**Eventy, EventHandler**

**Trida ktera chce event vyhodit ho musi definovat :**

public event EventHandler OnFilterCanExecuteChanged;

**Na vyhazovani eventu si vytvorime metodu:**

private void FireOnFilterCanExecuteChanged()

{

OnFilterCanExecuteChanged?.Invoke(this, null);

}

**Trida ktera chce event poslouchat (pravdepodobne jiny ViewModel) se musi zaregistrovat jako posluchac:**

MotivletFilters.OnFilterCanExecuteChanged += OnFilterCanExecuteChanged;

**Trida ktera chce event poslouchat musi mit metodu ktera se provede kdyz sem dorazi event:**

private void OnFilterCanExecuteChanged(object sender, EventArgs e)

{

NewValuesFilters.ButtonGroupCanExecute = MotivletFilters.Filters.Any(d => d.SelectedValues.Any()) && NewValuesFilters.Filters.Any(d => d.SelectedValues.Any()) && ChangingInProgress == false;

}

**Pri likvidaci teto tridy se musi posluchac odregistrovat :**

#region IDispose Implementation

private bool m\_disposed;

public virtual void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (!m\_disposed)

{

if (disposing)

{

}

MotivletFilters.OnFilterCanExecuteChanged -= OnFilterCanExecuteChanged;

}

m\_disposed = true;

}

#endregion

EventHandler <genericky>

**Slouzi na predani objektu jako EventArgs**

**Chci predat napr boolean**

**Trida ktera vyhazuje ma genericky event**

public event EventHandler<bool> ChangingInProgressChanged;

**a vyhazovaci metodu s parametrem:**

private void FireChangingInProgressChanged(bool parameter)

{

ChangingInProgressChanged?.Invoke(this, parameter);

}

**Trida ktera chce byt posluchacem se registruje**

DataViewModel.ChangingInProgressChanged += ChangingInProgress;

**A ma metodu s parametem ktera se provede kdyz sem event dorazi:**

private void ChangingInProgress(object sender, bool e)

{

Filters.ChangingInProgress = e;

}

**Odregistrace pri likvidaci tridy:**

#region IDispose Implementation

private bool m\_disposed;

public virtual void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (!m\_disposed)

{

if (disposing)

{

}

DataViewModel.ChangingInProgressChanged -= ChangingInProgress;

}

m\_disposed = true;

}

#endregion