HOWTO: интервьюирование .net - разработчика

Типтюк А.А.

13.03.2009 v 01.00.0006

Аннотация

Я не великий специалист в HR и даже в нете, но исторически сложилось так, что сейчас у нас на конторе девелоперами командую я и собеседую кандидатов тоже я. Поговорив со многими кандидатами, я был несколько удивлен структурой их знаний и отсутствием у многих базовых представлений в том же ООП (СумГУ – большой привет :).

Думаю, если я перечислю традиционный набор "контактных площадок", на тестировании которых обычно строится интервью со мной – это может быть интересно как потенциальным кандидатам, так и коллегам, которым приходится заниматься аттестацией при рекрутинге.

Содержание

1 НЕТ - разработчики. Зачем ?

И в разговоре с приемлемым кандидатом, и при подготовке себя как специалиста всплывает вопрос – какие задачи будут решаться разработчиком? Поэтому начну с перечисления традиционных направлений .net-разработки, чтобы было ясно, к чему нужны конкретные знания.

С чем я сталкивался:

desktop - приложения Тут фронт работ бывает разной глубины. Вам могут поручить клепать в существующем проекте формочки, однако речь также может идти о проектировании и создании с нуля приложения с замороченой внутренней логикой, многопотоковостью (как следствие – синхронизация backgroud потоков, обеспечивающих поведение, с потоками элементов интерфейса); может понадобиться работа с БД и с чем угодно другим. В любом случае, для этой работы надо уметь проектировать формы win forms, ориентироваться в структуре приложения WinForms и быть готовым пристегнуть к этому все, чего потребует от вас конкретная задача.

работа с данными Я свято уверую в то, что учетные задачи – надежный кусок хлеба на всю оставшуюся жизнь. Сегодня человечество уже слишком зависит от своих железяк и никогда не откажется от возможности считать свои деньги, товар, ресурсы, время и бог весть что еще автоматизировано. Поэтому я предпочитаю заниматься ДБ-ориентированными задачами и высоко ценю кадры, готовые к такому роду деятельности. Работа с данными редко ведется сама по себе; обычно слой данных встроен в некий продукт, клиентский (desktop app) или серверный (сервис, консольное приложение, desktop app со специфической архитектурой).

серверные решения Подразумевает способность написать приложение без богатого интерфейса или вообще без интерфейса, которое будет в фоне выполнять требуемые действия. Традиционно, разработка подобных продуктов требует умений обращаться с коммуникациями, данными и потоками.

Вещи, нужные в разной степени, в которых я, однако, слаб:

коммуникации Сплошь и рядом требуется обеспечить передачу данных между элементами распределённой системы. Начиная с традиционного случая **клиент – TCP/IP – сервер**, и усложняя до посинения, требуется мочь нужным образом построить обмен данными между пунктами A и B.

девайсы Программирование взаимодействия с устройствами на низком уровне. Аппаратные протоколы, промышленные протоколы, шины, промэлектроника, usb, UART, ethernet, modbus, I_2C, IEEE xxxx и т.п.

тестирование В моём представлении хороший тестер должен быть программистом; делить тестеров и разработчиков на изолированные касты мне не нравится. Тестер – это просто такая вот роль в разработке. Однако, программирование тестов – отдельный специфический набор навыков.

парсеры, интерпретаторы Программирование всяческих сценариев. Создание своих средств отработки сценариев или построение требуемых продуктов на готовых решениях типа LUA[1].

экзотика GDI+, DirectX, Open GL, AI (искусственные интеллекты), image & voice recognition etc. Из всяких экзотических занятий пока по работе столкнулся только с векторным рисованием с помощью GDI+. Сам не занимался, но такой род деятельности у нас есть. Ну а вообще специфических направлений, очевидно, может быть масса.

2 И что нам таки надо?

2.1 .net, (censored), объектный

Не одно поколение программистов персональных компьютеров эффективно проработало всю свою жизнь с рудиментарными представлениями об ООП или вообще без них. Заканчивая эпохой FoxPro, VBA, 1C и Delphi включительно – разработчик имеет возможность писать код не особенно задумываясь об объектной модели, опираясь на заготовки методов, генерируемых средой; либо вообще не связываться с ООП и весь свой код решать в процедурном стиле.

Для меня ярким примером была дискуссия с коллегой, в ходе которой он пытался доказать мне, что обработчик нажатия кнопки – это метод кнопки, а не формы, на которой кнопка лежит. Показательно то, что этот человек продуктивно программирует всю жизнь; я не подвергаю никаким сомнениям его компетенцию. Просто у него не было необходимости разбираться с этим вопросом чтобы эффективно разрабатывать в visual FoxPro.

