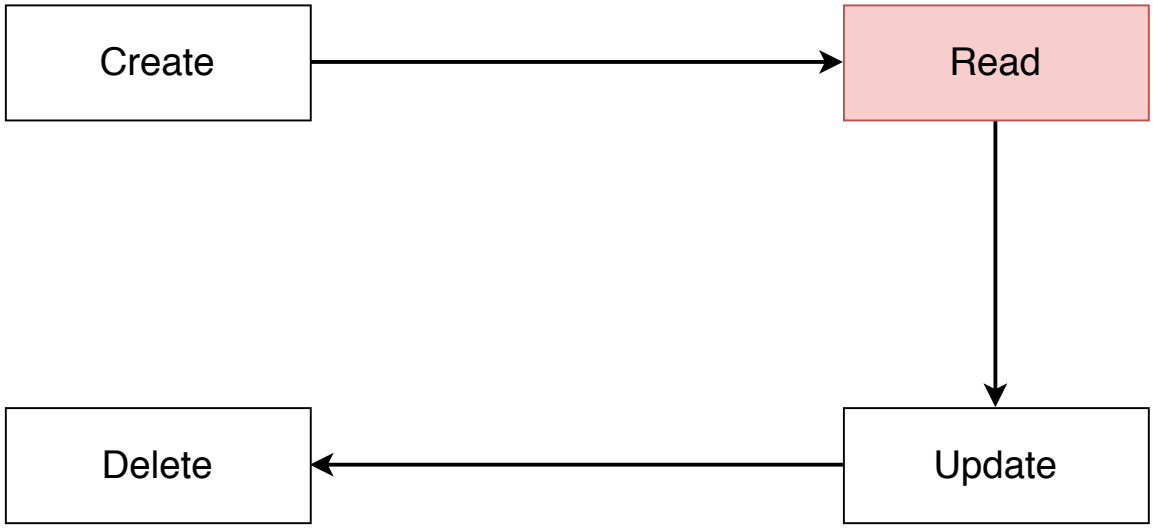
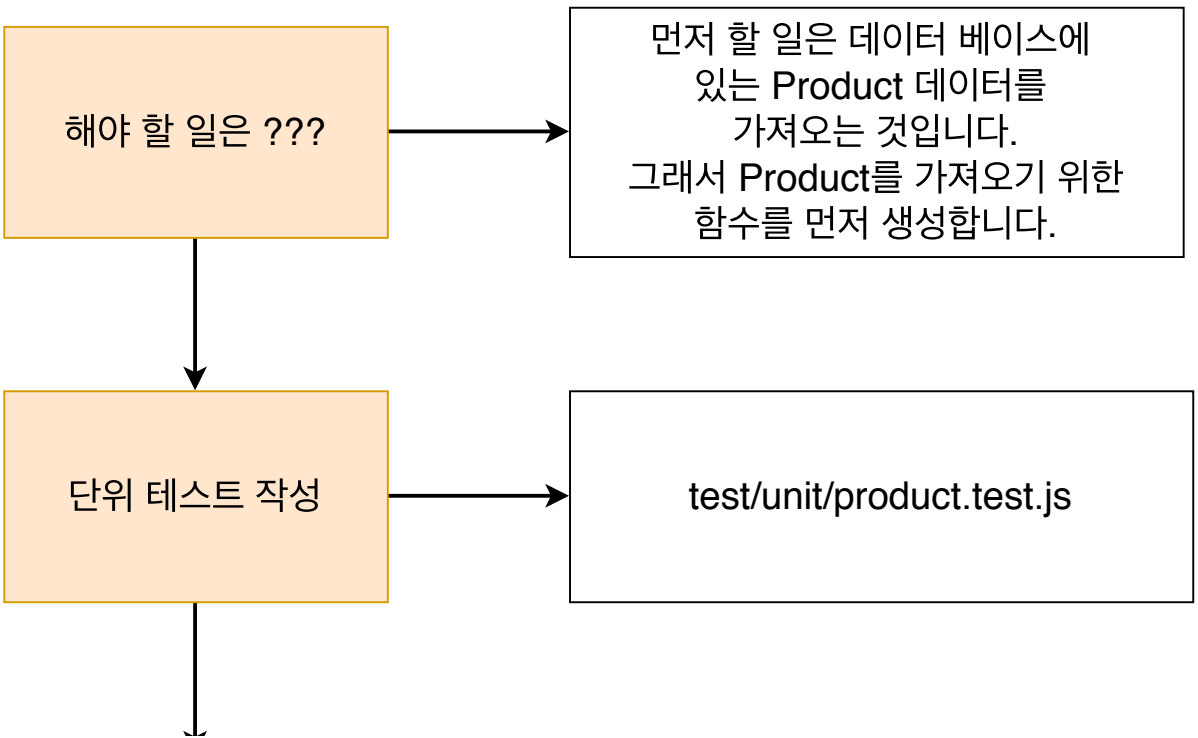
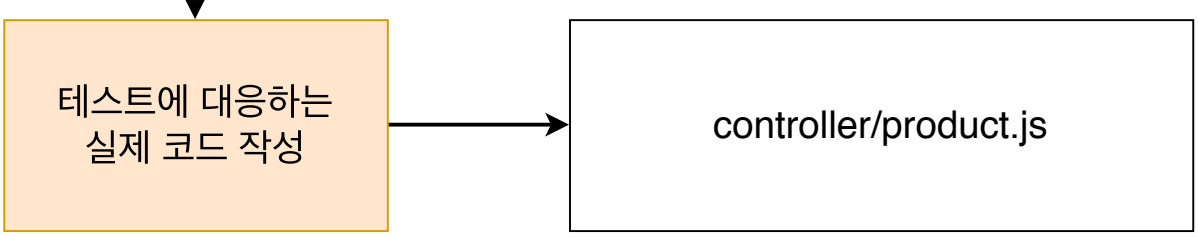
Read 시작

