

# 1

## Before the Search

Before starting your job search, there are some preliminary tasks to perform. There's no point applying for jobs without knowing what you like, for example. Just being a good coder isn't enough—you have to understand what the market wants and how you can adapt your own skills to find the right job for yourself.

### Know Yourself

Stereotypes to the contrary, all programmers are *not* alike. Knowing what kind of programmer you are is crucial to finding the right kind of job. While you can probably do many different kinds of programming tasks, they won't all turn your crank in the same manner. Doing something you don't really enjoy is fine on a short-term basis, but you need to be interested in and excited by what you're doing to sustain you over the long term. The best programmers are passionate about their work, and you can't truly be passionate about something that's only moderately interesting to you.

If you're not sure what you like or dislike, ask yourself some questions:

- ❑ **Are you a systems programmer or an application developer?** Systems programmers work on the code that keeps computer systems running: frameworks, tools, compilers, drivers, servers, and so on. Other programmers are their primary audience, and there's little interaction with nonprogrammers. Application developers, on the other hand, work on the pieces that those nonprogrammers use to do their own work, and there's often more interaction with nontechnical people.
- ❑ **Do you like coding user interfaces?** If so, and if you're skilled at it, consider yourself lucky. User interface design is finicky work, easy to criticize, and hard to do well, especially when internationalization and accessibility issues are taken into account.
- ❑ **Are you a good debugger?** If you think finding problems in your own code is bad enough, imagine what it's like to fix problems with someone else's code. It requires strong analytical and problem-solving skills. Finding and fixing bugs can be extremely rewarding in its own right, but it's definitely *not* for everyone.

## Chapter 1: Before the Search

---

- ❑ **Do you like testing?** Testing — also referred to as *quality assurance*, or QA for short — is often maligned by inexperienced programmers, but those who've been around the block once or twice value independent testing. Skilled testers are hard to find, and programming skills are usually required to write tools and automated test cases.
- ❑ **Are you an architect or a coder?** Every coding job includes some kind of design aspect, but certain jobs lean more one way than the other. If you enjoy the designing more than the coding, a position as a software architect might be more appealing. That said, architecture positions can involve *a lot* of interaction with others and little or no coding, though you need a good understanding of how to code in order to be an effective architect. Unless you take formal training in software architecture, the usual route to becoming an architect is to code first and then display an aptitude for designing and fitting together different pieces of a project.

While the preceding questions deal with the different kinds of programming that might interest you, there are also nonprogramming questions to consider:

- ❑ **Does management interest you?** Some coders have a long-term goal of becoming a manager, but others shiver at the very thought. If management is your goal, however, you'll need to develop leadership skills and demonstrate that you can manage the human parts of the software development equation as well as the technical pieces. If management is *not* your goal, look for companies with good *technical* career paths so you're not forced to manage people in order to be promoted.
- ❑ **Do you want to work for a big company?** There are advantages and disadvantages to working at big companies. For example, a large company usually offers more job stability and some kind of career path. It may also have a name brand that nontechnies recognize. On the other hand, you may feel stifled by the bureaucracy, rigidity, and intercompany rivalry that is often found within bigger companies.
- ❑ **Do you want to work for a small company?** The pay may be less, but getting in on the ground floor at a new company can ensure future advancement (and possibly substantial remuneration) as the company grows and succeeds. The downside, of course, is that most new ventures fail and you may be out of a job within a year or two.
- ❑ **Are open-source projects preferable?** The vast majority of programming jobs have usually involved proprietary, closed-source projects, which some programmers find objectionable. There's been a small shift in favor of more open software development, which provides more opportunities for people like yourself to participate in open-source projects and still be paid for that participation.
- ❑ **Do you want long-term or short-term projects?** Some programmers crave change, spending a few months at most on each project. If you like short-term projects and don't mind traveling, a gig with a consulting company might make more sense than a more conventional corporate job.

It's important to realize that there are no universal answers to these questions, no right or wrong way to answer them. The more truthful you can be with yourself in answering them, however, the more likely you'll be able to find the kind of programming job you truly enjoy.

