



Patterns Exposed

This week's interview:
Confessions of a Singleton

HeadFirst: Today we are pleased to bring you an interview with a Singleton object. Why don't you begin by telling us a bit about yourself?

Singleton: Well, I'm totally unique; there is just one of me!

HeadFirst: One?

Singleton: Yes, one. I'm based on the Singleton Pattern, which ensures that at any time there is only one instance of me.

HeadFirst: Isn't that sort of a waste? Someone took the time to develop a full-blown class and now all we can get is one object out of it?

Singleton: Not at all! There is power in ONE. Let's say you have an object that contains registry settings. You don't want multiple copies of that object and its values running around—that would lead to chaos. By using an object like me you can ensure that every object in your application is making use of the same global resource.

HeadFirst: Tell us more...

Singleton: Oh, I'm good for all kinds of things. Being single sometimes has its advantages, you know. I'm often used to manage pools of resources, like connection or thread pools.

HeadFirst: Still, only one of your kind? That sounds lonely.

Singleton: Because there's only one of me, I do keep busy, but it would be nice if more developers knew me—many developers run into bugs because they have multiple copies of objects floating around they're not even aware of.

HeadFirst: So, if we may ask, how do you know there is only one of you? Can't anyone with a new operator create a "new you"?

Singleton: Nope! I'm truly unique.

HeadFirst: Well, do developers swear an oath not to instantiate you more than once?

Singleton: Of course not. The truth be told...well, this is getting kind of personal but...I have no public constructor.

HeadFirst: NO PUBLIC CONSTRUCTOR! Oh, sorry, no public constructor?

Singleton: That's right. My constructor is declared private.

HeadFirst: How does that work? How do you EVER get instantiated?

Singleton: You see, to get a hold of a Singleton object, you don't instantiate one, you just ask for an instance. So my class has a static method called `getInstance()`. Call that, and I'll show up at once, ready to work. In fact, I may already be helping other objects when you request me.

HeadFirst: Well, Mr. Singleton, there seems to be a lot under your covers to make all this work. Thanks for revealing yourself and we hope to speak with you again soon!