

<https://github.com/Kobdik/DynaRepo>

DYNALIB

Библиотека для ускорения доступа к
данным на основе ADO.NET

Кобди́ков
Туле́ген

Промежуточное ПО. Брокер данных.



Стандартное использование EF в WEB API приложении:

1. Классы сущностей модели
2. Контроллеры и методы действий
3. Отображение методов HTTP на действия
4. Привязка моделей
5. Исполнение SQL операторов
6. ООП и функциональный стиль
7. Сериализация данных в поток
8. LINQ to Entities
9. Снижение скорости получения данных

<https://github.com/Kobdik/DynaRepo>

```
using Kobdik.Common;
using Kobdik.DataModule;

static DataMod dataMod = DataMod.Current();

IDynaObject dynaObject = dataMod.GetDynaObject("Invoice");

using (FileStream rfs = new FileStream("Invoice_Params.json", FileMode.Open))
{
    //считываем параметры из входного потока
    dynaObject.ReadPropStream(rfs, "sel");
}
```

```
//select-запрос с имитацией выгрузки в json-поток  
using (FileStream fs = new FileStream("Invoice.json", FileMode.Create))  
{  
    dynaObject.SelectToStream(fs);  
}
```

Данные выборки куда-то утекли,
даже не рефлексирюя :)



Dictionary First

T_QryDict таблица описания запросов

Byte	Qry_Id	код запроса
String	Qry_Name	имя запроса
String	Qry_Head	заголовок

T_PamDict параметры select -запросов

Byte	Qry_Id	код запроса
Byte	Pam_Id	код параметра
String	Pam_Name	имя параметра
String	Pam_Head	заголовок
Byte	Pam_Type	DbType enum
Short	Pam_Size	размер поля
String	Note	пояснения

T_ColDict таблица колонок запросов

Byte	Qry_Id	код запроса
Byte	Col_Id	код колонки
String	Col_Name	имя колонки
String	Col_Head	заголовок
Byte	Col_Type	DbType enum
Short	Col_Size	размер поля
String	Note	пояснения
Byte	Col_Flag	битовая маска

T_QryDict - таблица описания запросов:

Qry_Id	Qry_Name	Qry_Head	Col_Def
27	Invoice	Счета	27
28	InvoCut	Счета(усеченный запрос)	28

T_PamDict - таблица параметров select-запросов:

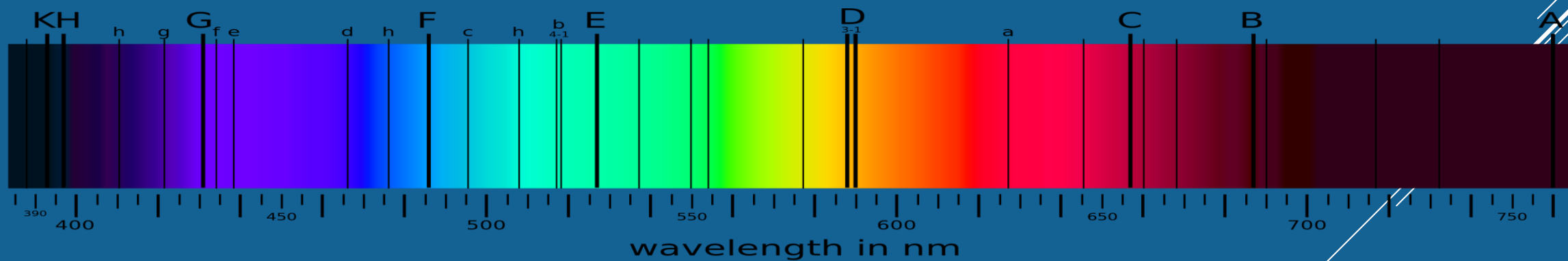
Qry_Id	Pam_Id	Pam_Name	Pam_Head	Pam_Type	Pam_Size
27	1	Dt_Fst	Начальная дата	40	3
27	2	Dt_Lst	Конечная дата	40	3

T_ColDict - таблица описания колонок полей запросов:

Qry_Id	Col_Id	Col_Name	Col_Head	Col_Type	Col_Size	Note	Col_Flag
27	0	Idn	Код начисления	56	4	idn,out	9
27	1	Org	Контрагент	52	2	sel,det	6
27
27	11	Usr	Пользователь	167	15	usr	32
27	12	Pnt	Получатель	48	1	sel,det	6
28	0	Idn	Код начисления	56	4	idn,out	9
28	1	Dt_Invo	Дата начисления	40	3	sel,det	6
28	2	Val	Начислено	62	8	sel,det,out	14
28	3	Note	Примечание	167	100	sel,det,out	14

Набор битовых флагов

idn - 1, sel - 2, det - 4, out - 8, opt - 16, usr - 32.



```
CREATE TABLE dbo.T_Invoice(  
    Idn int IDENTITY(1,1) NOT NULL,  
    Org smallint NOT NULL,  
    Knd tinyint NOT NULL,  
    Dt_Invo date NOT NULL,  
    Val float NOT NULL,  
    Note varchar(100) NOT NULL,  
    Sdoc tinyint NOT NULL,  
    Dt_Sdoc date NOT NULL,  
    Lic int NOT NULL,  
    Usr varchar(15) NOT NULL,  
    Pnt tinyint NOT NULL,  
    CONSTRAINT [PK_T_Invoice]  
    PRIMARY KEY CLUSTERED ( [Idn] ASC ))  
ON [PRIMARY]
```

