Провайдеры типов без боли и магии

Роман Неволин, EPAM


WARNING
F# INSIDE

Задача: я хочу получить JSON с вопросами по URL, взять первый вопрос и напечать ссылку на него

Задача : «детское» решение

1. Скачать текст по ссылке
2. Разбить текст на «блоки»
3. Выделить блоки с вопросом
4. Выбрать первый из этих блоков.
5. Найти в нем ссылку и напечатать

Задача : традиционное C# решение

```
var handler = new HttpClientHandler {
    AutomaticDecompression = DecompressionMethods.GZip
                           | DecompressionMethods.Deflate
};
using (var http = new HttpClient(handler))
{
    var json = http.GetStringAsync(QuestionUrl).Result;
    var questions = JsonConvert
        .DeserializeObject<Result<Question>>(json)
        .Items;
    Console.WriteLine( questions.FirstOrDefault()?.Link );
}
```

Задача : традиционное C# решение

```
var handler = new HttpClientHandler {
    AutomaticDecompression = DecompressionMethods.GZip
                           | DecompressionMethods.Deflate
};
using (var http = new HttpClient(handler))
{
    var json = http.GetStringAsync(QuestionUrl).Result;
    var questions = JsonConvert
        .DeserializeObject<Result<Question>>(json)
        .Items;
    Console.WriteLine( questions.FirstOrDefault()?.Link );
}
```

Задача : традиционное C# решение

```
var handler = new HttpClientHandler {
    AutomaticDecompression = DecompressionMethods.GZip
                           | DecompressionMethods.Deflate
};
using (var http = new HttpClient(handler))
{
    var json = http.GetStringAsync(QuestionUrl).Result;
    var questions = JsonConvert
        .DeserializeObject<Result<Question>>(json)
        .Items;
    Console.WriteLine( questions.FirstOrDefault()?.Link );
}
```

Задача : традиционное C# решение

```
var handler = new HttpClientHandler {
    AutomaticDecompression = DecompressionMethods.GZip
                           | DecompressionMethods.Deflate
};
using (var http = new HttpClient(handler))
{
    var json = http.GetStringAsync(QuestionUrl).Result;
    var questions = JsonConvert
        .DeserializeObject<Result<Question>>(json)
        .Items;
    Console.WriteLine( questions.FirstOrDefault()?.Link );
}
```

Задача : традиционное C# решение

```
var questions = DeserializeJsonFromUrl<Result<Question>>
    (QuestionUrl).Items;
Console.WriteLine(questions.FirstOrDefault()?.Link);
```

Задача : традиционное C# решение

```
public class Result<T>
{
    1 reference
    public T[] Items { get; set; }
    0 references
    public bool HasMore { get; set; }
    0 references
    public int QuotaMax { get; set; }
    0 references
    public int QuotaRemaining { get; set; }
}
```

Задача : традиционное C# решение

```
public class Result<T>
{
    1 reference
    public T[] Items { get; set; }
    0 references
    public bool HasMore { get; set; }
    0 references
    public int QuotaMax { get; set; }
    0 references
    public int QuotaRemaining { get; set; }
}
```

```
public class Question
{
    0 references
    public string[] Tags { get; set; }
    0 references
    public User Owner { get; set; }
    0 references
    public bool IsAnswered { get; set; }
    0 references
    public int ViewCount { get; set; }
    0 references
    public int AnswerCount { get; set; }
    0 references
    public DateTime ProtectedDate { get; set; }
    0 references
    public long AcceptedAnswerId { get; set; }
    0 references
    public int Score { get; set; }
    0 references
    public DateTime LastActivityDate { get; set; }
    0 references
    public DateTime CreationDate { get; set; }
    0 references
    public DateTime LastEditDate { get; set; }
    0 references
    public long QuestionId { get; set; }
    1 reference
    public string Link { get; set; }
    0 references
    public string Title { get; set; }
}
```

Задача : традиционное C# решение

```
public class Result<T>
{
    1 reference
    public T[] Items { get; set; }
    0 references
    public bool HasMore { get; set; }
    0 references
    public int QuotaMax { get; set; }
    0 references
    public int QuotaRemaining { get; set; }
}
```

```
public class User
{
    0 references
    public int Reputation { get; set; }
    0 references
    public long UserId { get; set; }
    0 references
    public string UserType { get; set; }
    0 references
    public int AcceptRate { get; set; }
    0 references
    public string ProfileImage { get; set; }
    0 references
    public string DisplayName { get; set; }
    0 references
    public string Link { get; set; }
}
```

```
public class Question
{
    0 references
    public string[] Tags { get; set; }
    0 references
    public User Owner { get; set; }
    0 references
    public bool IsAnswered { get; set; }
    0 references
    public int ViewCount { get; set; }
    0 references
    public int AnswerCount { get; set; }
    0 references
    public DateTime ProtectedDate { get; set; }
    0 references
    public long AcceptedAnswerId { get; set; }
    0 references
    public int Score { get; set; }
    0 references
    public DateTime LastActivityDate { get; set; }
    0 references
    public DateTime CreationDate { get; set; }
    0 references
    public DateTime LastEditDate { get; set; }
    0 references
    public long QuestionId { get; set; }
    1 reference
    public string Link { get; set; }
    0 references
    public string Title { get; set; }
}
```

Задача : «динамическое» решение

```
dynamic questions = DeserializeJsonFromUrl
    (QuestionUrl).Items;
Console.WriteLine(questions.FirstOrDefault()?.Link);
```

Что здесь не так?

