны для сокрытия реализации рутинных операций по логированию событий, которые происходят во время работы Java-приложений.

Библиотеки для JSON

JSON, формат передачи информации от клиента к серверу, сегодня используется чаще всего. Есть несколько хороших библиотек, которые работают с JSON, самые популярные — [Jackson](#) и [google-gson](#).

- [Введение в Jackson Framework](#)

Google Guava

[Guava](#) — проект с основными библиотеками для Java, разработанными Google. Здесь можно найти новые типы коллекций (multimap, multiset и другие), неизменяемые коллекции, графы, функциональные, утилиты для параллелизма, ввода/вывода, хэширования, обработки строк и многое другое.

Apache Commons

[Commons](#) — огромный проект, содержащий множество полезных Java-утилит самого разного предназначения. Так, библиотеки Apache Commons лежат в основе Tomcat, Hibernate и ряда других крупных проектов. Библиотек в Apache Commons очень много. Упомянём Commons IO, которая упрощает выполнения операций ввода-вывода, Commons CSV для работы с csv-файлами, Commons Math для работы со сложными математическими и статистическими операциями и вычислениями, Commons CLI для анализа аргументов командной строки.

8. API-клиенты

REST — стиль именования эндпоинтов для доступа к ресурсам по сети в человекочитаемом формате. Современному Java-разработчику лучше ориентироваться в идеологии REST, а также знать **Spring RestTemplate** — очень полезную библиотеку для создания REST-клиента.

- [RestTemplate docs](#)

9. Паттерны проектирования

Если начинающий разработчик знаком с паттернами (шаблонами) проектирования, то есть правилами хорошего тона в Java-программировании, да ещё и умеет применять их на практике, он мгновенно повышает свою стоимость на рынке труда. Новички часто недооценивают паттерны, поскольку во время учёбы редко создаются сложные приложения. Тем не менее, если не применять паттерны на серьезных проектах, поддержка и адаптация кода становится чрезвычайно сложной задачей.

Так что не поленитесь, изучите паттерны и применяйте их в персональных проектах. Ваш будущий работодатель будет за это очень благодарен.

- [Паттерны проектирования в Java](#)
- [Паттерны проектирования, часть 2](#)
- [Паттерны проектирования: Singleton](#)
- [Паттерны проектирования: FactoryMethod](#)
- [Паттерны проектирования: AbstractFactory](#)
- [Паттерн проектирования “Стратегия”](#)

10. Дополнительные знания

Алгоритмы и структуры данных

“Алгоритмы и структуры данных” — это название целого курса, читаемого в технических вузах. В нём раскрываются теоретические основы построения разных структур данных. А на практических занятиях с ними учатся работать — класть и извлекать данные, искать и сортировать их. Собственно, под “Алгоритмами” в таком словосочетании понимают именно сортировку и поиск. За долгие годы специалисты в области Computer Science разработали множество алгоритмов. Некоторые из них носят учебный характер, так как при относительной простоте реализации они не слишком эффективны в работе. Например, работают медленно, что может быть ощутимо на больших пулах данных. Или же потребляют много памяти. Другие алгоритмы оказались очень эффективны. Настолько, что их внесли в официальные библиотеки большинства языков программирования. Соответственно, сегодня не обязательно разрабатывать такие алгоритмы самостоятельно. Достаточно знать, где они лежат.

Твой Java-дайджест

Полезные статьи о Java: обучение, программирование, карьера

Введите свой e-mail

И тем не менее большинство опытных разработчиков рекомендует новичкам проходить “школу алгоритмов” — реализовывать их самостоятельно во время учёбы. Это развивает программистское мышление. А ещё — помогает проходить собеседования, там очень любят задавать задачки на сортировку и поиск.

- [Алгоритмы сортировки в теории и на практике](#)

Сервлеты

Сервлет — это способ обработки запроса пользователя. Их сегодня используют не везде и не всегда, но получить о них представление будет полезно.

- [Ваше первое приложение с использованием Java-сервлетов](#)

HTML и CSS

Основы верстки должны знать все. Эти знания получить довольно просто, и если вы вдруг ещё этого не сделали, уделите этому занятию пару дней. Заодно отдохнёте от более сложных тем.

XML

Расширяемый язык разметки раньше повсеместно применялся в Java-разработке. Его постепенно вытесняет JSON, но XML встречается и сегодня. Изучать его несложно, так что можно уделить этому языку толику внимания.