idn | sel

```
CREATE PROC dbo.sel_Invoice  
@Dt_Fst date, @Dt_Lst date  
AS  
SELECT Idn, Org, Knd, Dt_Invo, Val, Note, Lic, Pnt  
FROM dbo.T_Invoice  
--WHERE Dt_Fst=@Dt_Fst, Dt_Lst=@Dt_Lst  
RETURN 0;
```

idn | sel | det | usr

```
CREATE PROC dbo.det_Invoice @Idn int
AS
SELECT Idn, Org, Knd, Dt_Invo, Val, Note, Lic, Pnt
FROM dbo.T_Invoice
WHERE Idn=@Idn
RETURN 0;
```

idn | det | out | usr

```
CREATE PROC dbo.ins_Invoice
```

```
@Idn int out, @Org smallint, @Knd tinyint, @Dt_Invo date, @Val float, @Note  
varchar(100), @Sdoc tinyint, @Dt_Sdoc date, @Lic int, @Ushr varchar(15), @Pnt  
tinyint
```

```
AS
```

```
INSERT INTO dbo.T_Invoice
```

```
(Org, Knd, Dt_Invo, Val, Note, Sdoc, Dt_Sdoc, Lic, Ushr, Pnt)
```

```
VALUES (@Org, @Knd, @Dt_Invo, @Val, @Note, @Sdoc, @Dt_Sdoc, @Lic, @Ushr, @Pnt)
```

```
SET @Idn=CAST(IDENT_CURRENT('dbo.T_Invoice') AS int)
```

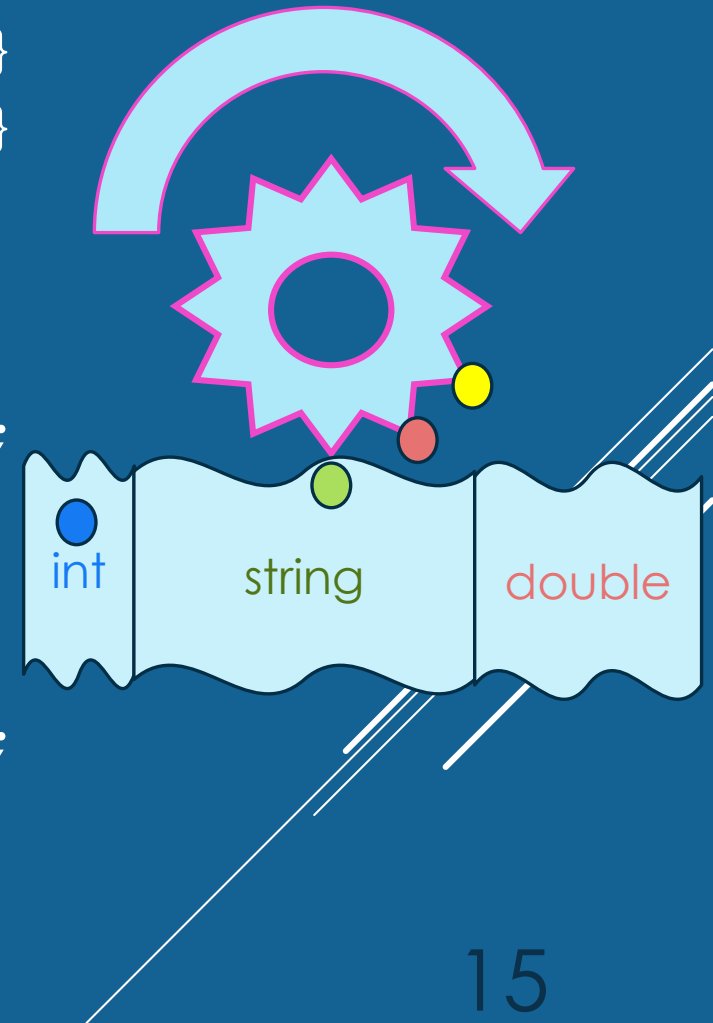
```
RETURN 0;
```

idn | det | out | usr

```
CREATE PROC dbo.upd_Invoice
@Idn int out, @Org smallint, @Knd tinyint, @Dt_Invo date, @Val float, @Note
varchar(100), @Sdoc tinyint, @Dt_Sdoc date, @Lic int, @Ushr varchar(15), @Pnt
tinyint
AS
UPDATE dbo.T_Invoice SET
Org=@Org, Knd=@Knd, Dt_Invo=@Dt_Invo, Val=@Val, Note=@Note, Sdoc=@Sdoc,
Dt_Sdoc=@Dt_Sdoc, Lic=@Lic, Ushr=@Ushr, Pnt=@Pnt
WHERE Idn=@Idn
RETURN 0;
```

DynaObject – основной компонент библиотеки.

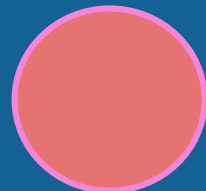
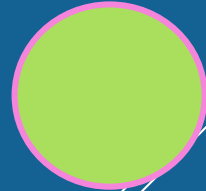
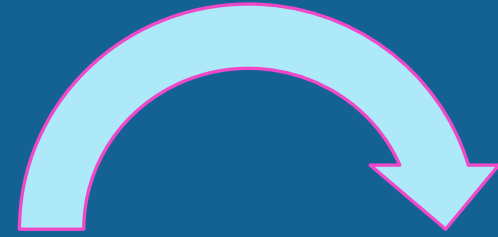
```
public interface IDynaObject : IDisposable
{
    Dictionary<String, IDynaProp> ParmDict { get; }
    Dictionary<String, IDynaProp> PropDict { get; }
    //адаптеры чтения записи в потоки
    IStreamReader StreamReader { get; }
    IStreamWriter StreamWriter { get; }
    void ReadPropStream(Stream stream, string cmd);
    //исполняют запрос и пишут в поток
    void SelectToStream(Stream stream);
    void DetailToStream(Stream stream, int idn);
    void ActionToStream(Stream stream, string cmd);
    //вспомогательный метод
    string GetInfo(string kind);
}
```



```
public interface IDynaProp
{
    string GetName();
    DbType GetDbType();
    Type GetPropType();
    int GetSize();
    byte GetFlags();
    Object Value { get; set; }
    int Ordinal { get; set; }
    void WriteProp(IDataRecord record, IPropWriter writer);
}
```

```
public override void WriteProp(IDataRecord record, IPropWriter writer)
{
    writer.WriteProp(GetName(), record.GetString(Ordinal));
}
```

```
public override void WriteProp(IDataRecord record, IPropWriter writer)
{
    writer.WriteProp(GetName(), reader.GetDouble(Ordinal));
}
```



