What happens when you run `manage.py test`?

Adam Johnson

A black box?

- You write your tests
- Test runner runs them
- How?

🤷‍♀️ Why learn this?

- Middleware → change cookies, set headers, log requests, …
- Test runner → load tests differently, change settings, block HTTP requests, …

✅ Emoji Output

```
$ python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
💥❎❌⏭✅✅✅✳

...

----------------------------------------------------------------------
Ran 8 tests in 0.003s

FAILED (failures=1, errors=1, skipped=1, expected failures=1,
unexpected successes=1)
Destroying test database for alias 'default'...
```

🔍 Dissecting a Test Run

```
from django.test import TestCase


class ExampleTests(TestCase):
    def test_one(self):
        pass
```

Run

```
$ python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default'...
```

📣 Verbose

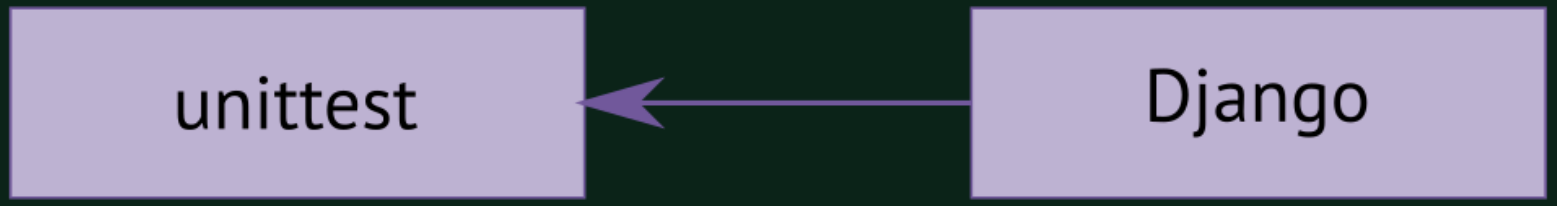
```
$ python manage.py test -v 3
Creating test database for alias 'default' ('file:memorydb_de
Operations to perform:
  Synchronize unmigrated apps: core
  Apply all migrations: (none)
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
Running migrations:
  No migrations to apply.
System check identified no issues (0 silenced).
test_one (example.tests.test_example.ExampleTests) ... ok

----------------------------------------------------------------
Ran 1 test in 0.004s

OK
Destroying test database for alias 'default' ('file:memorydb_
```

Process

1. Create the test databases.
2. Migrate the databases.
3. Run the system checks.
4. Run the tests.
5. Report on the test count and success/failure.
6. Destroy the test databases.


unittest
Django

`test` management command

```
def handle(self, *test_labels, **options):
    TestRunner = get_runner(settings, options['testrunner'])
    ...
    test_runner = TestRunner(**options)
    ...
    failures = test_runner.run_tests(test_labels)
    ...
```

- Short command, < 100 lines

🔍 DiscoverRunner class

```
class DiscoverRunner:
    """A Django test runner that uses unittest2 test
    discovery."""

    test_suite = unittest.TestSuite
    parallel_test_suite = ParallelTestSuite
    test_runner = unittest.TextTestRunner
    test_loader = unittest.defaultTestLoader
```

🔍 run_tests()

```
def run_tests(self, test_labels, extra_tests=None, **kwargs):
    self.setup_test_environment()
    suite = self.build_suite(test_labels, extra_tests)
    databases = self.get_databases(suite)
    old_config = self.setup_databases(aliases=databases)
    self.run_checks(databases)
    result = self.run_suite(suite)
    self.teardown_databases(old_config)
    self.teardown_test_environment()
    return self.suite_result(suite, result)
```

🔍 build_suite()

```
def build_suite(self, test_labels=None, extra_tests=None,
                **kwargs):
    suite = self.test_suite()
    test_labels = test_labels or ['.']

    for label in test_labels:
        tests = self.test_loader.loadTestsFromName(label)
        suite.addTests(tests)

    if self.parallel > 1:
        suite = self.parallel_test_suite(suite, ...)

    return suite
```

test_suite, parallel_test_suite, test_loader

🔍 run_suite()

```
def run_suite(self, suite, **kwargs):
    kwargs = self.get_test_runner_kwargs()
    runner = self.test_runner(**kwargs)
    return runner.run(suite)
```

test_runner

unittest.TextTestRunner

```
class TextTestRunner(object):
    """A test runner class that displays results in textual
    form. It prints out the names of tests as they are run,
    errors as they occur, and a summary of the results at the
    end of the test run.
    """
    resultclass = TextTestResult

    def __init__(self, ..., resultclass=None, ...):
```

✌️ Two ways to customize

- Override `test` management command
- Override `DiscoverRunner` with the `TEST_RUNNER` setting

Superfast Runner

- Example simplest customization
- Make tests execute instantly

Superfast Runner

Custom test runner:

```
# example/test.py
from django.test.runner import DiscoverRunner


class SuperFastTestRunner(DiscoverRunner):
    def run_tests(self, *args, **kwargs):
        print("All tests passed! A+")
        failures = 0
        return failures
```

Superfast Runner

Settings file:

```
TEST_RUNNER = "example.test.SuperFastTestRunner"
```

Superfast Runner

Now very fast:

```
$ python manage.py test
All tests passed! A+
```

✅ Emoji Output

- Override `DiscoverRunner` with custom subclass
- Override the `TextTestResult` class used with a custom subclass

🔍 DiscoverRunner .get_test_runner_kwargs()

```
def get_test_runner_kwargs(self):
    return {
        'failfast': self.failfast,
        'resultclass': self.get_resultclass(),
        'verbosity': self.verbosity,
        'buffer': self.buffer,
    }
```

🔍 DiscoverRunner .get_resultclass()

```
def get_resultclass(self):
    if self.debug_sql:
        return DebugSQLTextTestResult
    elif self.pdb:
        return PDBDebugResult
```

Replace implicit `None` with our subclass

✅ Emoji output

```
class EmojiTestRunner(DiscoverRunner):
    def get_resultclass(self):
        klass = super().get_resultclass()
        if klass is None:
            return EmojiTestResult
        return klass
```

✅ Emoji output

```
class EmojiTestResult(unittest.TextTestResult):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # If the "dots" style was going to be used,
        # show emoji instead
        self.emojis = self.dots
        self.dots = False

    def addSuccess(self, test):
        super().addSuccess(test)
        if self.emojis:
            self.stream.write('✅')
            self.stream.flush()

    ... # One method per result type
```

✅ Emoji Output

```
$ python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
💥❎❌⏭✅✅✅✳

...

----------------------------------------------------------------------
Ran 8 tests in 0.003s

FAILED (failures=1, errors=1, skipped=1, expected failures=1,
unexpected successes=1)
Destroying test database for alias 'default'...
```

 No Composition

- Classes-referring-to-classes simple but limited
- Debug SQL or PDB alongside Emoji?

No Composition

- Not composable = no plugins
- Two extensions:
  - unittest-xml-reporting
  - django-slow-tests

 pytest

- Composition with hooks
- Over 700 plugins

Thank you! 

- Adam Johnson
- @adamchainz on GitHub & Twitter
- me@adamj.eu
- github.com/adamchainz/talk-what-happens-when-you-run-manage.py-test