

Ruby on Rails Unit Testing Assertions

BASIC ASSERTIONS

assert(boolean, message)

```
assert(person.name == "John", "Name was expected to be John.")
assert(item.errors.invalid?(:price))
```

assert_equal(expected, actual, message)

```
assert_equal(person.name, "John", "Name was expected to be John.")
assert_equal("can't be empty", product.errors.on(:price))
```

assert_not_equal(expected, actual, message)

```
assert_not_equal(person.name, "Mary", "Name was Mary and it should not be.")
assert_not_equal("is not a number", product.errors.on(:price))
```

assert_raise(Exception, message) { block... }

```
assert_raise(ZeroDivisionError, "Cannot divide by zero!") { 100 / 0 }
assert_raise(ActiveRecord::RecordNotFound) { Product.find(bad_id) }
```

assert_nothing_raised(Exception, message) { block... }

```
assert_nothing_raised(ZeroDivisionError) { 100 / [0,1].max }
assert_nothing_raised(ActiveRecord::RecordNotFound) { Product.find(good_id) }
```

assert_nil(object, message)

```
assert_nil( product, "Expected product to be nil." )
assert_nil( Wine.find(:first, :conditions => 'id = 1000') )
```

assert_not_nil(object, message)

```
assert_not_nil( product, "Product should not be nil." )
assert_not_nil( Wine.find(:first, :conditions => 'id = 1') )
```

assert_valid(activerecord_object)

```
same as: assert(object.valid?)
assert_valid(@person)
assert_valid( Wine.find(1) )
```

flunk(message)

```
always fails immediately; same as: assert(false, message)
flunk("Quantity should not be greater than 100") if quantity > 100
flunk("Either user or account should be valid") unless user.valid? || account.valid?
```

Ruby on Rails Unit Testing Assertions (continued)

ADVANCED ASSERTIONS

assert_match(pattern, string, message)

```
assert_match(/^\d,\d{3},\d{3}$/, "1,000,000", "Should match this format.")
```

assert_no_match(pattern, string, message)

```
assert_no_match(/\d{3},\d{2}$/, "1,000,000", "Should not match this format.")
```

assert_in_delta(expected_float, actual_float, delta, message)

```
assert_in_delta(100.0, price, 20.0, "Price should be between 80.00 and 120.00")
```

```
assert_in_delta(2, length, 1, "Length should be 1-3 feet.")
```

assert_instance_of(klass, object, message)

```
assert_instance_of( User, person, "person should be an instance of User" )
```

assert_kind_of(klass, object, message)

```
assert_kind_of( User, person, "person should be a kind of User" )
```

```
assert_kind_of( Class, User, "User should be a kind of Class" )
```

assert_respond_to(object, symbol, message)

Instances only respond to instance methods, classes only respond to class methods

```
assert_respond_to( person, :full_name, "No response to full_name" )
```

```
assert_respond_to( User, :custom_find, "No response to custom_find" )
```

assert_throws(expected_symbol, message) { block... }

```
assert_throws(:done, "Array should be empty") { throw :done if [].empty? }
```

assert_nothing_thrown(message) { block... }

```
assert_nothing_thrown("Array should not be empty") { throw :done if [1].empty? }
```

Ruby on Rails Unit Testing Assertions (continued)

RARE ASSERTIONS & DEFAULT ERROR MESSAGES

assert_same(expected, actual, message)

same as: assert_equal(expected, actual)

```
assert_same( person.name, "John")
```

assert_not_same(expected, actual, message)

same as: assert_not_equal(expected, actual)

```
assert_not_same( person.name, "Mary")
```

assert_operator(object1, operator, object2, message)

same as: assert(object1.operator(object2))

```
assert_operator( 1000, :<, 2000, "Expected 1000 to be less than 2000" )
```

```
assert_operator( user, :old_enough?, Time.now(), "User should be old enough")
```

assert_send([receiver, symbol, arg1, arg2], message)

same as: assert(receiver.message(arg1, arg2))

```
assert_send([product, :decrement_inventory, qty], "Decrement should succeed")
```

From: /activerecord/lib/active_record/validations.rb

```
@@default_error_messages = {  
  :inclusion           => "is not included in the list",  
  :exclusion           => "is reserved",  
  :invalid            => "is invalid",  
  :confirmation       => "doesn't match confirmation",  
  :accepted           => "must be accepted",  
  :empty              => "can't be empty",  
  :blank              => "can't be blank",  
  :too_long           => "is too long (maximum is %d characters)",  
  :too_short          => "is too short (minimum is %d characters)",  
  :wrong_length       => "is the wrong length (should be %d characters)",  
  :taken              => "has already been taken",  
  :not_a_number       => "is not a number",  
  :greater_than       => "must be greater than %d",  
  :greater_than_or_equal_to => "must be greater than or equal to %d",  
  :equal_to           => "must be equal to %d",  
  :less_than          => "must be less than %d",  
  :less_than_or_equal_to => "must be less than or equal to %d",  
  :odd                => "must be odd",  
  :even               => "must be even"  
}
```