Операции записи простых свойств в поток.

```
public interface IPropWriter
{
    void WriteProp(String propName, String value);
    void WriteProp(String propName, Byte value);
    void WriteProp(String propName, Int16 value);
    void WriteProp(String propName, Int32 value);
    void WriteProp(String propName, DateTime value);
    void WriteProp(String propName, Double value);
}
```

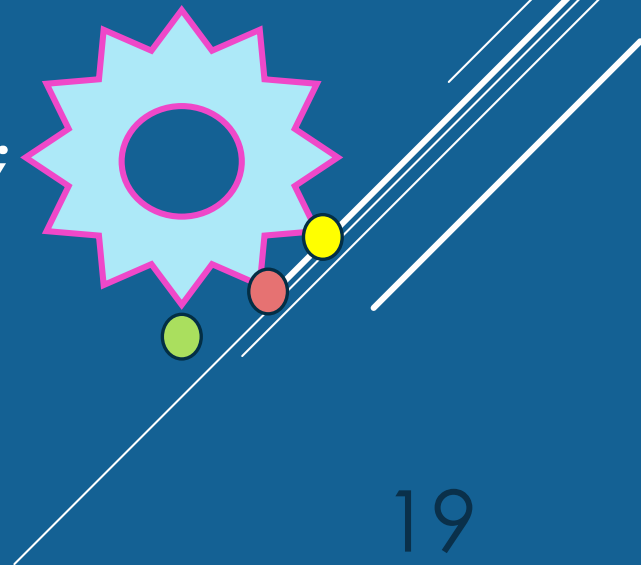
Sealed классы, реализующие IDynaProp – StringProp, ByteProp, Int16Prop, Int32Prop, DateProp, DoubleProp.

```
public interface IStreamWriter : IPropWriter
{
    byte GetStreamType();
    void Open(Stream stream);
    void PushArr();
    void PushObj();
    void PushArrProp(string propName);
    void PushObjProp(string propName);
    void Pop();
    void Close();
    string Result { get; }
}
```

Реализации: BinStreamWriter, JsonStreamWriter, XmlStreamWriter.

```
{"selected": [{row}, {row}, ..., {row}],
  "message": "Ok", "sel_time": "11:19", "time_ms": 15}
```

```
StreamWriter.Open(stream);  
StreamWriter.PushObj();  
StreamWriter.PushArrProp("selected");  
DateTime fst = DateTime.Now;  
selReader = Select();  
if (selReader != null)  
{  
    while (selReader.Read())  
    {  
        StreamWriter.PushObj();  
        WriteRecord(selReader, ReadList, StreamWriter);  
        StreamWriter.Pop();  
    }  
    selReader.Close();  
};  
StreamWriter.Pop();
```



```
private void WriteRecord(IDataRecord record,  
List<IDynaProp> props, IPropWriter writer)  
{  
    foreach (var prop in props)  
        prop.WriteProp(record, writer);  
}
```



```
DateTime lst = DateTime.Now;  
TimeSpan ts = lst - fst;  
StreamWriter.Pop();  
//В контексте объекта-контейнера  
StreamWriter.WriteProp("message", Query.Result);  
StreamWriter.WriteProp("sel_time", lst.ToShortTimeString());  
StreamWriter.WriteProp("time_ms", ts.Milliseconds);  
StreamWriter.Pop();  
StreamWriter.Close();
```

На выходе получаем json-файл следующего вида, где **{row}** - сокращение для записи строк:

```
{"selected": [{row}, {row}, ..., {row}], "message": "Ok", "sel_time": "11:19", "time_ms": 15}
```

```
config.Routes.MapHttpRoute(  
    name: "DynaSelect",  
    routeTemplate: "api/Dyna/sel/{qry}",  
    defaults: new { controller = "Dyna", action = "SelectJson" }  
);  
  
[HttpPost]  
[Authorize]  
public IHttpActionResult SelectJson(string qry)  
{  
    if (!dataMod.CheckAccess(qry, 0, User.Identity.Name))  
        return BadRequest("У нет прав на просмотр данных!");  
    IDynaObject dynaObject = dataMod.GetDynaObject(qry);  
    if (dynaObject == null) return BadRequest(dataMod.lastError);  
    Task<Stream> readTask = Request.Content.ReadAsStreamAsync();  
    return new SelectJsonStreamResult(dynaObject, readTask);  
}
```

```
config.Routes.MapHttpRoute(  
    name: "DynaDetail",  
    routeTemplate: "api/Dyna/get/{qry}/{idn}",  
    defaults: new { controller = "Dyna", action = "DetailJson" }  
);  
  
[HttpGet]  
[Authorize]  
public IHttpActionResult DetailJson(string qry, int idn)  
{  
    if (!dataMod.CheckAccess(qry, 1, User.Identity.Name))  
        return BadRequest("У Вас нет прав на просмотр детализации!");  
  
    IDynaObject dynaObject = dataMod.GetDynaObject(qry, true);  
    if (dynaObject == null) return BadRequest(dataMod.lastError);  
  
    return new DetailJsonStreamResult(dynaObject, idn);  
}
```

```
config.Routes.MapHttpRoute(  
    name: "DynaAction",  
    routeTemplate: "api/Dyna/{cmd}/{qry}",  
    defaults: new { controller = "Dyna", action = "ActionJson" }  
);  
  
[HttpPost]  
[Authorize]  
public IHttpActionResult ActionJson(string cmd, string qry)  
{  
    if (!dataMod.CheckAccess(qry, 2, User.Identity.Name))  
        return BadRequest("У Вас нет прав на ИЗМЕНЕНИЕ данных!");  
    IDynaObject dynaObject = dataMod.GetDynaObject(qry, true);  
    if (dynaObject == null) return BadRequest(dataMod.lastError);  
  
    Task<Stream> readTask = Request.Content.ReadAsStreamAsync();  
    return new ActionJsonStreamResult(dynaObject, readTask, cmd);  
}
```