CRUD 작성 순서



Read 부분 소스 작성




테스트에 대응하는
실제 코드 작성
controller/product.js

getProducts 단위 테스트 작성

Read 부분 소스 작성

해야 할 일은 ???

→ getProducts 함수를 호출할 때 Product Model의 Find 메소드가 호출이 되는지 확인 해주기

단위 테스트 작성

```
it("should call Product.find({})", async () => {
    await productController.getProducts(req, res, next);
    expect(Product.find).toHaveBeenCalledWith({});
});
```

테스트에 대응하는 실제 코드 작성

```
exports.getProducts = async (req, res, next) => {
  await Product.find({});
};
```

해야 할 일은 ???

→ 데이터 베이스에서 Product 가져오기를 성공했다면 성공 Status(200)를 전달해주기

```
it("should return 200 response", async () => {
  await productController.getProducts(req, res, next);
```

단위 테스트 작성

```
  expect(res.statusCode).toBe(200);
  expect(res._isEndCalled()).toBeTruthy();
});
```

테스트에 대응하는
실제 코드 작성

```
exports.getProducts = async (req, res, next) => {
    await productModel.find({});
    res.status(200).send();
}
```

해야 할 일은 ???

데이터 베이스에서 가져온
Product 데이터들을
클라이언트에 전달해주기

단위 테스트 작성

```
it("should return json body in response", async () => {
  Product.find.mockReturnValue(allProducts);
  await productController.getProducts(req, res, next);
  expect(res._getJSONData()).toStrictEqual(allProducts);
});
```

테스트에 대응하는
실제 코드 작성

```
exports.getProducts = async (req, res, next) => {
    const allProducts = await Product.find({});
    res.status(200).json(allProducts);
};
```

getProducts 에러 처리 단위 테스트 작성

Read 부분 소스 작성

해야 할 일은 ???

→ Product 데이터를 가져올 때
에러가 나면
그것에 맞는 에러 처리를 해주기

단위 테스트 작성

```
it("should handle errors", async () => {
  const errorMessage = { message: "Error finding product dat
  const rejectedPromise = Promise.reject(errorMessage);
  Product.find.mockReturnValue(rejectedPromise);
  await productController.getProducts(req, res, next);
  expect(next).toHaveBeenCalledWith(errorMessage);
});
```

테스트에 대응하는
실제 코드 작성

```
exports.getProducts = async (req, res, next) => {
  try {
    const allProducts = await Product.find({});
    res.status(200).json(allProducts);
  } catch (err) {
    next(err);
  }
};
```

getProducts 통합 테스트 작성

Read 부분 소스 작성

해야 할 일은 ???

Product 데이터 불러오기

통합 테스트 작성

```
test("GET /api/products", async () => {
  const response = await request(app).get('/api/products');
  expect(response.statusCode).toBe(200);
  expect(Array.isArray(response.body)).toBeTruthy();
  expect(response.body[0].name).toBeDefined();
  expect(response.body[0].description).toBeDefined();
});
```

테스트에 대응하는
실제 코드 작성

```
exports.getProducts = async (req, res, next) => {
  try {
    const allProducts = await Product.find({});
    res.status(200).json(allProducts);
  } catch (err) {
    next(err);
  }
};
```

toBeDefined → 변수가 undefined가 아닌지 체크합니다.

toBeTruhty → value 값이 무엇인지보다 value가 True인지 아닌지를 확인 합니다.

getProductById 단위 테스트 작성

Read 부분 소스 작성

해야 할 일은 ???