Мне кажется подобная ситуация имеет место и в средствах вроде php и perl-а.

Однако, платформы вроде java или .net всё-таки требуют с этим самым ООП разобраться. Объясняется это тем, что программы, исполняемые на этих платформах, принципиально отличаются содержанием от традиционных программных продуктов.

Традиционные средства разработки предоставляют архитектору понятие объектов и классов только на этапе проектирования. После компиляции продукта в исполнимый код получается аморфная каша из машинных кодов, ресурсов и отладочной информации, разбавленная арматурой RTTI (информация о типах, используемая при выполнении программы).

В противоположность такому подходу, .net и java генерируют промежуточный код, строго структурированный сообразно объектной модели продукта. При этом на уровне структуры исполнимого файла однозначно указывается, что некий код является методом определённого класса. Вообще говоря, эти платформы складывают в исполнимый модуль всю информацию о компоновке классов, методов, полей и свойств и позволяют практически полностью воссоздать исходный текст приложения из исполнимого варианта. С минимальными оговорками. На этих платформах невозможно написать код, который не будет иметь "классовой принадлежности", т.е. не будет ничьим методом. Даже в тех местах, где среда предоставляет декоративную возможность оформить код

как "ничей" – платформе все равно будет сгенерирован класс-заглушка, который станет средой обитания этого кода.

Все эти обстоятельства приводят к тому, что без основательного знакомства с ООП в .net-е делать нечего.

Просто перечислю вопросы, которые я практически всегда задаю:

- Что такое виртуальный метод?
- Что такое абстрактный метод?
- Что такое статические члены и методы класса? (в Delphi это процедуры и функции объявленные с модификатором **class**)
- Зачем нужны конструкторы и деструкторы?
- Что такое метод?
- Что такое абстрактный класс?
- Что такое статический класс?
- Что такое статический конструктор?

Звучит глупо, но обычно разговора вокруг этих вопросов достаточно, чтобы понять, с кем имеешь дело.

2.2 парадигмы программирования

В ООП на сегодняшний день есть ряд устоявшихся парадигм. Я не буду говорить про ужасы вроде UML или паттернов. Есть ряд конкретных вещей, которые нужны в жизни и их можно или знать или нет.

вопросы на тему:

- Что такое интерфейс?
- Что такое делегат? (для кандидатов, претендующих на знание .net)
- Что такое прототип функции ? (для кандидатов, не претендующих на знание .net)
- Что такое обработчик события?
- Что такое свойства (property)?
- Exception handling. Что такое исключительная ситуация? Зачем этот механизм? Как они обрабатываются? Объяснение конструкций try - finally, try - catch (try - except в delphi)

- Доп. вопрос для граждан с опытом в Delphi. Что такое dynamic и virtual? Общая эрудиция. По большому счету, тест на умение читать манюалы и интерес к ним.
- Для вундеркиндов: что такое атрибуты ?
- Для вундеркиндов: что такое женерики (Generic)? Что такое List<TMyOwnType>? Усложнение вопроса – что такое Dictionary<TKeyType, TValueType>? Что такое Hashtable? Как сделать цикл foreach по Dictionary<a,b>? А по Hashtable (сам уже не помню :)?

2.3 специфика управляемого кода

Отсутствие указателей, неявное освобождение памяти и ресурсов, garbage collectioning, подсчёт ссылок. Все эти вещи меняют традиционное представление о программировании. Точнее, для программирования в среде с такими механизмами традиционное представление о программировании приходится корректировать.

Ряд вопросов на эту тематику:

- Что выполняет в .net функцию конструктора - деструктора ? (самому надо бы уточнить)
- Выражение **using** в C#.
- Что такое IDisposable?
- Garbage collectioning – что это?

2.4 платформа

На самом деле, здесь может быть нескончаемая масса вопросов. Вот немногие, которые пришли мне на ум:

- В общих чертах, в чем различие между managed и unmanaged кодом? Что такое MSIL?
- Что такое assembly references?
- Что обозначает директива internal?
- Что обозначает директива sealed? (Склеротичка для себя: Можно вспомнить 1 фворк и констр-ры.)
- Что такое Global Assembly Cache (GAC)? Кто такие strong assembly names? Что такое gacutil, sn?
- Что такое Reflections? Как пользоваться, что делал с их помощью, что можно сделать?
- Что такое StringBuilder? Как им пользоваться?

