Unit testing using Mock objects and dependency injection

Agenda

1. Unit testing
2. Mock objects
3. Dependency injection
4. Automated dependency injection

Unit Testing

1. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use.
2. A unit smallest amount of testable code. Often a single method/function

Features of good unit tests

1. They should be fast
2. They should never use
    1. A database
    2. An app server (or server of any kind)
    3. File/Network I/O or file system

Benefits of unit testing

1. Facilitates change
    1. Unit testing allows the programmer to refactor code at a later date, and make sure the module still works correctly (e.g., in regression testing). Whenever a change causes a fault, it can be quickly identified and fixed.
2. Documentation
    1. Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit and how to use it can look at the unit tests to gain a basic understanding of the unit's API.
3. Design improvements

Mock objects

1. Mock objects are simulated objects that mimic the behavior of real objects
2. If an object has any of the following characteristics, it may be useful to use a mock object in its place:
    1. supplies non-deterministic results (e.g. the current time or the current temperature);
    2. has states that are difficult to create or reproduce (e.g. a network error);
    3. is slow (e.g. a complete database, which would have to be initialized before the test);

Dependency Injection

1. Dependency injection means giving an object its instance variables.
2. Manual
    1. Simple: Nothing to learn, no dependencies.
    2. No reflection magic: In IDE it is easy to find out who calls the constructors.
    3. Even developers who do not understand DI can follow and contribute to projects.
3. Automated
    1. Using manual caller has to know about all the dependencies, using automated DI it will inject all the dependencies so caller do not have to know about dependencies

Why developers hate unit tests

1. Have to write more code
2. Wants to jump to cool new feature quickly
3. requires them to face their own potentially imperfect code