Должно быть утомительно писать
заглушки на десятки однообразных
методов действий контроллеров?



- ✓ Без классов сущностей модели
- ✓ Контроллеры и методы действий
- ✓ Отображение методов HTTP на действия
- ✓ Без привязка моделей
- ✓ Исполнение SQL процедур
- ✓ Разделение ООП и функционального стиля
- ✓ Потокосное чтение и запись данных

DynaQuery для работы с LINQ to Objects

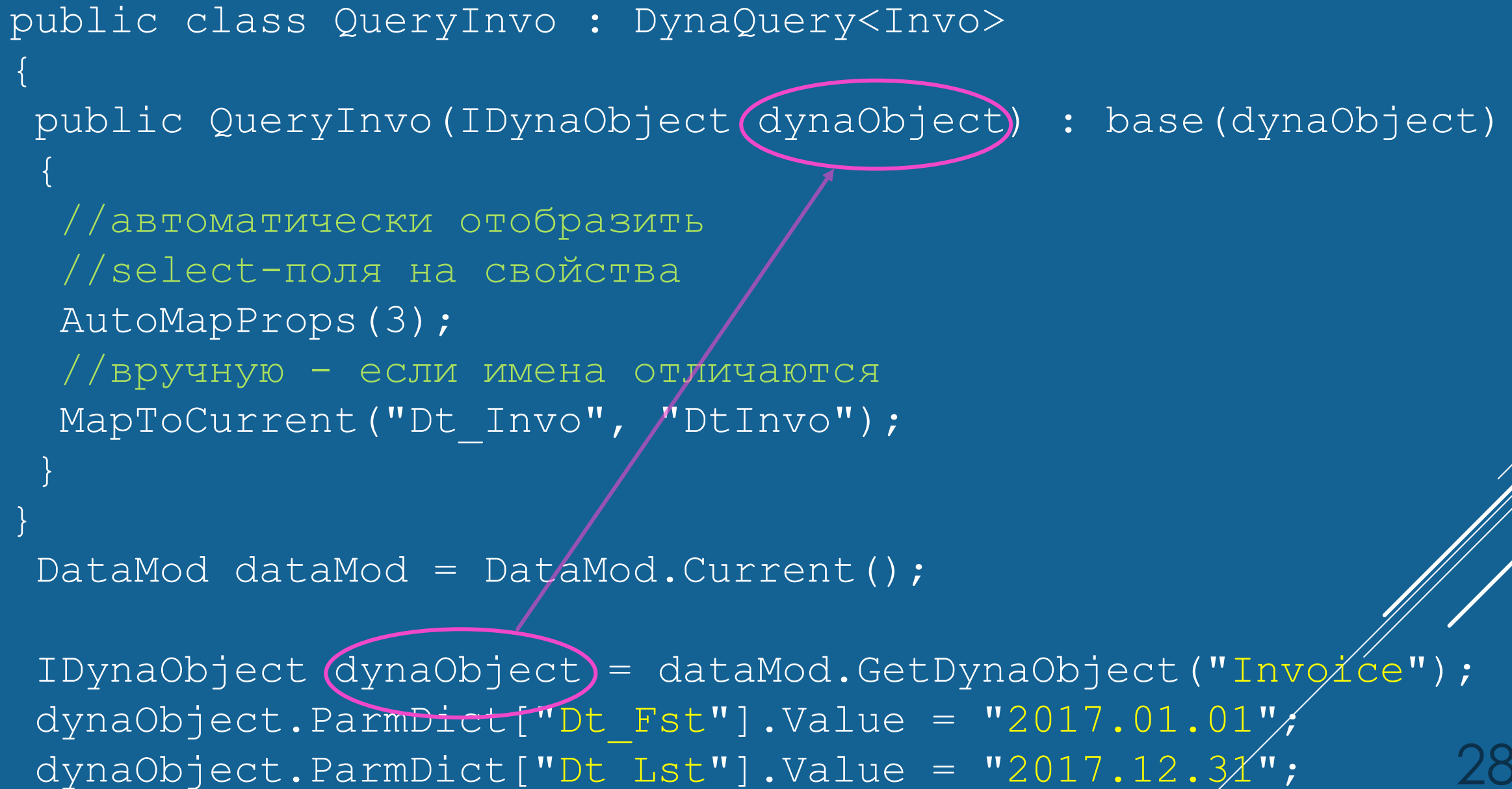
```
DynaQuery<T> : IEnumerable<T>, IEnumerator<T>.  
T Current;  
bool MoveNext();  
void Reset();
```

```
public class Invo  
{  
    public Invo() { }  
    public Int32 Idn { get; set; }  
    public DateTime DtInvo { get; set; }  
    public double Val { get; set; }  
    public string Note { get; set; }  
}
```

```
public class QueryInvo : DynaQuery<Invo>
{
    public QueryInvo(IDynaObject dynaObject) : base(dynaObject)
    {
        //автоматически отобразить
        //select-поля на свойства
        AutoMapperProps(3);
        //вручную - если имена отличаются
        MapToCurrent("Dt_Invo", "DtInvo");
    }
}

DataMod dataMod = DataMod.Current();

IDynaObject dynaObject = dataMod.GetDynaObject("Invoice");
dynaObject.ParmDict["Dt_Fst"].Value = "2017.01.01";
dynaObject.ParmDict["Dt_Lst"].Value = "2017.12.31";
```



