

## 2. Integration Tests

Integrationstests sind besonders relevant, wenn Sie Aufgaben an beteiligte Komponenten delegieren.

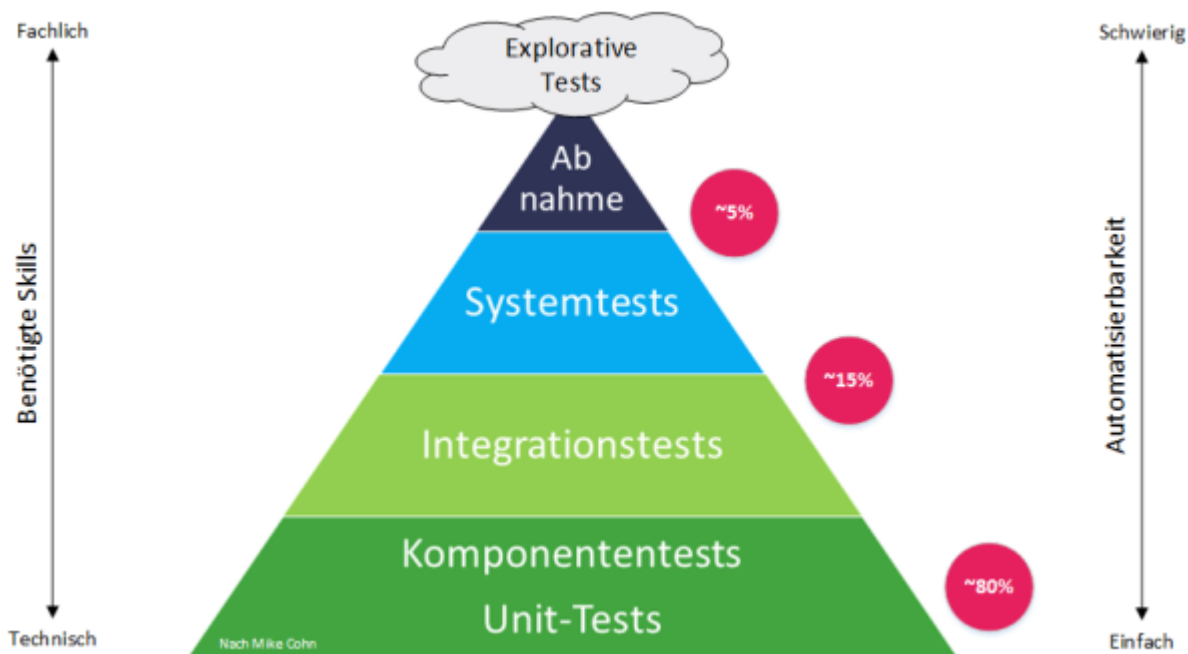
### Was sind Integration Tests?

Bisher haben Sie hauptsächlich etwas über Unit Tests gelernt. Komponententests sind eine grossartige Möglichkeit, vorhersehbaren und stabilen Code zu erstellen. Aber am Ende des Tages muss Ihre Anwendung funktionieren, wenn sie startet!

Ein **Integration Test** prüft das Zusammenspiel von mehrerer Komponenten einer Anwendung. Dafür kann es erforderlich sein, sich wie ein Konsument oder Benutzer der Anwendung zu verhalten (Black-Box Test), indem Sie:

- eine HTTP-REST-API aufrufen
- eine Python-API aufrufen
- einen Webdienst aufrufen
- eine Befehlszeile ausführen

### Integration Tests bilden 2. Stufe der Testpyramide



### Wie sind Integration Tests zu implementieren?

Jede dieser Arten von **Integration Tests** kann auf die gleiche Weise wie ein Komponententest geschrieben werden, indem dem *Arrange-Act-Assert* Muster gefolgt wird. Der wichtigste Unterschied besteht darin, dass Integration Tests mehrere Komponenten gleichzeitig prüfen und daher mehr Seiteneffekte haben als ein Komponententest. Ausserdem erfordern Integrationstests u.U. mehr Vorrichtungen, wie eine Datenbank, einen Netzwerk-Socket oder eine Konfigurationsdatei.

Durch den Verbund von mehreren Komponenten sowie den erwähnten Seiteneffekten wird das **FIRST Principle** verletzt, das Sie aus dem Kapitel über Unit Testing kennen. Dieses Prinzip gilt daher nicht Integration Tests. Aus diesem Grund empfiehlt es sich, Komponententests und Integrationstests zu trennen. Die Erstellung von Vorrichtungen, die für eine Integration wie eine Testdatenbank erforderlich sind, und die Testfälle selbst dauern oft viel länger als Unit Tests. Daher möchten Sie Integration Tests möglicherweise nur ausführen, bevor Sie sie in die Produktion übertragen, anstatt einmal bei jedem Commit.

Eine einfache Möglichkeit, Unit und Integration Tests zu trennen, besteht darin, sie einfach in verschiedene Ordner zu legen:

```
project/
├── my_app/
│   └── __init__.py
└── tests/
    ├── unit/
    │   ├── __init__.py
    │   └── test_sum.py
    └── integration/
        ├── __init__.py
        └── test_integration.py
```

---

Quellen: - [Writing Integration Tests](#)

From:  
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:  
[https://wiki.bzz.ch/modul/m320/learningunits/lu97/theorie/lu5-kapitel\\_2](https://wiki.bzz.ch/modul/m320/learningunits/lu97/theorie/lu5-kapitel_2)

Last update: **2024/03/28 14:07**

