howest
hogeschool

Object Oriented Architectures and Secure Development

Mock repositories in the N tier model

Matthias Blomme
Mattias De Wael
Frédéric Vlummens

What is a mock repository?

- There isn’t always a need to use a MySQL database when writing tests
  - Are you testing your domain classes? Or rather your SQL connection and syntax?
  - Using a production DB is discouraged
    - You can setup a test DB for testing purposes
    - Can be lots of work
  - Alternative: mocking using a fake repository
- As long as your class implements the repository interface, you’re good to go!

- Note: in-memory repositories can also be used for purposes other than testing

InMemoryRepository

```
public class InMemoryProductsRepository implements ProductsRepository {

    private final List<Product> products = new ArrayList<>();

    @Override
    public List<Product> getProducts() {
        return Collections.unmodifiableList(products);
    }

    @Override
    public void addProduct(Product product) {
        products.add(product);
    }
}
```

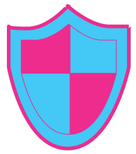
InMemoryRepository

```
public class InMemoryProductsRepository implements ProductsRepository {

    private final List<Product> products = new ArrayList<>();

    @Override
    public List<Product> getProducts() {
        return Collections.unmodifiableList(products);
    }

    @Override
    public void addProduct(Product product) {
        products.add(product);
    }
}
```



howest hogeschool

Secure coding guideline 6-12: Do not expose modifiable collections

- https://www.oracle.com/technetwork/java/seccodeguide-139067.html#6-12

```
public List<Product> getProducts() {
    return Collections.unmodifiableList(products);
}
```

- Classes that expose collections either through public variables or get methods have the potential for side effects, where calling classes can modify contents of the collection. Developers should consider exposing read-only copies of collections relating to security authentication or internal state.

Compare: GoodProgram vs MaliciousProgram

- GoodProgram:

```
public List<Product> getProducts() {
    return Collections.unmodifiableList(products);
}

ProductsRepository repo = Repositories.getProductsRepository();

Product newProduct = new Product(0, "lemonade", 5.99);
repo.addProduct(newProduct);

repo.getProducts().remove(0);
```

Exception in thread "main" java.lang.UnsupportedOperationException

howest hogeschool

Compare: GoodProgram vs MaliciousProgram

- MaliciousProgram:

```
public List<Product> getProducts() {
    return products; // DO NOT DO THIS!
}

ProductsRepository repo = Repositories.getProductsRepository();

Product newProduct = new Product(0, "lemonade", 5.99);
repo.addProduct(newProduct);

repo.getProducts().remove(0);
```

Product at position 0 is removed from the list!

Writing tests: plugging in the MockRepository instead of the MySqlRepository

- In **main**:

```
public class Repositories {

    private static final ProductsRepository REPO = new MySqlProductsRepository();

    private Repositories() {}

    public static ProductsRepository getProductsRepository() {
        return REPO;
    }

}
```

Writing tests: plugging in the MockRepository instead of the MySqlRepository

- In **test**:

```
public class Repositories {

    private static final ProductsRepository REPO = new InMemoryProductsRepository();

    private Repositories() {}

    public static ProductsRepository getProductsRepository() {
        return REPO;
    }

}
```

Writing tests: plugging in the MockRepository instead of the MySqlRepository

- During run→ Repositories.getProductsRepository() returns the MySqlProductsRepository
- During test → Repositories.getProductsRepository() returns the InMemoryProductsRepository

Writing tests: plugging in the MockRepository instead of the MySqlRepository

- In **test**:

```
public class Repositories {

    private                                                    sRepository();

    private

    public s
        retur
    }

}
```

If your classes have the same name and are stored in the same packages in both main and test, during testing the system will take your test classes instead of the main classes. This allows for easy testing and plugging in other classes in test than in production.

howest hogeschool