## Know the Market

Knowing what you'd like to do is great, but don't box yourself in too narrowly. You also need to understand the current job market and how it constrains your search for the "ideal" job, especially during an economic downturn like the one that burst the original Internet bubble of the late '90s.

### Basic Market Information

There are a number of sources of information about what's hot and what's not in the developer job market, including the following:

- ❑ **Online job listings** — Large job sites such as Dice (which specializes in technology-related career listings), Monster, and HotJobs are your first line of research into what companies want.
- ❑ **Bookstores** — Even though more and more programmer documentation is available online, printed books are still a significant market for technical publishers. The number of books published on any given topic is a good indication of the degree to which skills related to that topic are valued by potential employers. Look out especially for niche topics that are suddenly going mainstream.
- ❑ **Social networking and bookmarking sites** — Some of these sites enable you to find potential employers, or enable them to find you. Others provide an indirect "pulse" of the market by the bookmarks and comments other programmers leave about various technologies and employers.
- ❑ **Professional development courses** — Colleges and universities try to keep abreast of what companies want, and create professional development courses around those needs.

If you're not in college or university anymore, find out what languages and technologies the local institutions and/or your alma mater are requiring of their computer science students; although academic needs don't always coincide with what employers want, educational institutions on the whole try to graduate students with practical skills that employers can use.

### What About Outsourcing?

The rise of *outsourcing* — having an outside company handle tasks that aren't central to a company's lines of business — is always a topic of heated discussion within the development community. Outsourcing is not new, of course — companies have long outsourced tasks such as payroll administration and property maintenance — but the growing number of well-educated people in developing nations and the fact that software development can be done anywhere there's Internet access have made it possible for companies to *offshore* tasks that would normally have required a *local* workforce, whether or not the jobs were outsourced. The disparity in wages between offshore and local talent can result in large savings for companies that offshore their software development — at least that's the promise that the outsourcing companies make in their sales pitches.

Many software developers find themselves worrying whether outsourcing (and offshoring in particular) is going to put them out of a job, especially those who work in information technology (IT) departments within a larger company looking to cut costs wherever they can. These fears are not unfounded, unfortunately, so when you're looking for a job consider taking steps to avoid landing a job that will be outsourced at some point in the future. Following are some suggestions:

- ❑ **Work for software development firms** — A software firm's *raison d'être* is the intellectual property it develops. While medium and large firms may open development centers in other parts of the world, the smart ones are unlikely to move their entire operations to other countries or entrust their future to outside firms. That said, some companies will outsource all or substantial parts of a project to the developing world for cost reasons, so it pays to research a company's behaviors and policies in this regard.
- ❑ **Work for an outsourcer** — For various reasons, many outsourcing firms end up hiring personnel in the developed world, including the United States.
- ❑ **Move up the programmer food chain** — Design-oriented jobs are less likely to be outsourced. Good coders are cheap and plentiful, but good designers are much harder to find. (This assumes, of course, that good design skills are separate from good coding skills, and not everyone takes that view, although many companies do.)
- ❑ **Take a management job** — Management can be a refuge from outsourcing, so a management-oriented career path is one option to consider.

Of all these options, moving up the food chain is usually the best approach. The more nonprogramming knowledge your job requires, or the more interaction with customers, the less likely you are to be outsourced. There's no *guarantee* you'll never be outsourced, of course, or that you'll always keep your job. Your company may shutter or downsize the project you're working on at any point, after all, and put you back on the street. This is why developing reusable and marketable skills throughout your career is extremely important.

## Develop Marketable Skills

The appendix covers how your résumé is primarily a *marketing tool* to get you job interviews. This assumes, of course, that you have *marketable skills* to offer a prospective employer. You can only stretch the truth so far, and if even you exaggerate or outright lie about your skills and what you've accomplished in the past, you probably won't make it through technical interviews designed specifically to weed out the liars and exaggerators of this world. What you need to do, then, is develop skills and accomplishments that will make you stand out from the crowd both on paper and in the interviews, especially if you're entering the job market for the first time. Here are some approaches you can take:

