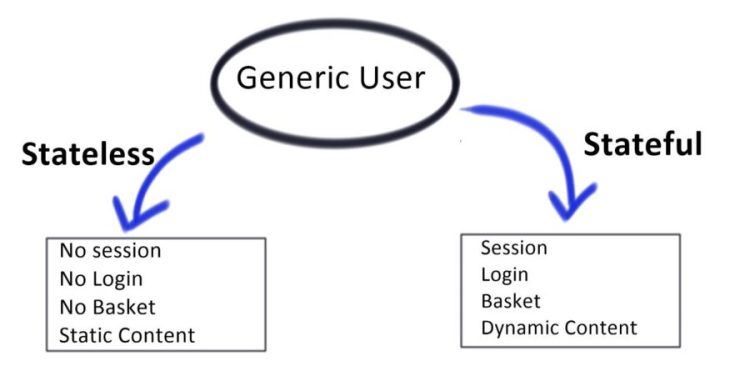
1. **The book states, “Interactions with an MVC application follow a natural cycle of user actions and view updates, where the view is assumed to be stateless.” What does it mean for the view to be stateless?**

Stateless applications don’t store data, whereas stateful applications require backing storage. Any data that flows via a stateless service is typically transitory and the state is stored only in a separate back-end service like a database. A stateless view means that the messages on the screen can be separated from the user’s stored data.

[](https://www.google.com/url?sa=i&url=https%3A%2F%2Finterviewbubble.com%2Fwhat-is-the-difference-between-stateful-and-stateless%2F&psig=AOvVaw3Qb1YWO3DvDqpivaZWxkkI&ust=1583780418662000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCJCo34fIi-gCFQAAAAAdAAAAABAD)

1. **The book identifies two kinds of models. Briefly describe each of them.**

The View model transfers data from controller to view. The Domain model contains data on the business side, along with the operations, transformations, and rules for creating, storing, and manipulating the data.

1. **Give an example of separation of concerns from your own life experience. This should be a simple, everyday example.**

My car is a good example of separation of concerns. For example, the primary UI consisting of steering wheel and pedals is relatively straight forward. However, the “controller” (engine) under the hood is visually separated from the UI, and I don’t need to understand the “model” of how all the machinery and belts work in order to effectively drive the car.

1. **What is a view engine?**

The view engine processes user requests and returns the required view. It is responsible for creating HTML from your views.

1. **The book notes that the three-tier structure, or n-tier model, is “the most widely used pattern for business applications.” Why do you think that this is true? An answer like, “Because it works well,” is not a sufficient answer to this question.**

The three-tier structure makes it easier to perform automated unit tests. The segmentation breaks up a complex system into smaller pieces which makes it easier to manage and edit code without impacting other segments.

1. **This question requires some outside research. When we study UWP, you will see that the UWP design pattern is the Model-View-ViewModel (MVVM). What is the difference between MVC and MVVM that makes the first good for ASP.NET MVC and the second good for UWP?**

Model-View-ViewModel (MVVM) is a UI architectural design pattern for decoupling UI and non-UI code. With MVVM, you define your UI declaratively in XAML and use data binding markup to link it to other layers containing data and commands. MVC is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts Model, View and Controller.

1. **Describe the two parts of the dependency injection (DI) design pattern.**

Dependency injection is basically providing the objects that an object needs (its dependencies) instead of having it construct them itself. Service and client.

1. **Give an example of loose coupling from your own life experience. This should be a simple, everyday example.**

The military organizational structure is a loose coupling. An aviation brigade knows very little about an artillery brigade. However, the aviation brigade usually has one artillery officer who is like the API that allows the aviation brigade to interact with the artillery brigade.

1. **What are the two types of testing discussed in the book?**

Integration and functional testing.

1. **What are the seven steps of the test-driven development (TDD) workflow, as stated in the book?**

Add a test, run all tests and see if the new test fails, write the code, run tests, refactor code, repeat.