자바와 JUnit을 활용한 실용주의 단위 테스트

Pragmatic Unit Testing in Java with JUnit

초판 발행 · 2019년 6월 30일

지은이 · 제프 랭어, 앤디 헌트, 데이브 토마스
옮긴이 · 유동환
발행인 · 이종원
발행처 · (주)도서출판 길벗
출판사 등록일 · 1990년 12월 24일
주소 · 서울시 마포구 월드컵로 10길 56(서교동)
대표 전화 · 02)332-0931 | **팩스** · 02)323-0586
홈페이지 · www.gilbut.co.kr | **이메일** · gilbut@gilbut.co.kr

기획 및 책임편집 · 안윤경(yk78@gilbut.co.kr) | **디자인** · 박상희 | **제작** · 이준호, 손일순, 이진혁
영업마케팅 · 임태호, 전선하, 지운집, 박성용 | **영업관리** · 김명자 | **독자지원** · 송혜란, 홍혜진

교정교열 · 김윤지 | **전산편집** · 박진희 | **출력 · 인쇄** · 예림인쇄 | **제본** · 예림바인딩

ISBN 979-11-6050-838-3 93000
(길벗 도서번호 006814)

정가 26,000원

독자의 1초까지 아껴주는 정성 길벗출판사

(주)도서출판 길벗 | **www.gilbut.co.kr**

페이스북 · www.facebook.com/gbitbook
예제소스 · https://github.com/gilbutITbook/006814

목 차

1부 단위 테스트의 기초 ····· 024

1장 첫 번째 JUnit 테스트 만들기 ····· 025

1.1 단위 테스트를 작성하는 이유 026

1.2 JUnit의 기본: 첫 번째 테스트 통과 028

1.2.1 프로젝트 설정 029

1.2.2 JUnit 테스트 좀 더 이해 032

1.2.3 JUnit 실행 033

1.3 테스트 준비, 실행, 단언 036

1.4 테스트가 정말로 뭔가를 테스트하는가? 038

1.5 마치며 039

2장 JUnit 진짜로 써 보기 ····· 041

2.1 테스트 대상 이해: Profile 클래스 043

2.2 어떤 테스트를 작성할 수 있는지 결정 046

2.3 단일 경로 커버 048

2.4 두 번째 테스트 만들기 051

2.5 @Before 메서드로 테스트 초기화 052

2.6 이제 어떤가? 056

2.7 마치며 057

3장 JUnit 단언 깊게 파기 ····· 059

3.1 JUnit 단언 060

3.1.1 assertTrue 061
3.1.2 assertThat은 명확한 값을 비교 062
3.1.3 중요한 햄크레스트 매처 살펴보기 065
3.1.4 부동소수점 수를 두 개 비교 068
3.1.5 단언 설명 069

3.2 예외를 기대하는 세 가지 방법 070

3.2.1 단순한 방식: 애너테이션 사용 071
3.2.2 옛 방식: try/catch와 fail 071
3.2.3 새로운 방식: ExpectedException 규칙 072
3.2.4 예외 무시 074

3.3 마치며 075

4장 테스트 조직 ····· 077

4.1 AAA로 테스트 일관성 유지 078

4.2 동작 테스트 vs 메서드 테스트 080

4.3 테스트와 프로덕션 코드의 관계 081

4.3.1 테스트와 프로덕션 코드 분리 082
4.3.2 내부 데이터 노출 vs 내부 동작 노출 084

4.4 집중적인 단일 목적 테스트의 가치 085

4.5 문서로서의 테스트 087

4.5.1 일관성 있는 이름으로 테스트 문서화 087
4.5.2 테스트를 의미 있게 만들기 089

4.6 @Before와 @After (공통 초기화와 정리) 더 알기 090

4.6.1 BeforeClass와 AfterClass 애너테이션 092

4.7 녹색이 좋다: 테스트를 의미 있게 유지 094
4.7.1 테스트를 빠르게 094
4.7.2 테스트 제외 096
4.8 마치며 097

2부 빠른 암기법 습득 ····· 098

5장 좋은 테스트의 FIRST 속성 ····· 099

5.1 FIRST: 좋은 테스트 조건 100
5.2 [F]IRST: 빠르다 101
5.3 F[I]RST: 고립시킨다 108
5.4 FI[R]ST: 좋은 테스트는 반복 가능해야 한다 109
5.5 FIR[S]T: 스스로 검증 가능하다 113
5.6 FIRS[T]: 적시에 사용한다 115
5.7 마치며 116

6장 Right-BICEP: 무엇을 테스트할 것인가? ····· 119

6.1 [Right]-BICEP: 결과가 올바른가? 120
6.2 Right-[B]ICEP: 경계 조건은 맞는가? 122
6.3 경계 조건에서는 CORRECT를 기억하라 127
6.4 Right-B[I]CEP: 역 관계를 검사할 수 있는가? 128
6.5 Right-BI[C]EP: 다른 수단을 활용하여 교차 검사할 수 있는가? 131
6.6 Right-BIC[E]P: 오류 조건을 강제로 일어나게 할 수 있는가? 132

