Protocol versus Category

Asked 16 years ago Modified 11 years, 3 months ago Viewed 22k times

39

Can anyone explain the differences between **Protocols** and **Categories** in Objective-C? When do you use one over the other?

objective-c objective-c-category objective-c-protocol

Share

Improve this question

Follow

edited Sep 17, 2013 at 5:39

JiteshW

2,205 ● 4 ● 34 ● 61

asked Dec 11, 2008 at 21:22

Coocoo4Cocoa

50.7k ● 50 ● 152 ● 176

7 Answers

Sorted by: Highest score (default)

92

A protocol is the same thing as an interface in Java: it's essentially a contract that says, **"Any class that implements this protocol will also implement these methods."**

A category, on the other hand, just binds methods to a class. For example, in **Cocoa**, I can create a category for `NSObject` that will allow me to add methods to the

`NSObject` class (and, of course, all subclasses), *even though I don't really have access to* `NSObject`.

To summarize: a protocol specifies what methods a class will implement; a category adds methods to an existing class.

The proper use of each, then, should be clear: **Use protocols to declare a set of methods that a class must implement, and use categories to add methods to an existing class.**

Share Improve this answer

Follow

edited Jun 14, 2012 at 14:00

The iOSDev

5,267 ● 7 ● 42 ● 78

answered Dec 11, 2008 at 21:40

mipadi

410k ● 90 ● 531 ● 487

30

A protocol says, "here are some methods I'd like *you* to implement." A category says, "I'm extending the functionality of this class with these additional methods."

Now, I suspect your confusion stems from Apple's use of the phrase "informal protocol". Here's the key (and most confusing) point: an informal protocol is actually not a protocol at all. It's actually a category on NSObject. Cocoa uses informal protocols pervasively to provide interfaces for delegates. Since the `@protocol` syntax didn't allow optional methods until Objective-C 2.0, Apple implemented optional methods to do nothing (or return a

dummy value) and required methods to throw an exception. There was no way to enforce this through the compiler.

Now, with Objective-C 2.0, the `@protocol` syntax supports the `@optional` keyword, marking some methods in a protocol as optional. Thus, your class conforms to a protocol so long as it implements all the methods marked as `@required`. The compiler can determine whether your class implements all the required methods, too, which is a huge time saver. The iPhone SDK exclusively uses the Objective-C 2.0 `@protocol` syntax, and I can't think of a good reason not to use it in any new development (except for Mac OS X Cocoa apps that need to run on earlier versions of Mac OS X).

Share Improve this answer

Follow

answered Dec 11, 2008 at 21:56

Alex

26.9k ● 3 ● 58 ● 74

16

Categories:

A category is a way of adding new methods to all instances of an existing class without modifying the class itself.

You use a category when you want to add functionality to an existing class without deriving from that class or re-writing the original class.

Let's say you are using `NSView` objects in cocoa, and you find yourself wishing that all instances of `NSView` were able to perform some action. Obviously, you can't rewrite the `NSView` class, and even if you derive from it, not all of the `NSView` objects in your program will be of your derived type. The solution is to create a category on `NSView`, which you then use in your program. As long as you `#import` the header file containing your category declaration, it will appear as though *every* `NSView` object responds to the methods you defined in the catagory source file.

Protocols:

A protocol is a collection of methods that any class can choose to implement.

You use a protocol when you want to provide a guarantee that a certain class will respond to a specific set of methods. When a class adopts a protocol, it promises to implement all of the methods declared in the protocol header. This means that any other classes which use that class can be certain that those methods will be implemented, without needing to know anyting else about the class.

This can be useful when creating a family of similar classes that all need to communicate with a common "controller" class. The communication between the controller class and the controlled classes can all be packaged into a single protocol.

Side note: the objective-c language does not support multiple inheritance (a class can only derive from one superclass), but much of the same functionality can be provided by protocols because a class can conform to several different protocols.

Share Improve this answer

Follow

edited Dec 12, 2008 at 0:51

answered Dec 11, 2008 at 21:41

James Eichele

119k ● 41 ● 182 ● 214

5

To my understanding Protocols are a bit like Java's Interfaces. Protocols declare methods , but the implementation is up to each class. Categories seems to be something like Ruby's mixins. With Categories you can add methods to existing classes. Even built-in classes.

Share Improve this answer

Follow

answered Dec 11, 2008 at 21:44

PEZ

17k ● 7 ● 47 ● 66

4

A protocol allows you to declare a list of methods which are not confined to any particular class or categories. The methods declared in the protocol can be adopted any class/categories. A class or category which adopts a

protocol must implements all the required methods declared in the protocol.

A category allows you to add additional methods to an existing class but they do not allow additional instance variables. The methods the category adds become part of the class type.

Share Improve this answer

Follow

answered Sep 4, 2011 at 18:53

Nam

41 • 1

+1 for mentioning that a protocol can also be adopted by a category. The syntax is this: `@interface ClassName(CategoryName) <ProtocolName>` . – herzbube Nov 1, 2020 at 10:40

1

Protocols are contracts to implement the specified methods. Any object that conforms to a protocol agrees to provide implementations for those methods. A good use of a protocol would be to define a set of callback methods for a delegate (where the delegate must respond to all methods).

Categories provide the ability to extend a current object by adding methods to it (class or instance methods). A good use for a category would be extending the NSString class to add functionality that wasn't there before, such as adding a method to create a new string that converts the receiver into 1337 5P34K.

```
NSString *test = @"Leet speak";
NSString *leet = [test stringByConvertingToLeet];
```

Share Improve this answer

Follow

answered Mar 19, 2009 at 10:11

dreamlax
95.3k ● 31 ● 164 ● 211

1

Definitions from S.G.Kochan's "Programming in Objective-C":

Categories:

A category provides an easy way for you to modularize the definition of a class into groups or categories of related methods. It also gives you an easy way to extend an existing class definition without even having access to the original source code for the class and without having to create a subclass.

Protocols:

A protocol is a list of methods that is shared among classes. The methods listed in the protocol do not have corresponding implementations; they're meant to be implemented by someone else (like you!). A protocol provides a way to define a set of methods that are somehow related with a specified name. The methods are typically documented so that you know how they are to perform and so that you can implement them in your own class definitions, if desired. A protocol list a set of methods, some of which you can optionally implement,

and others that you are required to implement. If you decide to implement all of the required methods for a particular protocol, you are said to conform to or adopt that protocol. You are allowed to define a protocol where all methods are optional, or one where all are required.

Share Improve this answer

Follow

answered Jun 14, 2012 at 11:00

Stalert John

21 ● 1