Mockito

A simple, powerful mock unit testing framework for Java

Motivation

- Database example
- If tests don't touch database …
  - Important modules remain untested
- If tests touch database …
  - Performance problems
  - Inconsistency
  - Unit test side effects

Desired Unit Test Qualities

- Atomic
  - Well-established code boundaries
- Order Independent and Isolated
  - No side effects
- Intention Revealing
  - Atomicity
- Easy to setup and run
  - Dependencies
  - Application environment
- Easy to implement
- Runs quickly

What is Mockito?

- Unit test outline
  - Setup
  - Test / Execute
  - Verify
- Test component use, in addition to output
  - jUnit: *assertEquals*
  - Mockito: *verify*
- Third party library
  - External JAR, add to project dependencies
  - Compatible with jUnit

Basic Usage

```
import static org.mockito.Mockito.*;

@Test
public function sanityCheck() {

  // Setup
  List mockedList = mock(List.class);

  // Test
  mockedList.add("one");
  mockedList.clear();

  // Verify
  verify(mockedList).add("one");
  verify(mockedList).clear();
}
```

- Create mock object
  - *Any* class can be mocked
- No class implementation invoked
  - Object records calls
- Verification
  - Checks calls on object
    - Parameters stored
  - Order not enforced by default

Stubbing

Code

```
@Test
public testWithStubbed() {

  LinkedList mockedList = mock(
    LinkedList.class
  );

  when(mockedList.get(0)).thenReturn(
    "first"
  );
  when(mockedList.get(1)).thenThrow(
    new RuntimeException()
  );

  System.out.println(mockedList.get(0));

  System.out.println(mockedList.get(1));

  System.out.println(mockedList.get(999));

  verify(mockedList).get(0);
}
```

Notes

- Mock objects by default
  - Do nothing when a method is called
  - Return *null*
- Stubbing allows adding implementation from unit test
  - Stubbing can be specific to arguments

Matchers

Code

```
@Test
public testWithMatchers() {

  when(
    mockedList.get(anyInt())
  ).thenReturn(
    "element"
  );

  when(
    mockedList.contains(argThat(isValid()))
  ).thenReturn(
    "element"
  );

  System.out.println(mockedList.get(999));

  verify(mockedList).get(anyInt());
}
```

Notes

- Allows stubbing depending on argument types
- Used for both stubbing and verifying
- Multiple types
  - Many built-in matchers
  - Allows user-defined matchers
- Argument matches are all or none

Spying on Real Objects

Code

```
@Test
public testWithSpied() {

  List list = new LinkedList();
  List spy = spy(list);

  when(spy.size()).thenReturn(100);

  spy.add("one");
  spy.add("two");

  System.out.println(spy.get(0));

  System.out.println(spy.size());

  verify(spy).add("one");
  verify(spy).add("two");

}
```

Notes

- Why?
  - Keep an object's original implementation
- Spied objects
  - Behave like real objects
  - Verification with implementation
  - Allows stubbing

Assignment Usage

- Using an external library
- Testing Network I/O
  - Slow
    - Touches network
  - Side effects
    - Non-isolated unit tests
  - Unnecessary
    - Networking classes are well-tested
- **Don't add JAR to SVN**

```java
@Test
public testControllerSendsMoveMessage() {

  // Setup
  NetworkFacade network = mock(NetworkFacade.class);
  Controller controller = new Controller(network);
  ... // start a game, other initialization
  Command movePieceCommand = new MovePieceCommand(
    piece, originalSquare, newSquare
  );

  // Test
  controller.movePiece(piece, originalSquare, newSquare);

  // Verify
  verify(movePieceCommand).execute();
  verify(network).sendMoviePieceCommand(
    argThat(new CommandMatcher()) // A matcher for commands
  )
}
```

Summary

- Motivation
  - Database example
- Mocking frameworks
- Mockito
  - Basic usage
  - Stubbing
  - Matchers
  - Spying
- Assignment usage
  - Network I/O

Questions?

Jon Tedesco

References

- Mockito
  - http://stackoverflow.com/questions/1414032/why-create-mock-objects
  - http://stackoverflow.com/questions/1414032/why-create-mock-objects
  - http://code.google.com/p/mockito/
  - http://code.google.com/p/mockito/wiki/MockitoVSEasyMock
  - http://docs.mockito.googlecode.com/hg/latest/org/mockito/Mockito.html
  - http://code.google.com/p/mockito/wiki/FeaturesAndMotivations