Basic

DynamicRow<User> row = new DynamicRow<User>();

row.DefineProperty("User", user);

row.DefineProperty("Login", user.Login);

row.DefineProperty("Name", user.Name);

foreach (var f in fields)

row.DefineProperty("Field\_" + f.FieldName, f.FieldValue);

Change detection on row :

row.PropertyChanged += (entity, args) => DynamicRowPropertyChange(someParameters, entity, args);

…

private static void DynamicRowPropertyChange(someParameters , PropertyChangedEventArgs args, DynamicRow< User > dynRow)

{

if (args.PropertyName.StartsWith("Field\_"))

{

…

}

}

Change detection on property (preferable to detection on row):

DynamicRow<User> row = new DynamicRow<User>();

row.DefineProperty("User", myObject);

// method 1 : second parameter is a « Value »

row.DefineProperty("Login", myObject.Login);

// method 2 : second parameter is a PropertyDescriptor with Getter, Setter (optional), OnChange (optional)

row.DefineProperty("Field\_Id", new PropertyDescriptor {

Getter = () => myObject. Id

// No setter : read only property

});

// editable property

row.DefineProperty("Field\_Name", new PropertyDescriptor {

Getter = () => myObject.Name,

Setter = (value) => { myObject.Name = value as string; return true;},

OnChange = (oldValue, newValue) =>{NotifyChange(…) ;}

});