A diagram consisting of two pink ovals. The first oval is located around the `dynaObject` parameter in the `QueryInvo` constructor signature. The second oval is located around the `dynaObject` variable in the assignment `IDynaObject dynaObject = dataMod.GetDynaObject("Invoice");`. A pink arrow points from the first oval down to the second oval, indicating the flow of the object reference.

```
QueryInvo queryInvo = new QueryInvo(dynaObject);  
var query =  
    from invo in queryInvo  
    where invo.Val > 1000  
    orderby invo.DtInvo  
    select invo;  
//Итерации по IEnumerable<Invo>  
foreach (Invo invo in query)  
{  
    Console.WriteLine("{0} {1} {2} {3}",  
        invo.Idn, invo.DtInvo, invo.Val, invo.Note);  
}
```

```
ALTER proc [dbo].[upd_InvoCut]
@Idn int out, @Dt_Invo date, @Val float out, @Note
varchar(100) out
as
if @Val < 0
    select @Val = 0, @Note = 'Начисление - не меньше 0!'
else
    update dbo.T_InvoCut
    set Dt_Invo=@Dt_Invo, Val=@Val, Note=@Note
    where Idn=@Idn
return 0;
```

```
IDynaObject dynaObject = dataMod.GetDynaObject("InvoCut");  
QueryInvo queryInvo = new QueryInvo(dynaObject);  
var query =  
    from invo in queryInvo  
    where invo.Val >= 1000  
    orderby invo.DtInvo  
    select invo;
```

```
Invo first = query.First();  
Console.WriteLine("Old {0} Val:{1} {2} Note:{3}", first.Idn,  
first.DtInvo, first.Val, first.Note);
```

```
first.DtInvo = DateTime.Parse("31.12.2012");  
first.Val = -1500;  
first.Note = "Данные изменены !";  
queryInvo.Update(first);
```

```
Console.WriteLine(dynaObject.GetInfo("props"));
```

```
Console.WriteLine("New {0} {1} Val:{2} Note:{3}", first.Idn,  
first.DtInvo, first.Val, first.Note);
```

Old 503 31.12.2010 Val:1000 Note:Начислено за работу

Id: 28, Name: InvoCut

Prop: Idn, Value: 503 Prop: Dt_Invo, Value: 31.12.2012 Prop:

Val, Value: 0 Prop: Note, Value: Начисление - не меньше 0!

Result: Ok

New 503 31.12.2012 Val:0 Note: Начисление - не меньше 0!

✓ LINQ to Objects

DynaLib vs EF

Скорость загрузки данных и записи в поток.

