HeimdallrComboBox

```
<ctrls:HeimdallrComboBox
    Grid.Row="3" Width="150" Height="40" Background="BlanchedAlmond"
    CornerRadius="10" BorderThickness="3" FontSize="20" Foreground="Black"
    PathIcon="Account" PathIconFill="{StaticResource DarkViolet}" IconHeight="30"
    IconWidth="30" CheckedBackground="#FFBA55D3"
    MouseOverBackground="{StaticResource DarkCharcoal}"
    DropDownBackground="#DCD7C9" ItemsSource="{Binding ProductUnitOptions}"
    DisplayMemberPath="Value" SelectedValuePath="Key"
    SelectedValue="{Binding SelectedUnit, Mode=TwoWay}"/>
```

사용방법 ->

1 CornerRadius 사용자가 Border 의 CornerRadius 설정

2 PathIcon 아이콘 지정 (아이콘만 지정 가능)

3 PathIconFill 아이콘의 색상지정

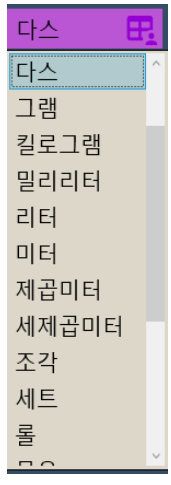
4 IconHeight 아이콘의 높이 지정

5 IconWidth 아이콘의 너비 지저

6 CheckedBackground 체크시 배경색상 지정

7 MouseOverBackground 마우스오버시 배경색상 지정

8 DropDownBackground 드롭다운시 배경색상 지정



9 ItemsSource="{Binding ProductUnitOptions}"

```
public ObservableCollection<KeyValuePair<ProductUnit, string>> ProductUnitOptions { get; }
```

ItemsSource 는 콤보박스에 표시할 항목들의 전체 컬렉션을 지정하는 속성입니다.

여기서는 ProductUnitOptions 라는 컬렉션(보통은 ObservableCollection<T>같은 형태)을 바인딩해서 콤보박스에 여러 항목을 띄우고 있습니다.

ProductUnitOptions 컬렉션 내부의 각 항목은 일반적으로
Key-Value 쌍(예: KeyValuePair<TKey, TValue>, 혹은 Dictionary<TKey, TValue>의 항목과 같은 구조)이라고 가정합니다.

ItemsSource 에 있는 각각의 아이템은 { Key, Value } 구조

DisplayMemberPath="Value"

콤보박스는 내부적으로 ItemsSource 컬렉션에 있는 각각의 개체(item)를 표시할 때, 보통 개체 자체를 ToString()으로 표시하거나, 직접 템플릿을 정의해야 합니다.

DisplayMemberPath는 "각 아이템의 어떤 프로퍼티 값을 화면에 텍스트로 보여줄지"를 지정합니다.

여기서 "Value"로 지정했으니, ProductUnitOptions 내 각 아이템의 Value 프로퍼티에 해당하는 값을 콤보박스 목록에서 보여준다는 뜻입니다.

예) 만약 아이템이 { Key = "kg", Value = "킬로그램" }이라면 콤보박스 목록에는 "킬로그램"이 표시됨.

"Value" 값은 특별한 키워드가 아니라 단지 클래스의 속성(Property) 이름입니다.

즉, DisplayMemberPath="DisplayName"이든 "Value"든, 이것은 전적으로 ItemsSource 에 바인딩된 개체의 속성 이름에 따라 달라집니다.

핵심 요약

이름	특별한 예약어인가요?	어디서 왔나요?
DisplayName	일반 속성 이름입니다.	여러분이 만든 클래스의 속성일 수 있음
Value	일반 속성 이름입니다.	예: KeyValuePair<TKey, TValue> 구조에서 나옴
Name, Title, Label 등	다 똑같이 일반 속성입니다.	의미만 다르고 역할은 동일합니다.

SelectedValuePath="Key"

콤보박스에서 아이템을 선택하면, 보통 선택된 항목 전체 개체가 SelectedItem 에 바인딩됩니다.

하지만 SelectedValuePath 를 지정하면, 선택된 아이템의 특정 프로퍼티 값만 SelectedValue 속성에 할당할 수 있습니다.

여기서는 "Key"를 지정했으므로, 선택된 아이템의 Key 프로퍼티가 SelectedValue에 바인딩됩니다.

예) 사용자가 "킬로그램" 항목을 선택하면, 내부적으로는 Key인 "kg"가 SelectedValue에 저장됨.

