Fedések

# Fogalom

A kódlefedettség egy egyszerű statisztika, amely méri a teszthalmaz által ellenőrzött teljes kódsorok számát. Különböző metrikákat használva kiszámítja az alkalmazás forráskódjában sikeresen lefutó tesztekben szereplő kódsorok teljes számát, általában százalékos formában kifejezve.

Ez annak felel meg, hogy elosztod a teszt alatt futó teljes kódsorok számát a tesztelésre vonatkozó kódsorok összegével (ha a statement coverage-et számolsz). Majd a eredményt megszorozod 100-zal, hogy százalékos lefedettséget kapj:

Kódlefedettség = (tesztelt kódsorok összessége / tesztelésre vonatkozó kódsorok összessége) \* 100

Például, ha egy osztályt tesztelsz, amely tartalmazza az alkalmazásod forráskódjában lévő 100 kódsort, akkor a tesztelés során az összes 100 kódsort tesztnek veted alá. Ha a teszt után ebből az osztályból kihagysz 40 sort, akkor azt mondhatod, hogy a tesztgyűjteményed 60%-os lefedettséget ért el.

Ebben az esetben a ténylegesen tesztelt kódsorok száma 60 volt, míg a tesztelésnek kitett sorok száma 100. Lehetséges, hogy a kihagyott 40 sorban rejtett hibák vannak, amiket nem szeretnél kiadni a termékkel. Ezért szeretnénk javítani a teszteket és a forráskódot, hogy növeljük a lefedettséget.

## Lehetséges 100 százalékos kódlefedettség?

Igen.

De a 100 százalékos kódlefedettség nem elegendő ok arra, hogy kijelentsük, hogy az alkalmazás hibamentes.

Bár idő- és erőforrás-igényes lehet, célszerű folyamatosan javítani a kódodat, ahelyett hogy törekednél a tökéletes 100% kódlefedettségre. A kódlefedettség lényege az, hogy felfedje a rejtett hibákat, és betekintést nyújtson abba, hogy mivel lehet még a teszthalmazt kiegészíteni. Ezáltal segít abban, hogy növeld a termék teszteltségét és ezáltal a minőségét és az abba vetett bizalmat.

## Kódlefedettség vs. Tesztlefedettség

Sokan, beleértve a tesztelőket és fejlesztőket is, szinonimaként használják a kód- és tesztlefedettséget. Habár ezek a fogalmak szorosan összefüggenek egymással, vannak néhány különbség közöttük.

### Tesztlefedettség

A tesztlefedettség a fekete doboz teszteléshez tartozik. Itt azt ellenőrizzük, hogy az adott szoftver zavartalanul működik az összes szükséges funkcionalitással. Amikor ezt a lefedettségi típust elemezzük, nem kell megértenünk a tesztelésre kerülő szoftver forráskódjának belső működési mechanizmusait.

Például az alkalmazás tervezése általában felsorolja azokat a konkrét funkciókat, amelyeket teljes mértékben teljesítenie kell. Mint minőségbiztosítási mérnök, szeretnéd biztosítani, hogy a tesztgyűjteményed lefedje az összes szükséges funkcionalitást.

### Kódlefedettség

Ahogy korábban említettük, a kódlefedettség ellenőrzi, hogy milyen mértékben fedi le a teszthalmaz a forráskódot. Ez egy white-box megközelítés, amely megköveteli tőled, hogy megértsd az alkalmazást tesztelő forráskód alapvető mechanizmusait.

# Kódlefedettség típusai

A kódlefedettség számos különböző módszert használhat. Az alábbiakban néhányat kifejtünk.

### Elágazás lefedettség (branch coverage)

A branch lefedettség ellenőrzi, hogy egy teszt lefedi-e a kód végrehajtási útvonalait. Olyan ellenőrzéseket tartalmaz, amelyek megvizsgálják, hogy a kód minden ága, beleértve a ciklusokat és feltételeket, legalább egyszer lefut-e a tesztelés során.

A branch lefedettség inkluzív, hiszen minden végrehajtási útvonalat ellenőriz, hogy ne maradjon ellenőrizetlenül. Például, ha olyan kódot írsz, amely csak akkor hajt végre egy műveletet, ha A > B, a branch lefedettség alkalmazása először ellenőrzi a feltételt (A > B) igaz állapotban, majd csekkolja az adott feltétel alatti minden utasítást (műveletet) is.

Például lehetséges, hogy a teszt nem fedi le az az esetet, amikor a feltétel (A > B) hamis. Az ág lefedéséhez szükség van olyan teszt futtatására, amelyben a logikai feltétel mind hamis.

### Utasításlefedettség (statement coverage)

Az utasításlefedettség specifikusabb. Míg a branch lefedettség csak a kód útvonalában található utasításokat vagy műveleteket ellenőrzi, az utasításlefedettség nem veszi figyelembe az olyan szülő kódot, amely a művelet végrehajtását előidézi.

Ez azt jelenti, hogy ha egy teszt nem fedi le az ágat, amely meghatározza egy művelet végrehajtását (ha az ágazati lefedettség nulla), akkor az utasításlefedettség is nulla erre az ágra vonatkozóan.

Például, ha A > B hamis, az adott ágon az utasítások nem fognak lefutni a teszt során. Azt mondhatjuk, hogy az utasításlefedettség részben függ az elágazás lefedettségtől.

Azonban néha előfordulhat, hogy egy teszt nem fedi le az ág alatt található utasításokat (műveleteket), ha valahol hiba van. (pl exception)

### Döntéslefedettség

A döntéslefedettség az elágazásfedés egy részhalmaza, amely ellenőrzi, hogy egy teszt lefedi-e az összes feltételt a forráskódban. Míg az elágazásfedés minden kódutat ellenőriz, beleértve a ciklusokat is, a döntéslefedettség csak a logikai feltételeket ellenőrzi.

A természetéből adódóan a döntéslefedettség figyelem kívül hagyhat hamis feltételeket.

### Cikluslefedettség

Ez ellenőrzi tesztgyűjteményben legalább egyszer lefutó ciklusok százalékát.

### Függvénylefedettség

Ez ellenőrzi a tesztelés során végrehajtott függvények százalékát.

# coverage.py

A coverage.py könyvtár az egyik legelterjedtebben használt pytest kódlefedettségi jelentést készítő eszköz. Egy egyszerű Python modul, amely átfogó pytest kódlefedettségi jelentéseket generál táblázat formátumban. Használhatod parancssori segédprogramként, vagy beépítheted a tesztkódodba, mint egy API-t a lefedettség elemzésének generálásához.

Az API opciót ajánlott használni, ha el szeretnéd kerülni a terminálban futtatott parancsok ismétlését minden egyes lefedettség elemzés során.

* parancssor: <https://coverage.readthedocs.io/en/7.2.7/>
* API: <https://coverage.readthedocs.io/en/7.2.7/api.html#api>

Az alábbiakban látható a terminál parancs a kódlefedettség végrehajtásához pytest segítségével a coverage.py használatával:

coverage run -m pytest

A fenti parancs a "test"-el kezdődő nevű összes pytest teszt tesztfuttatást elindítja.

Az API használatához csak annyit kell tenned a jelentések generálása során, hogy megadod a célmappát a tesztkódodban. Az API ezt követően felülírja a mappa HTML jelentését a későbbi tesztek során.

Példa:

import coverage

import pytest

cov = coverage.Coverage()

cov.start()

pytest.main(['-x', 'api/tests.py', '-vv'])

cov.stop()

cov.report()