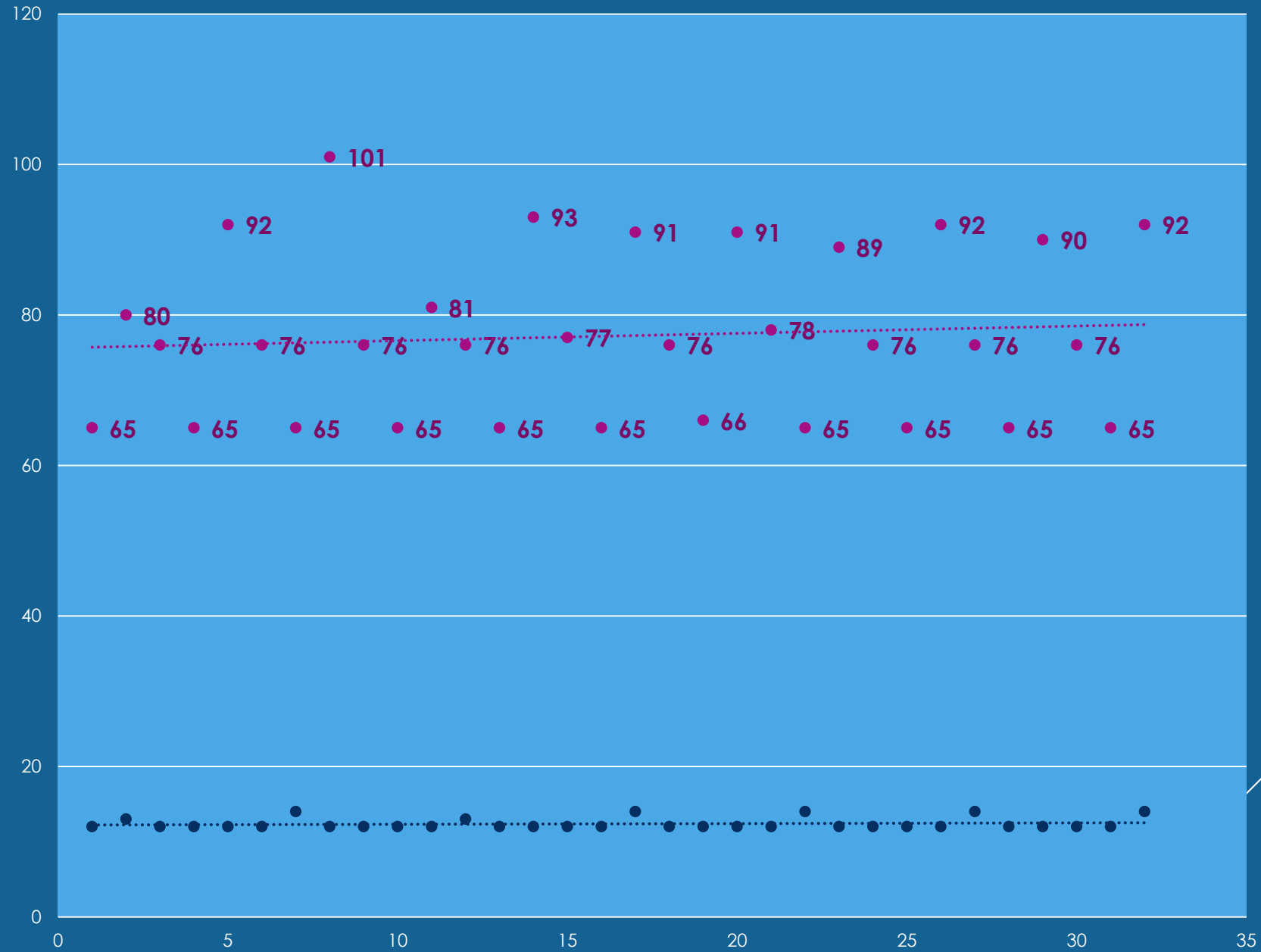


```
timeList.Clear();
for (int i = 0; i < max; i++)
{
    TestM(i);
}
Console.WriteLine("Done M. Время {0} ms.",
    timeList.Skip(1).Average());
using (FileStream fs = new FileStream("TestM.txt",
    FileMode.Create))
{
    StreamWriter sr = new StreamWriter(fs);
    foreach (int t in timeList)
        sr.WriteLine("{0}", t);
    sr.Flush();
    sr.Close();
}
```

```
private static void TestM(int num)
{
    using (var context = new TestModel())
    {
        Stopwatch stopWatch = new Stopwatch();
        stopWatch.Start();
        using (MemoryStream ms = new MemoryStream(1000000))
        {
            DataContractJsonSerializer ser =
                new DataContractJsonSerializer(typeof(IEnumerable<Invo>));
            ser.WriteObject(ms, context.Invos);
        }
        stopWatch.Stop();
        TimeSpan ts = stopWatch.Elapsed;    //Время ~78 ms
        timeList.Add(ts.Milliseconds);
    }
}
```

```
private static void TestM(int num)
{
    Stopwatch stopWatch = new Stopwatch();
    stopWatch.Start();
    //Первоначальная загрузка из БД 4569 записей
    IDynaObject dynaObject = dataMod.GetDynaObject("InvoCut");
    dynaObject.SelectToStream(null);
    stopWatch.Stop();
    TimeSpan ts = stopWatch.Elapsed; //Время 12 ms
    timeList.Add(ts.Milliseconds);
    //dynaObject.StreamWriter.Result.Length);
}
```

77.22 / 12.38 ~ 6.24



37

```
private static void TestR1(int num)
{
    using (var context = new TestModel())
    {
        int count = 0;
        double sum_gt = 0;
        Stopwatch stopWatch = new Stopwatch();
        stopWatch.Start();
        //Первоначальная загрузка из БД
        var query =
            from invo in context.Invos
            where invo.Val > 0
            orderby invo.Dt_Invo
            select invo;
```

```
//LINQ to Entities
foreach (Invo invo in query)
{
    count++;
    sum_gt += invo.Val;
}
stopWatch.Stop();
TimeSpan ts = stopWatch.Elapsed; //Время ~77.2 ms
timeList.Add(ts.Milliseconds);
}
}
```

Время ~ 14.6 ms для R1.	~ 126 917 записей/сек
Время ~ 66.6 ms для R4.	~ 115 283 записей/сек
Время ~133.6 ms для R8.	~ 110 502 записей/сек
Время ~282.3 ms для R16.	~ 105 470 записей/сек

```
private static void TestQ(string queryName, int num)
{
    IDynaObject dynaObject = dataMod.GetDynaObject(queryName);
    //Запрос без параметров
    QueryInvo queryInvo = new QueryInvo(dynaObject);
    int count = 0;
    double sum_gt = 0;
    Stopwatch stopWatch = new Stopwatch();
    stopWatch.Start();
    var query =
        from invo in queryInvo
        where invo.Val > 0
        orderby invo.DtInvo
        select invo;
```