- [Что такое XML?](#)

JavaScript

Опросы разработчиков показывают, что даже те из них, кто не имеет абсолютно никакого отношения к фронтенд-разработке, время от времени писали скрипты на JavaScript. Знание основ этого языка можно считать правилом хорошего тона, поэтому не поленитесь, почитайте о нём и создайте десяток-другой скриптов. Лишним не будет.

 +564 

Комментарии (19)

популярные новые старые

JavaCoder

Введите текст комментария

Andrey Karelin Уровень 41, Сумы, Украина 28 мая, 17:42 

А нельзя было бы вот это все с постепенным погружением по сложности давать с 30го по 40й уровни. А то напихали очередных JavaСог-ских задачек на алгоритмы...

Ответить  0 

Алексей Уровень 41, Чебоксары, Россия 21 февраля 2021, 00:21 

Какие замечательные статьи открываются с прохождением 40-го уровня. Аж глаза разбегаются, с чего бы начать.

Ответить  +1 

wan-derer.ru Уровень 40, Москва, Россия 30 сентября 2020, 12:48 

Писос какой-то... Зачем я в это ввязался?! сидел бы спокойно у себя в ларьке с шаурмой...

Ответить  +25 

Антон Шеленков Уровень 36, Москва, Россия 14 сентября 2020, 14:21 

Многое в точку, но по моему мнению не все. Стадию джуна я прошел, поэтому поделюсь мыслями о том, что бы я изменил в этом плане развития, чтоб максимально сократить время обучения до первой работы (и при этом не оставить сильных пробелов в знаниях):
0. О всех вещах, перечисленных на картинке надо иметь как минимум представление на уровне "что это такое и какую функцию выполняет"
1. Если уж идет речь о многопоточности, то знать ThreadPool, Executor, Callable, Locks - must have. Это то, с чем в реальности придется иметь дело.
2. Где теоретический минимум по сетям и протоколам? Надо знать модель OSI, HTTP (желательно еще

- понимать HTTPS), REST.
3. IntelliJ IDEA vs Eclipse - однозначно IntelliJ IDEA. Желательно освоить Ultimate Edition (можно ее купить или активировать по другому, если вы понимаете о чем я)
4. Maven vs Gradle - Maven легче и по ощущениям более распространен, чем Gradle. Имхо нет смысла начинать с Gradle
5. Алгоритмы и структуры - надо знать в основном структуры (стек, очередь, список, деревья) и сложность операций с ними. Всякие сортировки и более прошаренные структуры можно доучить потом.
6. NIO - все руки не доходят раскурить этот API:) Вряд ли понадобится в 99% случаев. Я б оставил на потом
7. Сервлеты - must have, JDBC стоит посмотреть мельком ради академического интереса, но вот учить API и как-то им грузиться точно лишнее - он слишком уж на низком уровне
8. Spring Data JPA поверх Hibernate. Изучение азов займет немного времени, но стоит того
9. MySQL - однозначно нафиг. Стоит начинать с PostgreSQL - он сложнее, но при этом намного более распространен, чем MySQL
10. NoSQL - лезть туда джуну далеко не обязательно. Больше выхлопа будет, если это время потратить на другие вещи

Ответить

+41

Антон Шеленков

Уровень 36, Москва, Россия

14 сентября 2020, 14:21

...

11. Spring Security - очень сложная вещь. Узнать только поверхностно, сделать по примеру аутентификацию и авторизацию на домашнем проекте и не копать глубже. Иначе точно зареетесь. Лучше подтянуть Spring Security уже в процессе работы
12. PowerMock и Spock я бы не трогал - лучше посмотреть в сторону Spring Testing
13. Вместо Log4j2 - Logback
14. Guava, Apache Commons можно пробежаться глазами и мб поискать подборки самых используемых классов и методов. Эти либы очень большие, большинство методов, что читаете, сразу же забудете. Самые полезные их классы проще подхватить на работе

Ответить

+25

Justinian

Judge в Mega City One MASTER

14 марта 2021, 16:55

...

Все грамотно, со всем согласен, только свои комментарии отмечу:

7. И сервлеты и JDBC это низкоуровневые АПИ.
JDBC API, в отличие от тех же сервлетов, это не шибко объемный и сложный для новичка АПИ, в той мере, в котором новички его изучают. Но JDBC необходимо знать, это основа фундамента работы с базами данных, а работа с базами данных, для десктоп бекендера по крайней мере, это приоритет №1.
На этом этапе достаточно уметь написать, протестировать CRUD на JDBC, уметь без гугля и подсказок, открыть идейку, создать проект и подключиться к базе данных. Понимать цепочку шагов, которые нужно сделать, чтобы с голого проекта, получить данные с БД или записать их. Знать ответы на основные вопросы собеседов (гуглится jdbc interview questions).
Это минимум, который очень важен для новичка, важнее наверное половины пунктов так точно.

