TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE:

- Modelos de prueba
- Clases de prueba en Django
- Pruebas Client
- ¿Cómo implementar las pruebas preinstaladas?

USANDO LOS MODELOS DE PRUEBA

Las pruebas automatizadas pueden actuar como el primer "usuario" del mundo real de su código, lo que le obliga a ser riguroso a la hora de definir y documentar correctamente cómo debe comportarse su sitio web. A menudo, éstas son la base de sus ejemplos de código y documentación. Por ello algunos procesos de desarrollo de software comienzan con la definición e implementación de la prueba, y luego el código se escribe para que coincida con el comportamiento requerido (por ejemplo: desarrollo basado en pruebas y en comportamiento).

MODELOS DE PRUEBAS

Hay numeroso tipos, niveles, y clasificaciones de pruebas y enfoques de pruebas. Las pruebas automáticas más importantes son:

- **Pruebas unitarias:** verifican el comportamiento funcional de un componente individual, a menudo de una clase y su nivel funcional.
- **Pruebas de regresión:** reproducen errores históricos. Cada prueba es inicialmente ejecutada para verificar que el error ha sido corregido, y éstas son ejecutadas de nuevo para asegurarnos que los errores no fueron reintroducidos con los futuros cambios en el código.
- **Pruebas de integración:** verifican cómo funcionan los grupos de componentes cuando se usan juntos. Las pruebas de integración son conscientes de las interacciones requeridas entre componentes, pero no necesariamente de las operaciones internas de cada uno. Pueden cubrir agrupaciones simples de componentes, hasta todo el sitio web.

Otros tipos comunes de pruebas incluyen pruebas de caja negra, caja blanca, manuales, automatizadas, canarias, de humo, de conformidad, de aceptación, funcionales, de rendimiento, de carga, y de esfuerzo.

Probar un sitio web es una tarea compleja, pues está compuesto por varias capas de lógica, desde el manejo de solicitudes a nivel HTTP, modelos de consultas, hasta la validación y procesamiento de formularios, y la representación de plantillas.

Django proporciona un marco de prueba con una pequeña jerarquía de clases que se basan en la librería `unittest` estándar Python. A pesar del nombre, este marco de prueba es adecuado tanto para pruebas unitarias como de integración. El marco de Django agrega métodos y herramientas API para ayudar a probar su comportamiento web y específico. Éstos le permiten simular solicitudes, insertar datos de prueba, e inspeccionar la salida de su aplicación. Django también proporciona una API (`LiveServerTestCase`) y herramientas para usar diferentes frameworks de pruebas; por ejemplo: puede integrarse con el framework Selenium para simular la interacción de un usuario con un navegador en vivo.

Para escribir una prueba, ésta se deriva de cualquiera de las clases base de prueba de Django (o `unittest`) (`SimpleTestCase`, `TransactionTestCase`, `TestCase`, `LiveServerTestCase`), y luego se escriben métodos separados para verificar que la funcionalidad específica esté como se esperaba (las pruebas usan métodos `"assert"` para probar que las expresiones dan valores `True` o `False`, o que dos valores son iguales, ente otros). Cuando inicia una ejecución de prueba, el marco ejecuta los métodos de prueba elegidos en sus clases derivadas. Los métodos de prueba se ejecutan de forma independiente, con un comportamiento común de configuración y/o desmontaje definido en la clase, tal como se muestra a continuación:

```
class YourTestClass(TestCase):
    def setUp(self):
        #Setup run before every test method.
        pass
    def tearDown(self):
        #Clean up run after every test method.
        pass
    def test_something_that_will_pass(self):
        self.assertFalse(False)
    def test_something_that_will_fail(self):
        self.assertTrue(False)
```

La mayoría de las pruebas se corren `django.test.TestCase`. Esta clase de prueba crea una base de datos limpia antes de que se ejecuten sus pruebas, y aplica cada función de prueba en su propia transacción. La clase también posee una prueba `Client`, la cual se puede utilizar para simular la interacción de un usuario con el código en el nivel de vista. Cuenta con:

- `test fixture` - Preparación necesaria para realizar las pruebas.
- `test case` - Caso concreto e individual que se quiere probar.
- `test suite` - Conjunto de casos de prueba.
- `test runner` - Componente que ejecuta los tests.

La aplicación crea un fichero tests.py por defecto. Si se necesita más complejidad, se pueden crear nuevos scripts de formato `test*.py`, y una vez escritos, se ejecutan desde la terminal.

¿CÓMO USAR Y REVISAR LAS APLICACIONES PREINSTALADAS?

A la hora de crear una aplicación de Django, una colección de archivos de código fuente, incluyendo modelos y vistas, que conviven en un solo paquete de Python y representen una aplicación completa de Django. Se crea un sitio web "esqueleto", que luego se llena con configuraciones específicas del sitio, urls, modelos, vistas, y plantillas.

El proceso es sencillo:

- Usar la herramienta `django-admin` para crear la carpeta del proyecto, los ficheros de plantillas básicos, y el script de gestión del proyecto (`manage.py`).
- Usar `manage.py` para crear una o más aplicaciones.
- Registrar las nuevas aplicaciones para incluirlas en el proyecto.
- Conectar el mapeador URL de cada aplicación.

Para revisar las aplicaciones pre-instaladas debemos ir a settings.py bajo la configuración llamada `INSTALLED_APPS`.