- ❑ **Upgrade your credentials** — Companies such as Google are well known for favoring job applicants with graduate degrees. Getting a master's or doctorate degree is one way to upgrade your credentials. Although pursuing a graduate degree is a large commitment on your part, you can upgrade your credentials in other ways, such as taking university or professional development courses or participating in programming contests.
- ❑ **Get certified** — Certification is a touchy issue in the software development profession, but there's no doubt that some jobs either prefer or require candidates to be certified in specific technologies, especially IT jobs.
- ❑ **Work on a side project** — A great way to expand your skill set is to work on a project that is not directly related to your primary work or study focus. Starting or joining an open-source development project is one way to go. Or if you're working at a company, see if they'll let you spend time on an ancillary project.

- ❑ **Do well in school** — Although grades aren't everything, they are one measure that companies use in order to rank new graduates with little job experience. The better your grades, especially in computer science and mathematics courses, the more you'll impress a potential employer.
- ❑ **Keep learning** — The end of formal education doesn't mean you should stop learning, especially when there's so much information about programming available from a wide variety of sources. Whether it's books or blogs, there's always a way to keep yourself current, no matter what type of programming you do. It's also a great way to expand your horizons and discover other areas of interest.
- ❑ **Be an intern** — New graduates who've managed to secure employment during their nonschool terms — especially those that participate in cooperative education programs — have a huge leg up over their peers who haven't yet ventured into the real world. Software development in the field is often very different from software development in an academic setting, and potential employers are very cognizant of this.

The key is to *keep learning*, no matter what stage of your career you're at. Marketable skills don't develop overnight; they take some effort and initiative on your part, but they'll have long-lasting effects on your career.

Note that one of the best ways to develop marketable skills is to *accomplish something*, whether it's in your current job, something you did as a side project, or something you worked on as an intern or for a class project. Being able to talk intelligently and confidently about a project for which you played a primary role in its success is incredibly important. Make sure you can describe the problem clearly and succinctly and how your project solved the problem, even to a nontechnical person. Displaying a passion for programming is always positive and one way to make yourself stand out from the other candidates.

## Sanitize Your Online Profile

Once you've applied for a job, the first step in landing an interview is to make it past the screeners. Screeners are the humans who examine prospective job applicants after their résumés have made it through the company's automated filtering system. Their task is to trim the applicant pool to manageable levels. They comb through the remaining résumés, looking for anomalies and discrepancies in each job application, or any indication that the prospective employee would be a poor fit for the company. Even if your résumé is perfectly tailored to make it past the filtering system, the screeners can stop your application dead in its tracks based on what they find out about you, which is why your online profile is so important.

Many screeners now look online to learn more about prospective employees. The impression they get from your online profile can affect your chances of being hired. If your résumé lists extensive experience with C# but they find a forum posting you made only six months ago asking how to open a file in C#, they'll probably conclude that you're exaggerating your experience level, putting your whole résumé into doubt. Or if they see disturbing and/or inflammatory material that they think you've authored, they may decide to pass you over for an interview, no matter how well your résumé reads or how long ago you wrote those things. No one's proud of everything they ever did in high school or college, but those who have grown up in the post-Internet era will see things follow them that they'd rather forget about, something the older generations rarely had to contend with.

## Chapter 1: Before the Search

---

At some point before you apply for a job, take a good look at your online profile. Put yourself in a company's shoes and see how much information — good or bad — they can find about you, or link to you. If your online profile is possibly going to prevent you from being hired, take some steps to sanitize your profile. Remove questionable material from the Web and from the search engines. Abandon old e-mail addresses and online identities.

Develop a newer, better online profile for yourself that doesn't throw any red flags in front of the screeners. Finding a good job is hard enough as it is — why make it harder?

## Summary

What you do *before* a formal job search is critical to finding the right kind of job. With that in mind, you should consider the following things:

- ❑ Know your likes and dislikes as a programmer and a prospective employee.
- ❑ Understand the market in order to find and apply for the best jobs.
- ❑ Develop the marketable skills that employers look for and that will enhance your career.
- ❑ Don't forget to check your public profile to make sure there are no surprises that can turn off potential employers.

When you have a good grasp of all of these points, you're ready to begin your job search.