※ Key 외에 사용가능한 값의 조건

SelectedValuePath 에 지정할 수 있는 값은 ItemsSource 의 각 항목 개체의 공개(public) 속성 이름이어야 합니다

해당 속성은 문자열, 정수, 개체 등 어떤 타입이라도 가능합니다

```
public class ProductUnit                                    class
{
    public string Code { get; set; }            // 예: "kg"
    public string DisplayName { get; set; }     // 예: "킬로그램"
    public string Description { get; set; }     // 예: "중량 단위"
}
```

```
// ViewModel.cs
public ObservableCollection<ProductUnit> ProductUnitOptions { get; set; }
public string SelectedUnitCode { get; set; }

<ctrls:HeimdallrComboBox
    ItemsSource="{Binding ProductUnitOptions}"
    DisplayMemberPath="DisplayName"
    SelectedValuePath="Code"
    SelectedValue="{Binding SelectedUnitCode, Mode=TwoWay}" />
```

※ 익명형식은 불가

```
ItemsSource = new[] {
    new { Id = 1, Name = "사과" },
    new { Id = 2, Name = "바나나" }
};
```

익명 타입은 XAML에서 속성 경로(SelectedValuePath="Id" 등) 접근이 불가능합니다

이 경우에는 명시적인 클래스를 사용하는 것이 좋습니다

SelectedValue="{Binding SelectedUnit, Mode=TwoWay}"

SelectedValue 는 콤보박스에서 사용자가 선택한 항목의 SelectedValuePath 프로퍼티 값을 나타내는 속성입니다.

여기서는 SelectedValuePath="Key" 이므로, 실제로는 사용자가 선택한 항목의 Key 값이 됩니다.

이 값을 뷰모델(ViewModel)의 SelectedUnit 속성과 바인딩하고 있습니다.

Mode=TwoWay 이므로, 사용자 선택 시 뷰모델의 SelectedUnit 속성도 자동으로 업데이트되고, 뷰모델에서 SelectedUnit 값을 변경하면 콤보박스의 선택 항목도 자동으로 변경됩니다.

여기서 콤보박스 목록에는 Value (예: "킬로그램")를 보여주고

사용자가 선택한 항목의 Key (예: "kg")가 SelectedValue 를 통해 뷰모델의 SelectedUnit 에 바인딩(양방향)된다.

예 -> 다스는 열거형으로 아래와 같은 코드입니다.

```
[UnitInfo("다스", 12)]  // 12개 묶음
[Description("제품 단위를 묶음으로 나타냅니다. 일반적으로 여러 개의 제품이 함께 묶여 판매되는 경우에 사용됩니다.")]
Dozen,
```

DisplayMemberPath나 SelectedValuePath에 쓰는 값은 그 속성 이름 그대로를 정확기재

KeyValuePair vs Dictionary

항목	KeyValuePair<TKey, TValue>	Dictionary<TKey, TValue>
정체	단일 키-값 쌍	키-값 쌍의 컬렉션 (맵)
기능	데이터 구조 없음 (단일 데이터)	탐색, 추가, 삭제 기능 있음
주요 목적	키-값을 하나의 개체로 표현	키로 값을 빠르게 조회/저장
예제 타입	KeyValuePair<string, int>	Dictionary<string, int>
XAML 바인딩	ComboBox 의 ItemsSource 로 적합	직접 바인딩 어려움, 변환 필요
LINQ 사용	dictionary.ToList() 하면 반환되는 단위	

1 KeyValuePair<TKey, TValue>란?

단순히 하나의 키와 값의 짝을 표현하는 구조체입니다.

예를 들면:

```
var pair = new KeyValuePair<string, int>("나이", 30);
Console.WriteLine(pair.Key);   // "나이"
Console.WriteLine(pair.Value); // 30
```

주로 LINQ나 Dictionary 내부에서 foreach 순회 시 사용됩니다:

```
foreach (KeyValuePair<string, string> kvp in myDictionary)
{
    Console.WriteLine($"Key: {kvp.Key}, Value: {kvp.Value}");
}
```

2 Dictionary<TKey, TValue>란?

KeyValuePair들을 저장한 **해시 기반의 컬렉션(맵 구조)**입니다.

키를 통해 값을 매우 빠르게 조회할 수 있는 것이 장점입니다.

```
var dict = new Dictionary<string, string>();
dict["KR"] = "대한민국";
dict["JP"] = "일본";

Console.WriteLine(dict["KR"]); // "대한민국"
```

Dictionary는 내부적으로 KeyValuePair<TKey, TValue>들의 집합입니다.

빠른 조회, 중복 방지 필요시(조회 성능 최우선일때 적합)
알고리즘/검색/데이터 매핑 등에서는 Dictionary가 더 효율적

UI와 사람이 읽는 코드에서는 List<T> (직접 만든 클래스)가 더 가독성이 좋고
유지보수가 쉽습니다.

HeimdallrIconRadioButton

```
<ctrls:HeimdallrIconRadioButton Grid.Row="7" Content="Setting"
                                PathIcon="Setting" Background="Transparent"
                                CheckedForeground="Yellow" MouseOverForeground="Blue"
                                GroupName="MenuGroup" Width="120" Height="40"
                                IconFill="AliceBlue" />
```

사용방법 ->

1 Content="Setting" 문자열 (문자열만 지정 가능)
2 PathIcon="Setting" 아이콘 지정 (아이콘만 지정 가능)
3 CheckedForeground="Yellow" 체크시 아이콘 및 문자열 색상
4 MouseOverForeground="Blue" 마우스오버시 아이콘 및 문자열 색상
5 GroupName

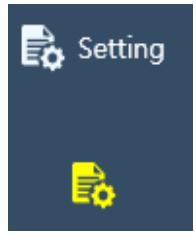
WPF에서 RadioButton 계열의 컨트롤들(예: RadioButton, 또는 ToggleButton 을 기반으로 한 커스텀 컨트롤들)에서 상호 배타적 선택 그룹을 만들기 위한 속성입니다

이 속성을 사용하면 동일한 그룹 이름을 가진 RadioButton 끼리 한 번에 하나만 선택될 수 있도록 연결됩니다.

