Developer interview

Vadim Platonov

2011–07–20

Readings Worth Notice

Artima

- Explore an area of expertise
  - rather than simply look for expertise and experience in the exact area in which the candidate will work, you should look for general programming talent and ability. Not only will these candidates adapt better in the future, they're also more likely to be innovative in the present.
- Have the candidate critique something
  - ask candidates to critique a system or platform that you both have in common, preferably something they will use on the job. For example, you might ask, "What parts of Java don't you like and why?"
- Have the candidate solve a small-scale design problem
- Look at their code
- Find out what books they read

Asymptomatic

- Great developers keep their skills fresh
  - the trademark of a great developer is one who has not finished learning and has a voracious appetite for new things.
- Great developers don't work alone
- Great developers are opinionated

  - ask a question like “Which do you prefer and why: Spaces or Tabs?” Surprisingly, there is only one correct answer to this question, and that answer isn’t “Spaces” or “Tabs”. The answer is an adamant, ravenous, blitzkrieg of a response one way or the other.
  - ask them about tools. Some will say that a good developer won’t care what tools they have to use, that they’ll be able to build using whatever they’re given. That’s true. But they’re going to prefer certain tools, and that’s the important part.

Software by Rob

- Pessimistic in the short term, optimistic in the long term
- Angered by sloppy code
- Great attention to detail
  - I’ll say it very loud, but I promise I’ll only say it once: I have never, ever, ever seen a great software developer who does not have amazing attention to detail.
- Not afraid to take risk

Interview structure

Introduction

Introduce yourself, explain what the company does and what the job entails. Explain the team, environment, etc. Explain why the company is a great place to work. Explain what the interview will entail.

Validation of the resume

Review the resume, and ask about one project the person liked. Ask them to go in-depth about that project. If the person really did the project, they will know the issues inside out. Question them on why they chose a framework, methodology, etc. Scan the resume for keywords. If you are concerned about something you see on the resume, ask a few questions to explain about the technology they used and why.

Basic coding

Ask them to write a short piece of code, right on the spot. Give them access to a box with an IDE (or other editor of their choice). Something simple, like the reversing a string or saving URL contents to a file. Review the code with them, if something is wrong, ask them why. Strong programmers use good variable names, etc.

Social interaction

Ask if they like to work alone or in a team.

System design and ability to solve problems (Senior)

Think of a situation in your experience where you had to design a module or small system. Ask them to design the system (architecturally, maybe with some details on APIs, etc). Ask for which frameworks, APIs, etc they would use, and why.

Reference

Junior dev

- Algorithms
  - How to find out if a number is a power of 2?
  - What's the complexity of locating the element in the middle of a linked list?
- Data structures
  - How would you represent the railway stations of a city in program's memory?
  - Which structure allows for a faster random element access: a hashtable or a sorted list?
- Ability to write programs
  - Solve the FizzBuzz problem.
  - Write the string reverse function which doesn't use built-in reverse functionality.
- Java
  - Do you have any experience in Java?

- Understanding of databases
  - Relational (SQL)
    - List main RDBMS vendors.
    - What is the meaning of 'null'?
    - What is a join? List the types of joins you know.
  - NoSQL
    - What do you know about NoSQL databases?
- Source version control systems
  - What are the benefits of using VCSes?
  - Which one do you like the most, if any? Why?

Non-junior dev

- Java
  - General
    - What's the difference between an `abstract class` and an `interface`?
  - Core
    - What's the difference between creating a string using its constructor or with string literals?
  - Collections
    - Which methods an object needs to implement in order to work correctly with hash-based collections?
    - What's the difference between the `Hashtable` and a `HashMap`?
  - OO design
    - In what situation would you use immutable objects?
    - What's the difference between entities and value objects in the domain model?
    - Which design patterns do you find yourself using the most?
- Source version control systems
  - Which one do you like the most, if any? Why?
- Previous work experience
  - Describe the architecture of the project you've been working on.
- Code quality standards
  - When you review peers code, what bothers you the most?

- Enthusiasm/Dedication to the job
  - Do you have any side projects (if not listed in CV)? Tell us about them.
  - Do you have any favourite technical blogs or tech-news sites you read periodically?

Answers

How to find out if a number is a power of 2?

Optimal solution: `(n & -n) == n`

Other solutions:

- Use the `log` function in the language (`log(n, 2) == floor(log(n, 2))`)
- Use some kind of bit shifting starting from `2` (`int t = 2; while (t < n) { t = t << 1; } t == n`)

What's the complexity of locating the element in the middle of a linked list?

Expected answer: O(N + N/2) as we need to traverse the list in order to find its size and then traverse to the middle element.

Bonus points: Java implementation of the `LinkedList` stores the size locally (in a transient field), so there's no need to recompute it by traversing elements.

How would you represent the railway stations of a city in program's memory?

Expected answer: As a graph.

Which structure allows for a faster random element access: a hashtable or a sorted list?

Expected answer: Hashtable allows O(1) access while sorted list is O(log2 N) if using binary search.

Bonus points: If only one bucket of the hashtable is filled (all of the inserted elements had hash collisions or hashtable contains only one bucket - degenerate case), the access cost will be O(n).

FizzBuzz

```
static void fizzBuzz() {
    for (int i = 0; i < 100; i++) {
        if (i % 3 == 0) {
            System.out.print("Fizz");
        }
        if (i % 5 == 0) {
            System.out.print("Buzz");
        }

        if (i % 3 != 0 && i % 5 != 0) {
            System.out.print(i);
        }
        System.out.print("\n");
    }
}
```

Reverse string

Without built-in functions:

```
static String reverse(String s) {
    int length = s.length();
    char[] result = new char[length];
    for (int i = length - 1; i >= 0; i--) {
        result[length - i - 1] = s.charAt(i);
    }
    return String.valueOf(result);
}
```

Bonus points:

```
static String reverse(String s) {
    return new StringBuilder(s).reverse().toString();
}
```

Explain the difference between `StringBuilder` and `StringBuffer`.