- Что такое Weak References?

- Вопрос на внимательность и память. Приведение типов.
  Разница между (Type2Cast)AObject и AObject as Type2Cast. Знаете ли вы про:
  foreach(DescendantType AObject in *[CollectionOfParentType]*)?
  Какое приведение будет использовано в этом случае?

2.5 reversing

Не за горами время (елси оно еще на настало), когда подрастет поколение разработчиков, которые не будут иметь представления о организации выполнения программ на уровне процессора, памяти, машинных кодов, регистров и стека. Оставаясь при этом высококлассными специалистами.

.net – мощнейшая высокоуровневая платформа. Однако, интересно вникнуть при этом в то, как она функционирует на низком уровне. Я все-таки не гнушаюсь спрашивать людей, в каких отношениях они состоят с ассемблером. Попадаются граждане с представлением и опытом. Я не говорю, что нужно начинать разбираться с машинными кодами и т.п. для того, чтобы стать привлекательным .net - разработчиком; но и эти знания не помешают, если они у вас есть :).

Также я считаю, что для зрелого .net - девелопера навыки .net reverse engineering необходимы. Я так понимаю, что средства для этого уже добавлены в VS 2008.

Кроме того, никто не мешает пользоваться наружными средствами вроде ildasm, IDA и .net Reflector [2].

Для себя лично я колупание в рефлекторе нахожу архиполезным. Очень часто прибегаю к этому при оптимизации. Каким образом устроен поиск элемента у List<T> (метод .Contains(T AObject))? Стоит ли там foreach или используется некий внутренний механизм хэширования? Я объявляю переменную в начале цикла. Действительно ли она будет сниматься из стекового кадра после каждого прохода и добавляться в начале цикла или оптимизатор это утрамбует? На все подобные вопросы я ищу ответы в рефлекторе. The code doesn't lie. Нередко мне бывает проще разобраться в рефлекторе как что-нибудь работает, чем найти нужную статью в MSDN.

2.6 работа с данными

Тема на самом деле весьма обширная, я бы выделил группы навыков:

реляционные БД, SQL

- Что такое database (база данных)?

- Что такое таблица, сущность, атрибуты, поля?

- Что такое индексы?

- Что такое SQL, DML, DDL?

- Для чего используют SQL; примеры того, что можно им делать.

- Кто такие триггеры, хранимые процедуры, PL-SQL, Transact-SQL?
- Каким образом в СУБД управляют правами доступа к данным?

парадигмы работы с данными .net и ADO.net

- Вы работали с данными в Delphi? Расскажите, как это было :). Что нового-другого в студии?
- Расскажите, что собой представлят объектная модель ADO.net.
- Кто такие типизированные и нетипизированные DataSet-ы?
- Что такое connected и disconnected модели работы с данными?
- Что такое транзакции в контексте работы с данными? Как пользоваться ими в ADO.net? Каким образом составлять транзакции из операций над типизированными DataSet-ами?
- Вопрос из серии "знаете ли вы, что...". Что такое SMO for MS SQL server? (напомнить за транзакции)

XML

- Что такое XML?
- Что такое XSLT?

построение отчетов

Никаких четких теорий на этот счет я не знаю. Можно или уметь и иметь опыт, или нет.

2.7 и прочее

коммуникации

автор не претендует на глубокое знание вопроса

Весь разговор сводится к вопросам о наличии знаний о TCP/IP вообще и умении пользоваться этим стеком протоколов с помощью .net.

работа с потоками

автор не претендует на глубокое знание вопроса

Вопрос, собственно, нехитрый – умеет ли кандидат работать с потоками? Что об этом знает, что с делал с использованием многопотоковости, как? Кто такие Invoke, BeginInvoke, где они есть, идея их использования.

работа с распределенными приложениями

автор не претендует на глубокое знание вопроса

- Что можно назвать архитектурой клиент-сервер?
- Что такое сервер приложений?
- Что такое web-service? Как пользоваться ими в .net (с помощью VS)?

Version Control

Весьма полезным навыком является умение работать с системами контроля версий. При собеседовании интересуюсь – доводилось ли сталкиваться. Самыми модными для нет-девелоперов считаю Subversion[3] и Team Foundation Server. Для Delphi юзаем Jedi VSC[4].

Unit Testing

Экзотический навык, сомневаюсь, чтобы в Сумах было много народу, кто в этом разбирается. Сюда же можно отнести оккультизм вроде UML, паттернов, Extreme Programming и т.п. прогрессивщину. Если попадется человек, которому эти матюги о чем-то говорят – можно мило побеседовать. К сожалению, в реальной жизни извлечь из этих умений пользу удается редко.

