**현장실습 주간보고서[4주차]**

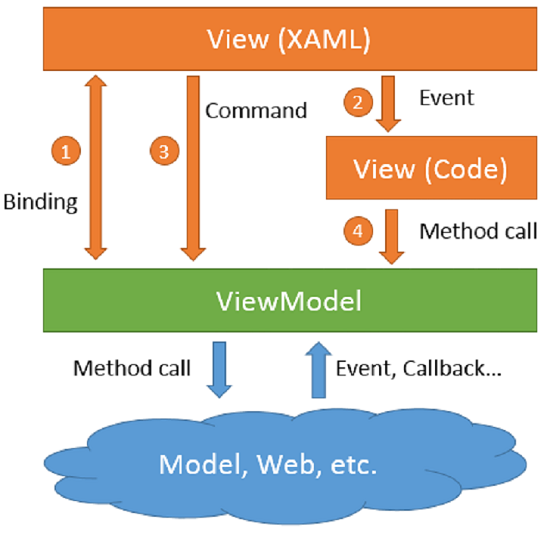
**21.1.25 ~ 21.1.29**

**아주대 미디어학부 황지훈**

**펜타큐브-개발**

**1. MVVM(Model-View-ViewModel) Design Pattern**

사용자들이 눈으로 보는 UI가 View에, 그리고 애플리케이션에서 사용되는 데이터를 처리하는 부분은 Model에 해당한다. ViewModel은 View와는 Binding이나 Command로 연결하고 Model과는 Data를 주고받는 역할을 담당한다. ViewModel은 View를 표현하기 위해 만든 View를 위한 Model이다. 이렇게 3가지 부분으로 역할을 나눠 코드를 관리하면 유지보수 및 개발효율이 좋아진다.



[[1]](#footnote-1)그림: MVVM design pattern

현재 프로젝트에서는 mapping결과를 연산하여 보여주는 Use case가 대표적인 MVVM 패턴이 적용된 부분이다. 다음은 Model, View, ViewModel이 서로 상호작용하는 시나리오이다.

1) Action(ex. 버튼클릭 등)이 View에 들어와서 Command 패턴에 의해 View Model로 Action이 전달된다.

2) View Model은 Model쪽으로 데이터를 요청하고 받는다.

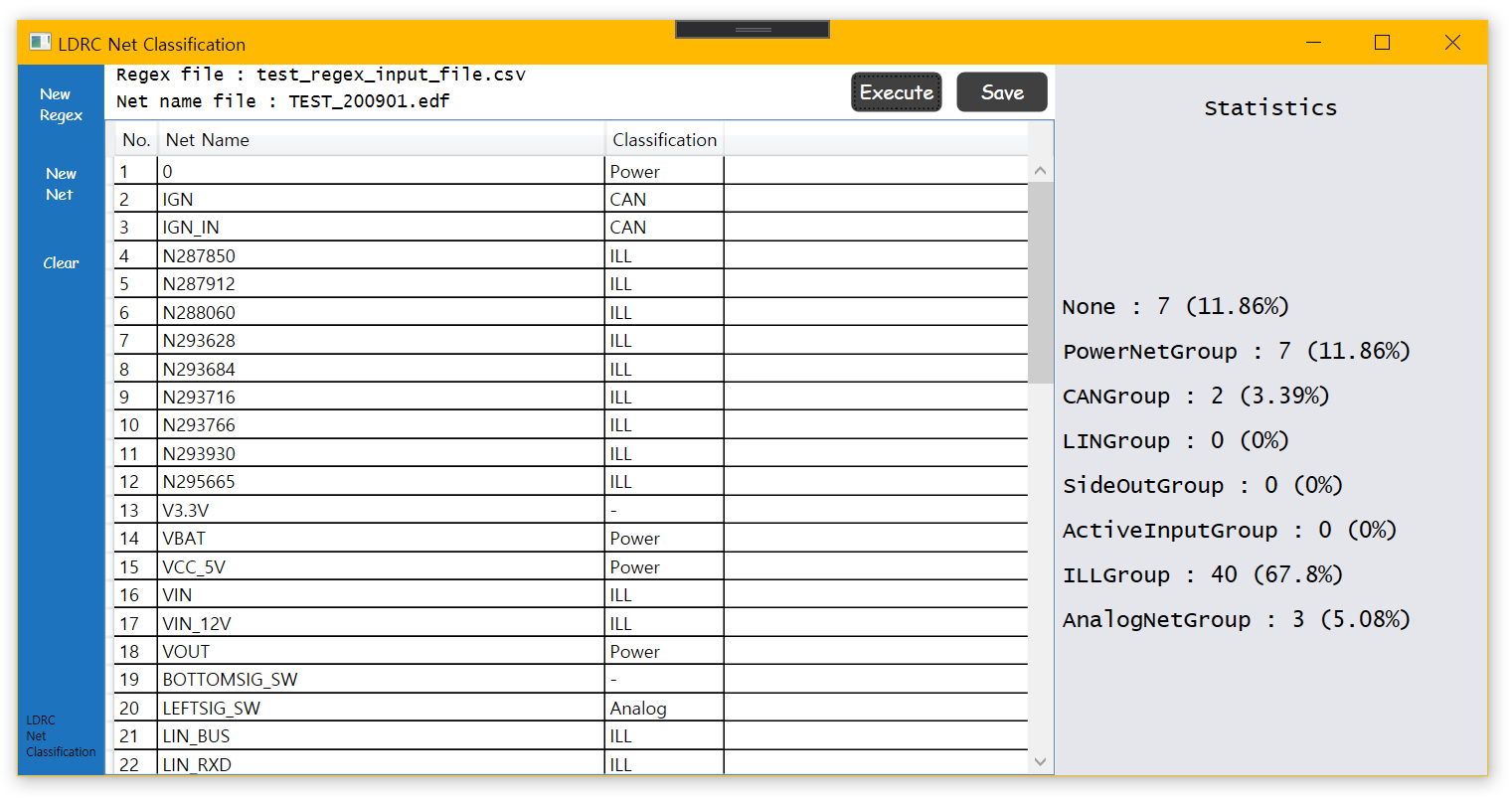
3) View Model은 View와 Binding된 property에 값을 적절히 넣기 위해 연산을 한다. (View를 위한 Data를 처리한다.)

