**현장실습 주간보고서[5주차]**

**21.02.01 ~ 21.02.05**

**아주대 미디어학부 황지훈**

**펜타큐브-개발**

**1. 엑셀 데이터 -> C# 도메인 클래스 로드**

이번 주차부터 새로운 프로젝트를 시작했다. 엑셀파일을 읽어 펜타큐브 프로그램이 읽을 수 있는 XML스키마에 맞게 데이터들을 삽입하는 프로젝트이다. 엑셀파일은 각 part(부품) 정보들이 각 sheet에 구성되어 있다. sheet에는 부품에 맞는 헤더 정보와 데이터가 작성되어 있는 형태이다.

이번주에는 엑셀 데이터를 읽어 C# class(ParsedExcelData)로 옮기는 과정까지 진행했다. 해당 C# class는 sheet단위로 객체가 생성되며 sheet의 각 row(각 part정보)는 string 타입을 담는 리스트로 구성했다.

DefaultExcelParser class는 엑셀 데이터를 읽어 parsing하여 class로 옮기는 역할을 한다. Parse() 메소드는 펜타큐브에서 제공되는 SpreadBook의 Open()을 이용하여 엑셀파일을 sheet단위로 읽고 sheet의 가장 상단에 위치하는 헤더 정보를 제외한 데이터들만 ParsedExcelData instance의 Context property에 만들어 저장했다.

VendorCode, CustomerCode 등 9개는 모든 부품들이 공통적으로 갖는 property라는 점을 이용하여 Polymorphism 패턴을 적용하여 각 부품단위로 class를 생성했다.

Capacitance, Resistance, Inductor의 경우 각 값에 해당하는 단위가 같은 셀에 존재한다. 예를 들어 Capacitance 값을 의미하는 셀에 “13mF”와 같이 값과 단위가 함께 하나의 셀에 존재한다. 따라서 해당 셀을 읽고 값과 단위를 따로 parsing해주는 함수를 구현했다. 단위가 명시되지 않은 셀에는 default 단위를 갖는 것으로 처리했다.

Pin List를 갖을 수 있는 부품은 BJT, FET, IC, Relay 총 4개이다. 따라서 4개의 부품 class에 대해서는 Pin List를 의미하는 property를 추가했다. 그리고 BJTPin, FETPin, ICPin, RelayPin class를 선언하여 각 Pin List가 해당 객체를 담을 수 있도록 구현했다.

**2. NUnit을 이용한 단위 테스트**

Unit Test는 개별적인 코드 단위가 예상대로 작동하는지 확인하는 테스트 방식이다. Unit Test의 중요한 특징은 테스트들은 서로 분리되어 있고, 테스트 코드들은 테스트되는 코드와 분리되어 있다는 것이다. 따라서 문제가 될 수 있는 부분을 단위로 테스트를 진행할 수 있다. 이는 문제를 쉽고 빠르게 찾을 수 있도록 도와준다. .NET Framework에서 Unit Test를 사용하기 위해 MS의 .NET Test sdk파일, JetBrains의 Annotation 그리고 NUnit, NUnit3 Test Adapter 패키지를 설치했다.

NUnit.Framework 라이브러리에서는 많은 attribute를 제공한다. [TestFixture] attribute는 Unit test가 포함된 class를 나타내고 [Test] attribute는 Test함수임을 나타낸다. [Test] attribute 대신에 [TestCase] attribute를 이용하여 Test함수에 입력 값을 보낼 수 있다.

NUnit Test code의 구조는 크게 2가지가 있다. AAA (Arrange / Act / Assert)구조는 고전적인 방법으로 오랫동안 사용되어 왔다. 그리고 GWT (Given / When / Then) 구조가 있다. Arrange / Act / Assert (AAA)는 테스트의 관점과 더불어 프로그래머의 관점에서 코드에 대해 생각하게 하는 경향이 있다. 이것은 애플리케이션 코드가 고려해야 할 문제이지 테스트 관점에서 고려할 문제가 아니라는 비판을 찾아볼 수 있었다. 하지만 Give / When / Then(GWT)은 요구 사항 관점에서 문제에 대해 더 많이 생각하게 하는 방식이다. 따라서 Test 프로젝트에서는 모두 GWT방식을 사용했다.

Assert 클래스에는 Assert.AreEqual(), Assert.That()등을 포함하여 NUnit에서 일반적으로 많이 사용되는 static method들이 많이 구현되어 있다. 이러한 함수들은 보통 GWT구조의 Then 파트에서 사용된다.

**3. yield 키워드**

yield 키워드를 이용하여 iterator 메소드를 구현할 수 있다. 사용자 정의 iterator 메소드를 만들고 메소드 안에 yield break과 yield return키워드를 이용하여 구현한다. Iterator 메소드의 반환 형식은 반드시 IEnumerable, IEnumerable<T>, IEnumerator, 또는 IEnumerator<T>여야 한다. yield return 문을 사용하면 각 요소를 따로따로 반환할 수 있다. 구현한 iterator 메소드에서 반환된 시퀀스는 foreach 문 또는 LINQ 쿼리 안에 위치하여 처리된다. 즉, 각각의 foreach 루프의 반복이 iterator 메소드를 호출한다. yield return 문이 반환을 하면 코드에서 현재 위치는 유지된다. 다음에 iterator 메소드가 호출되면 해당 직전에 yield return한 위치에서 실행이 다시 시작된다. yield break 문을 사용하여 iterator 메소드를 종료할 수 있다.

yield 또는 IEnumerable를 반환하는 iterator의 IEnumerator 형식은 object이다. 따라서 iterator가 IEnumerable<T> 또는 IEnumerator<T>를 반환하도록 구현하면 제네릭 형식 매개 변수로 암시적 형변환을 거쳐 yield return할 수 있다.

**4. 소감 및 차주계획**

이번 주에는 엑셀데이터를 part를 의미하는 도메인 class로 로드하는 과정을 구현했다. 다음주에는 로드한 class를 기반으로 펜타큐브 프로그램이 인식할 수 있는 XML스키마에 맞도록 XML파일로 작성하는 과정을 구현할 계획이다.