вменяемость, коммуникабельность

коммуникабельность – это способность неконфликтно обмениваться информацией, а не склонность к болтовне :)

Я промолчу о том, что в ходе интервью вы невольно даете оценку адекватности человека вообще. Важно что, общаясь, вы обращаете внимание на то, насколько внятно человек излагает свои мысли, и в какой степени он способен понимать чужие. Если придется заниматься вместе масштабными разработками – для вас критично будет, чтобы человек точно понимал постановку и внятно пояснял принципы работы своих решений.

Do you speak?

Brett: What?
Jules: What country are you from?
Brett: What? What? Wh - ?
Jules: "What" ain't no country I've ever heard of. They speak English in What?
Brett: What?
Jules: English, motherfucker, do you speak it?
Brett: Yes! Yes!
Jules: Then you know what I'm sayin'?
Brett: Yes!
Jules: Describe what Marsellus Wallace looks like!
Brett: What?
Jules: Say 'what' again. Say 'what' again, I dare you, I double dare you motherfucker, say what one more Goddamn time!
(Pulp Fiction)

Программист, знающий инглиш, впятеро более эффективен чем программист, его не знающий.

3 Начинаем разговор

Я не психиатр и не кадровик, я не буду сильно распространяться о формате интервью. Однако, замечу, что целесообразно расспросить человека для начала кто он, что он, чем занимался и чего умеет. Первым делом я прошу рассказать о себе (учёба, карьера) и детально выслушиваю историю участия в разработках. Расспрашиваю об архитектурных особенностях продуктов, в разработке которых кандидат участвовал; о том, каким именно участком он занимался. Пытаюсь при этом выяснить не только какими технологиями он владеет, но и в какой степени он способен усвоить метафору и модель продукта и объяснить её кому-то.

В ходе этих разговоров о погоде или после них можно обсудить глубину познаний собеседника в ООП и парадигмах, перечисленных в 2.2. В этой же окрестности уместны расспросы об опыте работы с .net, с БД, с Visual Studio или аналогами; с вещами вроде построения отчетов и т.п.

Если человек показывает хоть какой-то уровень в вопросах проектирования БД, я стараюсь поговорить с ним об этом побольше. Во-первых, просто выясняю габариты эрудиции. Во-вторых, могу задать какие-нибудь конкретные вопросы из серии "как сделать".

Приведу пару любимых, просто для примера.

- Есть таблицы a и b, в каждой из них определен первичный ключ. В таблице b есть поле, значения в котором ссылаются на записи из таблицы a по первичному ключу. Построить выражение, которое бы выбирало из таблицы a все записи, для которых существуют ссылающиеся записи в таблице b. Построить выражение, которое бы выбирало из таблицы a все записи, для которых **не** существуют ссылающиеся

записи в таблице b. Нюанс. В обоих случаях не использовать вложенных select-ов (и, как следствие, оборотов exists(. . .)).

- Пример для вундеркиндов. Построить шахматку.

  Есть таблица с информацией о платежах. Структура таблицы: (id – автоинкремент идентификатор первичный ключ; month – целое, месяц, в котором имел место платеж; PaymentSum – деньги с фикс. точкой, сумма платежа; acc – для простоты –целое, счёт, с которого взяты средства на платёж).

  Требуется построить выборку, результатом которой была бы структура данных: (acc, sum01, sum02, sum03, . . . , sum11, sum12, sumTotal), где acc – счет, sum01 .. sum12 – общая сумма платежей за месяц по счету, sumTotal – общая сумма платежей по счету за все месяцы.

  Таким образом, каждая строка выборки должна содержать помесячную оплату по некоторому счету и суммарную годовую оплату.

  Для простоты считаем, что год при построении выборки не важен (все данные за один год; в поле месяц – числа от 1 до 12).

4 Итоги подведём

Не знаю, много ли народу готовы продемонстрировать фундаментальные знания по всем вопросам, затронутым в данном индексе. Подозреваю себя в наличии пробелов и белых пятен по некоторым. Не уверен, что специалист с глубокими знаниями по всем пунктам обязательно будет эффективным разработчиком (хотя вероятность этого все-таки велика :). Данный документ – просто памятка для адекватной оценки себя или кого-то, список из разряда ”а не забыл ли я про . . . ?“. Что из него более важно, что менее, определяется конкретной ситуацией.

