**Design Pattern Interview Questions and Answers Introduction**

|  |
| --- |
|  |

Design patterns have become very popular interview questions, and in this section we present some of the design pattern questions being asked in interviews these days. What exactly are design patterns? Well, in software engineering there are design problems that commonly reappear in different projects. So, you can think of design patterns as ***guidelines*** to help you find a solution to some of those commonly reoccurring problems. Design patterns do not specify the actual code that needs to be used to solve a given problem, but just the guidelines.

**Where did design patterns originate from?**

Although design patterns were probably used by programmers for many years before, design patterns were *formally* introduced in computer science in a book called “Design Patterns: Elements of Reusable Object-Oriented Software”. This book was published in 1994 and is still considered to be the bible of design patterns. There are four people who wrote the book, and they are commonly referred to as the Gang of Four, or GoF (even Go4) for short – many times GoF is even used to refer to the book itself.

**Design patterns versus frameworks**

Are design patterns the same thing as frameworks? The answer to that is no. Design patterns are more like general guidelines on how to solve specific programming problems, but they do not specify the detailed code that’s necessary to solve those problems.

**Why should we use design patterns?**

|  |
| --- |
|  |

Design patterns are useful because they provide a pre-formulated solution to problems based on the experience of other programmers. This can save you a lot of time.

Design patterns also help in communication of a programming solution, and provide a common technical vocabulary between programmers.

**What about anti-patterns?**

|  |
| --- |
|  |

Of course, using the wrong design pattern can have a very negative effect on your code. So, it takes good judgement to use the correct design pattern. The term “anti-pattern” is used to describe a poor programming practice that leads to ineffective code.

Interview questions about design patterns are becoming more popular, especially for more mid to senior level programming jobs. You can get started on reading our set of questions by clicking Next, or clicking the page of interest on the left in the menu bar.

**When using the Observer pattern, what approaches can be used by the subject so that it’s observers can be more efficiently updated?**

Let’s go through a quick overview of the Observer pattern before answering the actual question.

In the Observer pattern, an object can broadcast any changes in it’s state to any “observers” – which are basically other classes – that want to know about it’s state. Why would an “observer” be interested in another object’s state? Well, because the state of the object can affect the state of the observer – read below for an actual example. So, the object being observed is called the “subject”. These observers are typically notified when the subject changes because the subject will call a method belonging to an observer.

**Observer Pattern is used in MVC**

|  |
| --- |
|  |

The Observer pattern plays a key role in the popular MVC – Model View Controller – architectural pattern. Any change in state to the Model – which represents the underlying data, and is the “subject” – will result in a notification to the View, which is the Observer.

Now, let’s answer the actual question. One problem that may arise with the Observer pattern is that a subject may be updating it’s state too frequently. This means that the subject is spending a great deal of time updating it’s observers, which is of course inefficient. In this scenario, a possible solution is to simply have the subject turn off all updates temporarily. Then, the subject can make the changes in that period of time. And, once the changes are complete, the subject can go ahead and update any and all observers with one big notification. This is far more efficient because of the fact that only notification is sent out to observers as opposed to constantly sending out notifications.

**Finding a good strategy with the Observer Pattern**

One other possible dilemma when dealing with the Observer pattern is figuring out a good strategy for observers to know what has been changed in the subject. Let’s consider an example website (like Yahoo.com) that has to update particular parts of the page (the Observer) when the data (the subject) changes. So when a stock price changes, Yahoo will have to update the stock portion of the page, and if there’s some breaking news, then Yahoo will have to update the “Latest News” portion of the page, etc.

In order for this process to be efficient, the Observer (which is the actual webpage) would need to know what data (the subject) has changed – whether that data is in the form of a database, XML file, or whatever. But, instead of having the page ask the data (or query the data) to find out what exactly has changed, it would probably be more efficient to have the data (the subject) pass on that information to the webpage. This information passed to the webpage from the subject could just be added to the normal update notification, and then the webpage can update the appropriate portion of the page that needs to change (maybe using AJAX).