→ Product Id로 데이터 가져오기 위한 함수 만들기

단위 테스트 작성

```
describe("Product Controller GetById", () => {
    Run | Debug
    it("should have a getProductById", () => {
        expect(typeof productController.getProductById).toBe("function");
    });
})
```

테스트에 대응하는 실제 코드 작성

```
exports.getProductById = async (req, res, next) => {};
```

해야 할 일은 ???

→ getProductById 함수를 호출할 때 Product Model의 findById 메소드를 route parameter와 함께 호출하기

단위 테스트 작성

```
it("should call Product.findById", async () => {
  req.params.productId = productId;
  await productController.getProductById(req, res, next);
```

```
  expect(Product.findById).toBeCalledWith(productId);
});
```

테스트에 대응하는
실제 코드 작성

```
exports.getProductById = async (req, res, next) => {
  await Product.findById(req.params.productId);
};
```

해야 할 일은 ???

가져온 Product 데이터를
json 형식과
성공 Status를 Return 해주기

단위 테스트 작성

```
it("should return json body and response code 200", async () => {
  Product.findById.mockReturnValue(newProduct);
  await productController.getProductById(req, res, next);
  expect(res.statusCode).toBe(200);
  expect(res._getJSONData()).toStrictEqual(newProduct);
  expect(res._isEndCalled()).toBeTruthy();
});
```

테스트에 대응하는
실제 코드 작성

```
exports.getProductById = async (req, res, next) => {
  const product = await Product.findById(req.params.productId);
  res.status(200).json(product);
};
```

해야 할 일은 ???

Product Id에 맞는
데이터가 없을 경우
404 STATUS를 전달

```
it("should return 404 when item doesnt exist", async () => {
```

단위 테스트 작성

```
  Product.findById.mockReturnValue(null);
  await productController.getProductById(req, res, next);
  expect(res.statusCode).toBe(404);
  expect(res._isEndCalled()).toBeTruthy();
});
```

테스트에 대응하는 실제 코드 작성

```
exports.getProductById = async (req, res, next) => {
  const product = await Product.findById(req.params.productId);
  if (product) {
    res.status(200).json(product);
  } else {
    res.status(404).send();
  }
};
```

해야 할 일은 ???

에러가 났을때 에러처리

단위 테스트 작성

```
it("should handle errors", async () => {
  const errorMessage = { message: "error" };
  const rejectedPromise = Promise.reject(errorMessage);
  Product.findById.mockReturnValue(rejectedPromise);
  await productController.getProductById(req, res, next);
  expect(next).toHaveBeenCalledWith(errorMessage);
});
```

테스트에 대응하는 실제 코드 작성

```
exports.getProductById = async (req, res, next) => {
  try {
    const product = await Product.findById(req.params.productId);
    if (product) {
      res.status(200).json(product);
    } else {
      res.status(404).send();
    }
  } catch (err) {
    next(err);
  }
};
```

getProductById 통합 테스트 작성

Read 부분 소스 작성

해야 할 일은 ???

Product Id로 데이터 가져오기

통합 테스트 작성

```
test("GET /api/products/:productId", async () => {
  const response = await request(app).get(`/api/products/${firstProduct._id}`);
  expect(response.statusCode).toBe(200);
  expect(response.body.name).toBe(firstProduct.name);
  expect(response.body.description).toBe(firstProduct.description);
});
```

테스트에 대응하는
실제 코드

```
exports.getProductById = async (req, res, next) => {
  try {
    const product = await Product.findById(req.params.productId);
    if (product) {
      res.status(200).json(product);
    } else {
      res.status(404).send();
    }
  } catch (err) {
    next(err);
  }
};
```

해야 할 일은 ???

Product Id로 데이터 가져올 때
존재하지 않는 Product ID 일 경우

통합 테스트 작성

```
it("GET id doenst exist /api/products/:productId", async() => {
    const response = await request(app).get('/api/products/5f5cb1e3646c57caf47a1788');
    expect(response.statusCode).toBe(404);
})
```

테스트에 대응하는
실제 코드

```
exports.getProductById = async (req, res, next) => {
  try {
    const product = await Product.findById(req.params.productId);
    if (product) {
      res.status(200).json(product);
    } else {
      res.status(404).send();
    }
  } catch (err) {
    next(err);
  }
};
```