※ 기본개념

- RadioButton은 한 번에 하나만 선택될 수 있는 옵션 집합을 제공하는 컨트롤입니다.
- XAML에서 RadioButton 여러 개를 사용하면, 기본적으로 같은 컨테이너 내에 있는 RadioButton들은 자동으로 상호 배타됩니다.
- 하지만 컨테이너가 다르거나, 명시적으로 그룹을 나누고 싶을 때 GroupName을 사용합니다.

```
<StackPanel>
  <RadioButton Content="Option A" GroupName="MenuGroup" />
  <RadioButton Content="Option B" GroupName="MenuGroup" />
  <RadioButton Content="Option C" GroupName="MenuGroup" />
</StackPanel>
```



위 세 개의 RadioButton 은 GroupName="MenuGroup"으로 지정되어 있어 한 번에 하나만 선택 가능합니다.

A를 선택하면 B와 C는 해제됨. 그룹이름이 없으면 간섭하지 않음

※ RadioButton, ToggleButton 만 GroupName 을 가질수 있음

HeimdallrNumericUpDown

```
<ctrls:HeimdallrNumericUpDown
    Grid.Row="9" Min="0" Max="1000" Step="10" Height="30" Width="100"
    Foreground="Black" FontSize="16"
    UpIconFill="#FF3DC2EC" DownIconFill="#FFD95F59"
    DownButtonIcon="ChevronDownEllipse" UpButtonIcon="ChevronUpEllipse"
    Value="{Binding SomeValue, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" />
```

사용방법 ->

1	Min	최소값 지정
2	Max	최대값 지정
3	Step	증감단위
4	UpIonFill	오른쪽 아이콘 색상 지정
5	DownIconFill	왼쪽 아이콘 색상지정
6	DownButtonIcon	왼쪽아이콘 지정
7	UpButtonIcon	오른족 아이콘 지정
8	Value	중앙 값 표기

※ Step 의 바인딩 + MVVM 자동 변환 방법

```
private ProductUnit _selectedUnit;
// ProductUnit 열거형 값입니다.
public ProductUnit SelectedUnit
{
    get => _selectedUnit;
    set
    {
        if (_selectedUnit != value)
        {
            _selectedUnit = value;
            OnPropertyChanged();
            // 단위 변경 시 Step도 자동 조정
            UpdateStep();
        }
    }
}
```

```
private double _step = 1;
public double Step
{
    get => _step;
    set
    {
        _step = value;
        OnPropertyChanged();
    }
}  바인딩 값
```

```
private void UpdateStep()
{
    switch (SelectedUnit)
    {
        case ProductUnit.Dozen:
            Step = 12;
            break;
        case ProductUnit.Kilogram:
            Step = 0.1;
            break;
        case ProductUnit.Gram:
            Step = 1;
            break;
        case ProductUnit.Each:
        case ProductUnit.Piece:
            Step = 1;
            break;
        default:
            Step = 1;
            break;
    }
}
            Step = 1;
            break;
        case ProductUnit.Each:
        case ProductUnit.Piece:
            Step = 1;
            break;
        default:
            Step = 1;
            break;
    }
}
```

xaml 에서 Step 속성을 바인딩

```
<ctrls:HeimdallrNumericUpDown
    Grid.Row="9"
    Min="0" Max="1000"
```

```
    Step="{Binding Step}" <!-- 여기 -->
    Height="30" Width="100"
    Foreground="Black" FontSize="16"
    UpIconFill="#FF3DC2EC" DownIconFill="#FFD95F59"
    DownButtonIcon="ChevronDownEllipse" UpButtonIcon="ChevronUpEllipse"
    Value="{Binding SomeValue, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" />
```

※ 단위 선택 ComboBox 예시 (선택 변경시 Step도 갱신됨)

```
<ComboBox ItemsSource="{Binding AvailableUnits}"
          SelectedItem="{Binding SelectedUnit}"
          DisplayMemberPath="Description" />
```

AvailableUnits 는 enum 목록이고, SelectedUnit 이 바뀔 때마다 Step 값도 UpdateStep()을 통해 자동 조정됩니다.

```
public IEnumerable<ProductUnit> AvailableUnits =>
    Enum.GetValues(typeof(ProductUnit)).Cast<ProductUnit>();
```

※ HeimdallrComboBox 와 HeimdallrNumericUpDown 의 협업(콜라보레이션)을 적용하여 증감숫자 변경방법