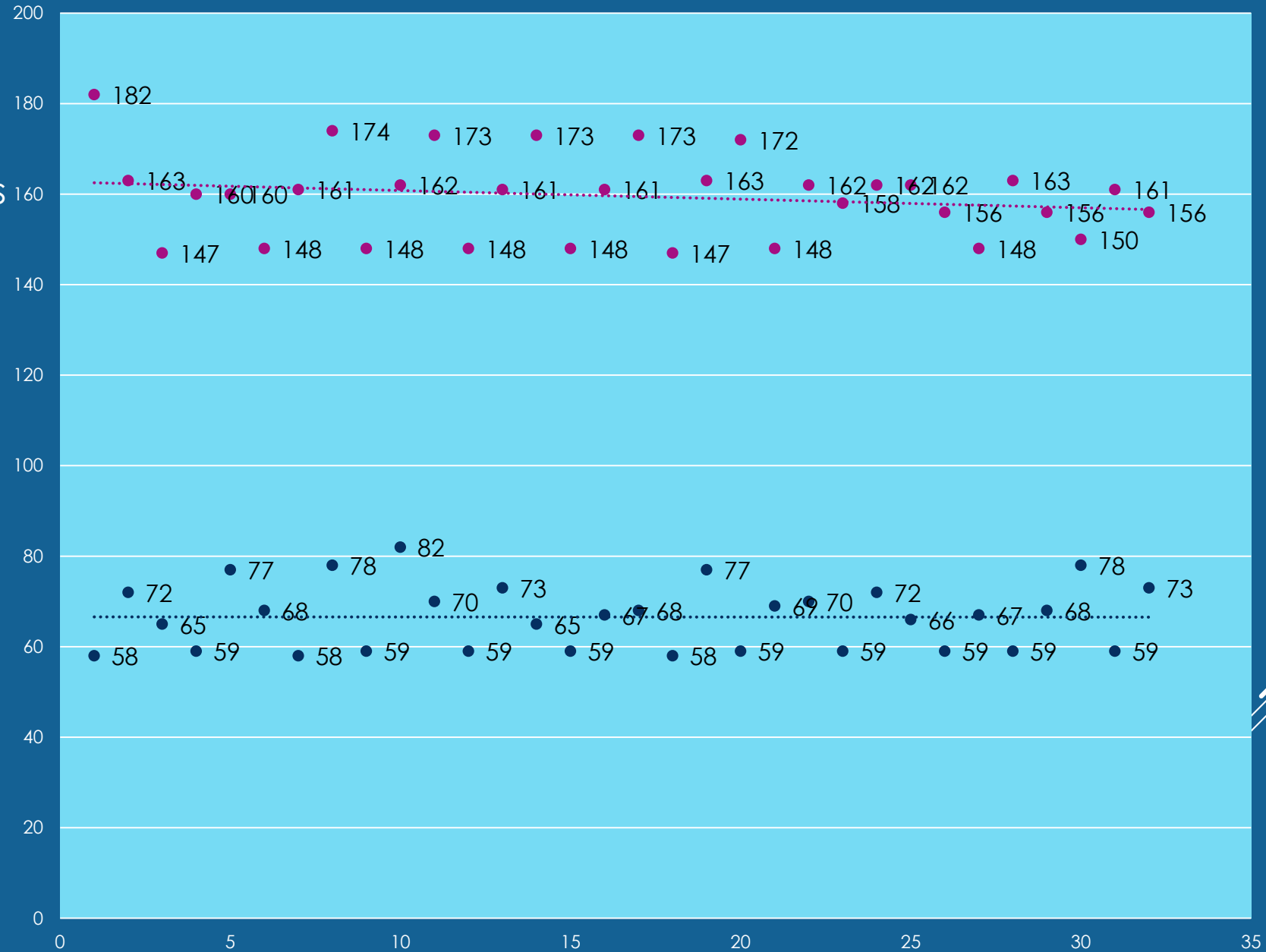


```
//LINQ to Objects
foreach (Invo invo in query)
{
    count++;
    sum_gt += invo.Val;
}
stopWatch.Stop();
TimeSpan ts = stopWatch.Elapsed;
timeList.Add(ts.Milliseconds);
}
```

Время	~ 14.6 ms	для R1.	~ 313 751	записей/сек,	в 2.47 раз
Время	~ 66.6 ms	для R4.	~ 274 579	записей/сек,	в 2.38 раз
Время	~133.6 ms	для R8.	~ 273 488	записей/сек,	в 2.47 раз
Время	~282.3 ms	для R16.	~ 259 004	записей/сек,	в 2.46 раз