6.7 Right-BICE[P]: 성능 조건은 기준에 부합하는가? 134
6.8 마치며 137

7장 경계 조건: CORRECT 기억법 ····· 139

7.1 [C]ORRECT: [C]onformance(준수) 141
7.2 C[O]RRECT: [O]rdering(순서) 143
7.3 CO[R]RECT: [R]ange(범위) 145
7.3.1 불변성을 검사하는 사용자 정의 매처 생성 149
7.3.2 불변 메서드를 내장하여 범위 테스트 150
7.4 COR[R]ECT: [R]eference(참조) 155
7.5 CORR[E]CT: [E]xistence(존재) 157
7.6 CORRE[C]T: [C]ardinality(기수) 158
7.7 CORREC[T]: [T]ime(시간) 161
7.8 마치며 163

3부 더 큰 설계 그림 ····· 164

8장 깔끔한 코드로 리팩토링하기 ····· 165

8.1 작은 리팩토링 166
8.1.1 리팩토링의 기회 167
8.1.2 메서드 추출: 두 번째로 중요한 리팩토링 친구 168
8.2 메서드를 위한 더 좋은 집 찾기 170
8.3 자동 및 수동 리팩토링 173

8.4 과한 리팩토링? 176
8.4.1 보상: 명확하고 테스트 가능한 단위들 178
8.4.2 성능 염려: 그러지 않아도 된다 178
8.5 마치며 180

9장 더 큰 설계 문제 ····· 181

9.1 Profile 클래스와 SRP 182
9.2 새로운 클래스 추출 186
9.3 명령–질의 분리 191
9.4 단위 테스트의 유지 보수 비용 194
9.4.1 자신을 보호하는 방법 195
9.4.2 깨진 테스트 고치기 196
9.5 다른 설계에 관한 생각들 198
9.6 마치며 202

10장 목 객체 사용 ····· 203

10.1 테스트 도전 과제 204
10.2 번거로운 동작을 스텁으로 대체 207
10.3 테스트를 지원하기 위한 설계 변경 211
10.4 스텁에 지능 더하기: 인자 검증 212
10.5 목 도구를 사용하여 테스트 단순화 215
10.6 마지막 하나의 단순화: 주입 도구 소개 218
10.7 목을 올바르게 사용할 때 중요한 것 220
10.8 마치며 222

11장 테스트 리팩토링 ····· 223

11.1 이해 검색 224
11.2 테스트 냄새: 불필요한 테스트 코드 226
11.3 테스트 냄새: 추상화 누락 228
11.4 테스트 냄새: 부적절한 정보 232
11.5 테스트 냄새: 부푼 생성 235
11.6 테스트 냄새: 다수의 단언 236
11.7 테스트 냄새: 테스트와 무관한 세부 사항들 238
11.8 테스트 냄새: 잘못된 조직 240
11.9 테스트 냄새: 암시적 의미 242
11.10 새로운 테스트 추가 244
11.11 마치며 245

4부 더 큰 단위 테스트 그림 ····· 246

12장 테스트 주도 개발 ····· 247

12.1 TDD의 주된 이익 249
12.2 단순하게 시작 250
12.3 또 다른 증분 추가 253
12.4 테스트 정리 256
12.5 또 다른 작은 증분 259
12.6 다수의 응답 지원: 작은 설계 우회로 262
12.7 인터페이스 확장 264
12.8 마지막 테스트들 268

12.9 문서로서의 테스트 270
12.10 TDD의 리듬 272
12.11 마치며 273

13장 까다로운 테스트 ····· 275

13.1 멀티스레드 코드 테스트 276
13.1.1 단순하고 똑똑하게 유지 277
13.1.2 모든 매칭 찾기 278
13.1.3 애플리케이션 로직 추출 279
13.1.4 스레드 로직의 테스트 지원을 위해 재설계 284
13.1.5 스레드 로직을 위한 테스트 작성 286
13.2 데이터베이스 테스트 288
13.2.1 고마워, Controller 289
13.2.2 데이터 문제 292
13.2.3 클린 룸 데이터베이스 테스트 293
13.2.4 controller를 목 처리 296
13.3 마치며 298

14장 프로젝트에서 테스트 ····· 299

14.1 빠른 도입 300
14.2 팀과 같은 편 되기 301
14.2.1 단위 테스트 표준 만들기 302
14.2.2 리뷰로 표준 준수 높이기 303
14.2.3 짝 프로그래밍을 이용한 리뷰 304

14.3 지속적 통합으로 수렴 305

14.4 코드 커버리지 307

14.4.1 커버리지는 어느 정도여야 하는가? 310

14.4.2 100% 커버리지는 진짜 좋은가? 311

14.4.3 코드 커버리지의 가치 312

14.5 마치며 313

부록 A 인텔리제이 IDEA와 넷빈즈에서 JUnit 설정 ····· 315

A.1 인텔리제이 IDEA 317

A.2 넷빈즈 322

찾아보기 327