Даже если человек ничего не знает про .net, но имеет опыт работы с java и MySQL – может иметь смысл его брать. После некоторой акклиматизации и обучения с него может быть много толку (в таком случае главное – оценить вероятность текучести. Брать человека на развитие стóт только если высока вероятность, что он вас не покинет).

Может быть ”крыза“ выгнала из теплых табуреток на рынок труда матёрых умельцев, но до неё кандидатов, готовых зрело разрабатывать на .net-е, мне не попадалось практически.

Так что при выборе кадров главное – оценить базовые умения (то же ООП, СУБД), и специфические вашей задаче знания.

Для потенциальных кандидатов. Тут перечислена масса областей взаимодействия с .net. Никто не имеет в виду, что вы моментально разберётесь со всем этим. Стóит с чего-то начать и что-нибудь писать. Если делать это достаточно настойчиво и глядеть в процессе по сторонам – есть все шансы многому научиться.

5 И кроме того

Кроме того, попадалась недавно сравнительно смежная статья – [5].

Досочинив некую бета-версию этого документа, я попросил Артёма Волка ознакомиться и выдать конструктивную критику. Он это сделал, но ума вкрутить его замечания к месту у меня не хватило. Я не придумал ничего лучше, чем привести их тут.

> AVolk (13:29:06 6/03/2009)
> да, ещё, в списке специализаций нет веб-программистов :)
>
> zx (13:29:25 6/03/2009)
> > веб-программистов :)
> они думаю не-нет-девелоперы.
>
> AVolk (13:29:38 6/03/2009)
> ну а ASP.NET?
>
> zx (13:29:53 6/03/2009)
> хз. это оккультизьм какойто.

Я на досуге обдумал этот момент. Моя позиция такова: .нет-девелопер – специалист, первичным умением которого является навык работы с фреймворком. А бойцы с asp.net-ом – либо являются web-разработчиками (что по-моему подразумевает кучу отдельных навыков), либо используют asp.net как элемент в коммуникационной подсистеме, и тогда заострять на этом внимания нечего.

> По-моему майкрософтовцы дико обижаются, когда .NET пишут .Net, а не .NET или .net, впрочем, это их проблемы :)

Меня это, на самом деле, тоже мало беспокоит. Но вроде поправил :).

> Про веб-сервисы – может, стоит упомянуть WCF, а то майкрософтовцы в очередной раз решили, что это будет их единственным и новым API для Remoting, веб-сервисов и компании. Кстати, раз есть вопрос о серверах приложений – можно спросить о CORBA/COM+/DCOM (для делфи) и Remoting (.NET), но это тоже уже жесть.

Затронул только то, с чем реально сталкивался хоть сколько-нибудь. В вопросе организации "многозвенок" мне самым важным кажется понимание общих принципов, способность подготовить публикуемые данные и готовность разобраться с конкретным решением, "вяжущим" звенья. Иметь глубокую эрудицию о ВСЕХ средствах такой связки мне кажется избыточным. Вполне достаточно уметь работать с чем-то одним и быть готовым поменять **это** на что-нибудь другое при очередной обнове.

> Жалко, что у тебя вопросы по доступу к данным заканчиваются на типизированных датасетах и ты не упонянул об ORM (хотя бы LINQ to SQL), не думаю, что кто-то в наших краях работал хотя бы с ним не говоря уже об Entity Framework и всяких явовских портах, но хотябы упомянуть об этом в докладе, может стоит, а то вдруг умники в аудитории будут? :)

На самом деле мне жаль только оттого, что я про все перечисленное практически ничего не знаю. Предлагаю ограничиться упоминанием о том, что ВСЕ ЭТО ЕСТЬ. И сойтись на мнении, что это важные дополнительные навыки.

Кроме того, я старался перечислить наиболее важные и базовые знания в профессии. С этой точки зрения дай бог как следует разбираться в ООП, SQL и собственно проблеме ORM. А уже догнать новобранца с этими скилами до нужного уровня в работе с конкретной OPM – решаемый вопрос.

Список литературы

[1] http://www.lua.org, http://luaforge.net, http://luaforge.net/projects/luainterface

[2] http://www.red-gate.com/products/reflector

[3] http://subversion.tigris.org/, http://tortoisesvn.tigris.org/

[4] http://jedivcs.sourceforge.net/

[5] Собеседование программистов в Челябинске. Что собственно надо знать.

`http://samolisov.blogspot.com/2007/11/blog-post_18.html`

[6] C# и платформа .NET 3.0 для профессионалов. Кристиан нейгел, Билл Ивьен и др. Диалектика, 2008. (wiley publishing)