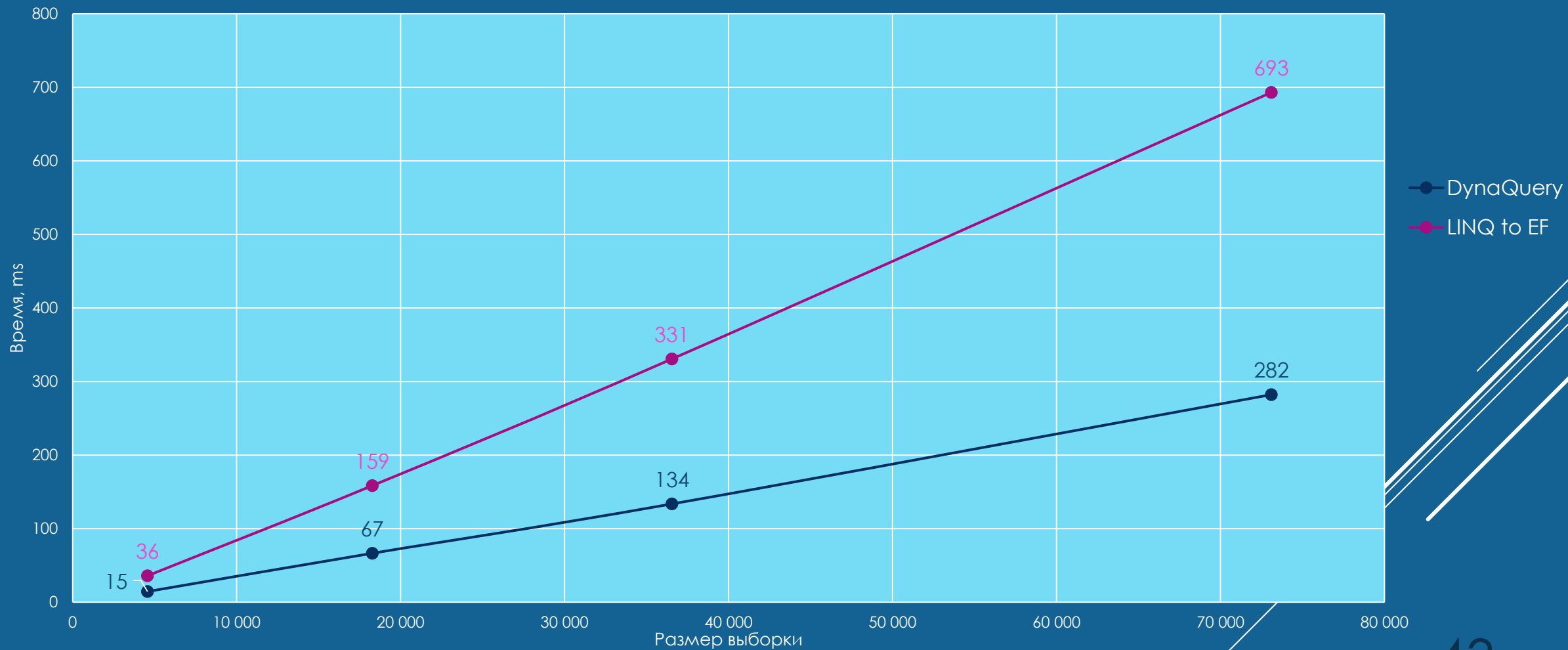
Выборка R4 – 18 276 записей

Среднее 158,53 ms

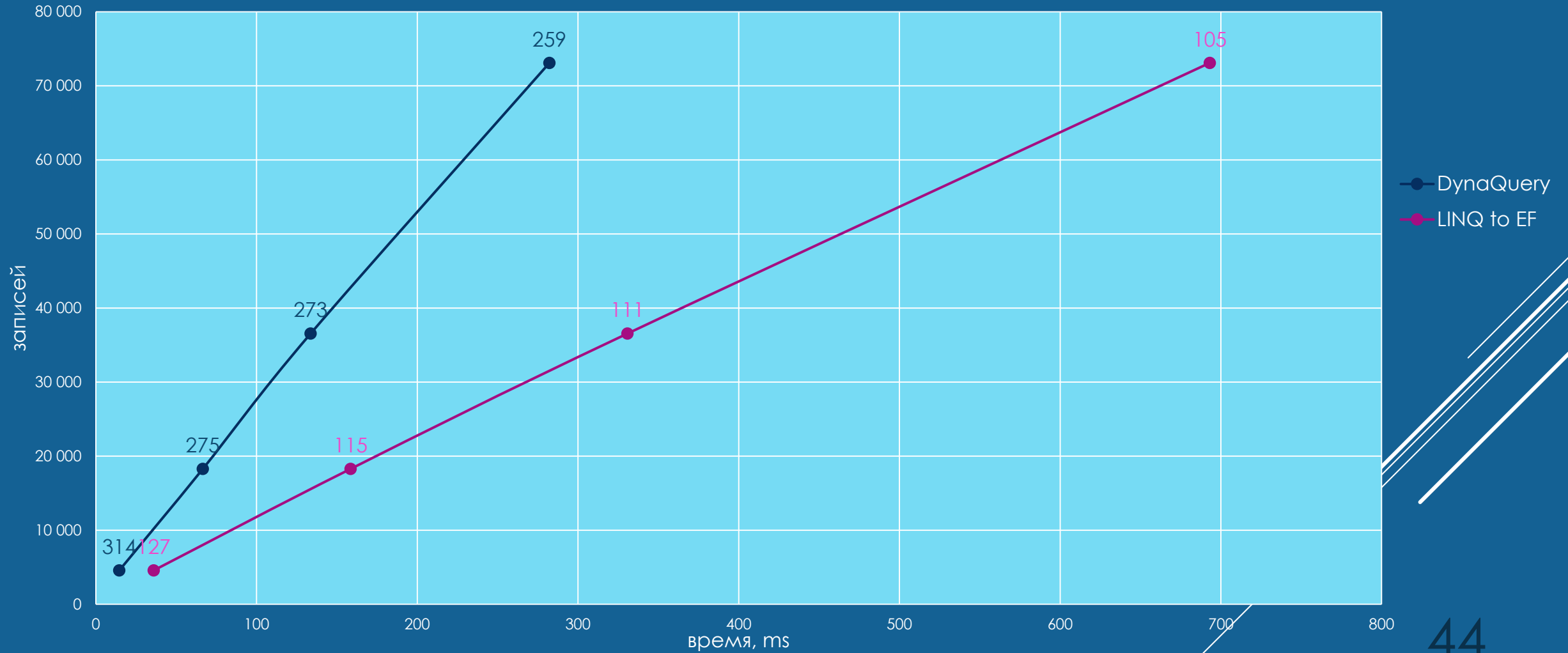


Среднее 66,56 ms

Средние значения затраченного времени от размера выборки



Количество загружаемых записей от времени



- ✓ Нет зависимости от сущностей модели
- ✓ Контроллеры и методы действий
- ✓ Отображение методов HTTP
- ✓ Без привязка моделей
- ✓ Исполнение хранимых процедур
- ✓ Разграничение ООП и функционального стиля
- ✓ Запись данных в поток
- ✓ LINQ to Objects
- ✓ Высокая скорость работы с данными

<https://github.com/Kobdik/DynaRepo>

Кобди́ков Туле́ген, ndi89@mail.ru