9. MySQL - однозначно нафиг. Стоит начинать с PostgreSQL - он сложнее, но при этом намного более распространен, чем MySQL

MySQL к PostgreSQL в плане популярности относятся как Мавен к Градлу наверное, при том что тенденции последних лет 4-5 очевидны, Градл любят и предпочитают, Постгрес очень заходит девкоммюнити, но и Мавен и MySQL продолжают повсеместно использовать и они держат внушительный кусок рынка.

Новичку про это думать не нужно, для него главное что на этапе обучения между этими двумя БД для него не будет почти никакой разницы, синтаксис и принцип работы одинаков. А ключевые отличия, фишки он ознакомится уже на работе, поэтому на этапе обучения можно использовать любую, какая понравится.

13. Вместо Log4j2 - Logback

На этапе обучения разницы нету, надо знать о логгировании в джаве в принципе, теорию, ответы на вопросы с собеседов как контрольный маркер, и уметь использовать логгирование в принципе, возможно немного поигравшись что, как и куда выводить.

На реальном проекте попасть может что угодно.
Понять нюансы, разницу между ними, что когда лучше использовать, без опыта на продакшене имхо сложно

Ответить

+5

Андрей Пушкин

Уровень 40, Россия

27 мая 2020, 09:22

...

Да, а от Java Script плавно переходите к React и Redux. Можно любые соседние брать - тоже хорошо. Может так получится, что отдельных UI-щиков нет и ты должен писать и бэкэнд, и фронтенд. А иногда и в базу залезать :) Короче, все надо уметь, но специализацию выбрать - к чему более тяготеешь. Спасибо за статью, товарищи!

Ответить

+6

Александр

Java Developer в ЕРАМ

9 января 2020, 12:22

...

Хорошая статья, особенно инфографика.

Ответить

+5

TheVoda

Уровень 35, Воронеж, Россия

6 декабря 2019, 05:09

...

Спасибо огромное, супер статья. Сколько бы не учился, а куда не заглянешь - везде эльфийский, а вы тут как раз весь эльфийский алфавит выложили)

Ответить

+10

Egor

Уровень 41, Санкт-Петербург, Россия

3 декабря 2019, 16:32

...

Господа Администраторы, было бы прикольно иметь возможность добавлять статьи в избранное, а потом просматривать/сортировать/удалять их в избранном) А то уже закладок в браузере пруд пруди)

Ответить

+

+19

+

Эллеонора Керри

Уровень 41

3 декабря 2019, 16:53

...

Здравствуйте) На сайте есть возможность добавлять статьи в закладки, нажав на иконку "В закладки"

Все материалы хранятся в разделе "Закладки" по ссылке: <https://javarush.ru/me/bookmarks>

Ответить

+

+13

+

Александр

Java Developer

4 декабря 2019, 08:41

...

Добрый день)
Ой, а они всегда были..?

Ответить

+

0

+

Эллеонора Керри

Уровень 41

4 декабря 2019, 11:15

...

Добрый день) Появились сравнительно недавно.

Ответить

+

0

+

Egor

Уровень 41, Санкт-Петербург, Россия

4 декабря 2019, 15:08

...

Крутяк, спасибо!)

Ответить

+

0

+

Ильгиз

Уровень 8, Уфа, Россия

22 января 2020, 13:16

...

кнопка была давно, но вот найти "у себя" эти закладки было невозможно, спасибо разработчикам сайта, что допилили эту функцию))

Ответить

+

+3

+

Олександр Фалендиш

Уровень 8, Харьков, Украина

5 апреля 2020, 18:44

...

добрый день. где именно эта иконка - закладки?

Ответить

+

0

+

Эллеонора Керри

Уровень 41

6 апреля 2020, 09:07

...

Здравствуйте! Переходите по ссылке <https://javarush.ru/me>, вторая вкладка будет "Закладки")

Ответить

+

0

+

Олександр Фалендиш

Уровень 8, Харьков, Украина

6 апреля 2020, 23:45

...

Спасибо. Это я видел. Как добавить в закладки?))

Ответить

+

0

+

Эллеонора Керри

Уровень 41

7 апреля 2020, 15:03

...

Ответить

+

+1

+

ОБУЧЕНИЕ


- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ

- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка

 **RUSH**

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

 Русский

▼



"Программистами не рождаются" © 2022 JavaRush