**In which scenario we will use singleton pattern. As I know one common reason for this is same instance will have scope for entire application. but what is other?**

Logger Classes   
The Singleton pattern is used in the design of logger classes. This classes are usually implemented as a singletons, and provides a global logging access point in all the application components without being necessary to create an object each time a logging operations is performed.   
  
Configuration Classes   
The Singleton pattern is used to design the classes which provides the configuration settings for an application. By implementing configuration classes as Singleton not only that we provide a global access point, but we also keep the instance we use as a cache object. When the class is instantiated( or when a value is read ) the singleton will keep the values in its internal structure. If the values are read from the database or from files this avoids the reloading the values each time the configuration parameters are used.   
  
Accesing resources in shared mode   
It can be used in the design of an application that needs to work with the serial port. Let's say that there are many classes in the application, working in an multi-threading environment, which needs to operate actions on the serial port. In this case a singleton with synchronized methods could be used to be used to manage all the operations on the serial port.   
  
Factories implemented as Singletons   
Let's assume that we design an application with a factory to generate new objects(Acount, Customer, Site, Address objects) with their ids, in an multithreading environment. If the factory is instantiated twice in 2 different threads then is possible to have 2 overlapping ids for 2 different objects. If we implement the Factory as a singleton we avoid this problem. Combining Abstract Factory or Factory Method and Singleton design patterns is a common practice.