4) Observer패턴에 의해 Binding된 property에 값이 쓰이면 View에서 데이터를 사용자에게 보인다. 이때 property의 set옵션에 SetProperty()를 사용한다.

만약 Collection은 Binding할 경우, ObservableCollection을 구현하는 collection만이 Binding 가능하다. 또한 Collection에는 primitive type보다 instance를 넣는 것이 Binding하기에 더 좋은 선택이다.

View와 ViewModel사이에 Command 패턴과 Data Binding 두 가지 패턴을 사용하여 구현했는데, 이 두가지 패턴을 이용하면 View와 View Model 사이의 dependency를 줄이면서 서로 상호작용할 수 있다는 장점이 있다. 또한 View와 ViewModel이 독립적으로 존재할 수 있기 때문에 각각 모듈화가 가능하며 reuse potential이 증가한다.

**2. UI 수정, ViewModel추가**



UI의 가장 오른쪽 부분을 수정했다. 기존에는 실행결과를 저장했던 기록을 보여주는 부분으로 사용할 계획이었으나, 각 Group에 net이 몇 개나 포함되는지 보여주도록 변경했다. 해당 부분은 1개의 TextBlock에 1개의 String data를 만들어 Binding했다.

또한 ViewModel에 해당하는 부분을 생성했다. 기존에는 MainWindow.xaml아래에 위치한 MainWindow.xaml.cs 파일(behind code)에 데이터 처리 코드를 작성했다. 하지만 MVVM Pattern을 적용하여 View를 위한 데이터를 처리하는 코드는 xaml아래가 아닌 독립적인 ViewModel파일로 옮겼다.

xaml코드에서 prism:ViewModelLocator.AutoWireViewModel="True" 옵션을 추가해주면 xaml(View part)과 ViewModel을 자동으로 연결한다. 이때 ViewModel에 해당하는 파일명은 xaml파일명 뒤에 “ViewModel”이 붙은 형태로 구성해야 ViewModel 코드와 연결될 수 있다. ViewModel 부분에서는 Command를 이용하여 특정 UI Control에서 event가 발생할 때 ViewModel에 작성된 함수가 호출되도록 했다.

**3. 매핑결과를 8개의 Group List에 저장**

Classification(분류결과)을 의미하는 8개 Group을 펜타큐브로부터 제공받은 자료구조(List)에 저장했다. 또한 8개의 List의 매핑 결과를 MappedElement instance로 구성된 MergedCollections이라는 하나의 List에 압축했다. MappedElement는 서로 다른 타입을 담는 8개의 Group List를 wrapping하기 위해 만든 class이다. MappedElement에는 3가지 property가 있다. 첫번째로 Group property는 Net이 속한 Group List에 저장된 reference이다. 예를 들어, 어떤 Net이 전원과 관련된 Net이라면 PowerNetGroup 객체가 생성되고 PowerNetGroups와 MappedElement의 Group property에 set된다. 두번째 property는 Net property이다. Net property는 매핑 결과와 상관 없이 Net자체의 온전한 정보만 갖는 Net reference로써 펜타큐브로부터 제공받은 Nets Dictionary에서 가져왔다. 마지막으로 Type property는 열거형 타입으로써 해당 Net이 어떤 Group에 속한 객체인지 명시한다. Net과 Type property가 UI의 DataGrid에 표현될 때 사용된다.

**4. 패턴 일치(Pattern Matching)**

특정 object가 어떤 타입으로 casting될 수 있는지 판단해주는 기능이다. If, switch문과 함께 사용된다. 패턴일치 이후에는 casting된 reference 타입으로 접근 가능한 메소드 들을 호출할 수 있다. Net Group들이 담는 객체 타입을 판별하기 위해 사용해봤지만 해당 부분은 수정한 상태이다.

**5. 임의의 타입으로 객체 생성**

Activator.CreateInstance()를 사용해서 런타임 중에 결정되지 않은 타입에 대해 new를 사용하지 않고 객체를 생성했다. CreateInstance()는 지정한 매개 변수와 가장 일치하는 생성자를 사용하여 지정한 유형의 인스턴스를 만든다.

도면파일로부터 읽어온 Net 들이 어떤 Group에 속할지 미리 알 수 없었기 때문에 CreateInstance()를 사용했지만, 결국 Type을 알아내기 위해 모든 Assembly를 읽어 들여야 했다. 모든 Assembly를 읽어드리는 과정 자체가 성능이 좋지 못하고 해당 코드는 이중 반복문 안에 위치해야 했기 때문에 더더욱 시간이 많이 요구된다. 따라서 Assembly를 사용하는 방법은 최적의 방법이 아니다. 결국 switch문을 사용하여 Group에 들어갈 객체 타입을 판단했고 new를 사용하여 객체를 생성했다.

**6. LINQ Pad**

간단한 C# 코드를 테스트해 볼 수 있는 프로그램이다. 한 줄 코드는 C# expression으로 2줄 이상은 C# statement로 테스트할 수 있다.

**7. C# out, ref 키워드**

out과 ref키워드를 사용하면 참조를 통해 argument를 전달할 수 있다. 이때 argument와 parameter의 prefix에 out 또는 ref 키워드를 명시해야 한다. ref 방식으로 인자를 전달할 경우 함수호출 전에 반드시 값을 초기화해야 한다. 또한 해당 함수 내에서는 reference의 값 자체가 변경될 가능성도 있고 ref가 참조하는 object의 내용이 변경될 수도 있다. C/C++에서 parameter에 &키워드를 명시하여 argument의 주소 값을 보내는 것과 유사하다. out 방식의 경우, out으로 보낸 reference에 값을 반환 받을 목적으로 사용하기 때문에, 함수호출 전에는 해당 reference를 초기화할 필요는 없고 함수 안에서는 반드시 reference를 초기화해야 한다.

**8. 소감 및 차주계획**

학부생으로 지내오면서 해왔던 코딩방식과 개발을 위한 코딩방식은 많이 차이가 있었다. 우선 작성한 코드는 남들이 이해하기 쉬운 구조로 이루어져야 하고 class, method 등의 이름도 적절하게 지어야 한다(가독성). 또한 프로젝트 규모가 커질수록 프로그램의 구조를 판단하기가 어려워지므로 여러가지 architectural design pattern을 적용할 수 있다. 그리고 같은 결과를 산출하더라도 성능적인 면을 고려해 가능한 최적의 코드를 작성해야 한다. 코드의 재사용 가능성을 늘리기 위해 다른 class들 간의 coupling을 줄이고 method는 하나의 주된 목적을 갖는 역할만 수행하도록 작성해야 한다.

다음주부터는 새로운 프로젝트를 진행할 예정이다.

1. https://docs.microsoft.com/en-us/archive/msdn-magazine/2013/march/mvvm-messenger-and-view-services-in-mvvm [↑](#footnote-ref-1)