- Мы легко можем ошибиться, и об ошибке станет известно только в рантайме

Что здесь не так?

- Мы легко можем ошибиться, и об ошибке станет известно только в рантайме
- Автодополнение не помогает нам писать код

Что здесь не так?

- Мы легко можем ошибиться, и об ошибке станет известно только в рантайме
- Автодополнение не помогает нам писать код
- Мы никак не застрахованы от изменений в API

Ах, как было бы прекрасно, если..

- Если бы типы генерировались по JSON

Ах, как было бы прекрасно, если..

- Если бы типы генерировались по JSON
- Если бы это происходило автоматически, как только мы ввели в IDE ссылку на JSON

Ах, как было бы прекрасно, если..

- Если бы типы генерировались по JSON
- Если бы это происходило автоматически, как только мы ввели в IDE ссылку на JSON
- Если бы сгенерированные типы автоматически обновлялись при изменении JSON

В общем, провайдеры — это отлично

В общем, провайдеры — это отлично,
но...

Но...

- Но провайдеры – сложный механизм с кучей подводных камней.

Но...

- Но провайдеры – сложный механизм с кучей подводных камней.
- Но бывает сложно понять, когда выполняется та или иная логика внутри провайдера

Но...

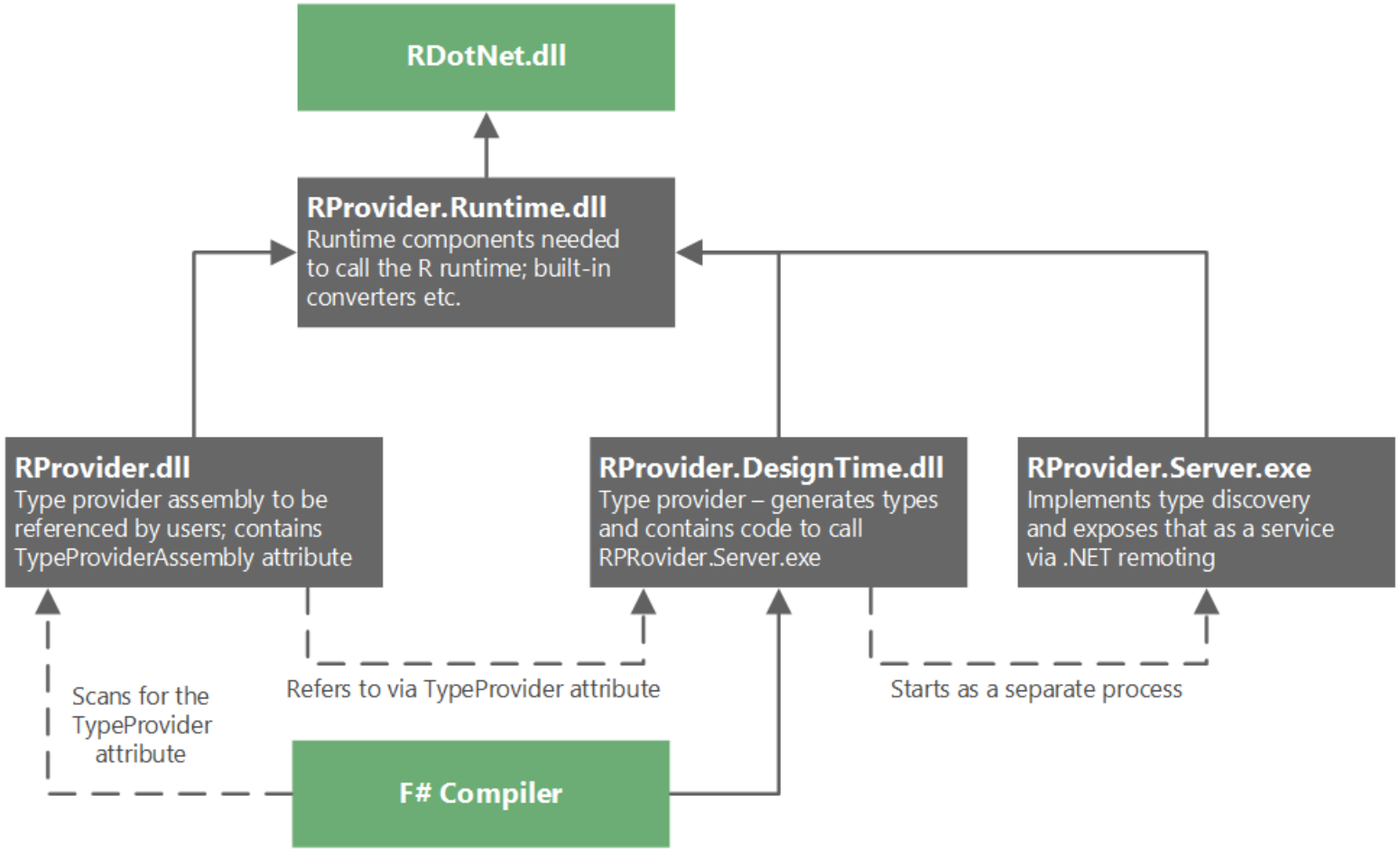
- Но провайдеры – сложный механизм с кучей подводных камней.
- Но бывает сложно понять, когда выполняется та или иная логика внутри провайдера
- Но провайдеры недостаточно хорошо специфицированы.

Но...

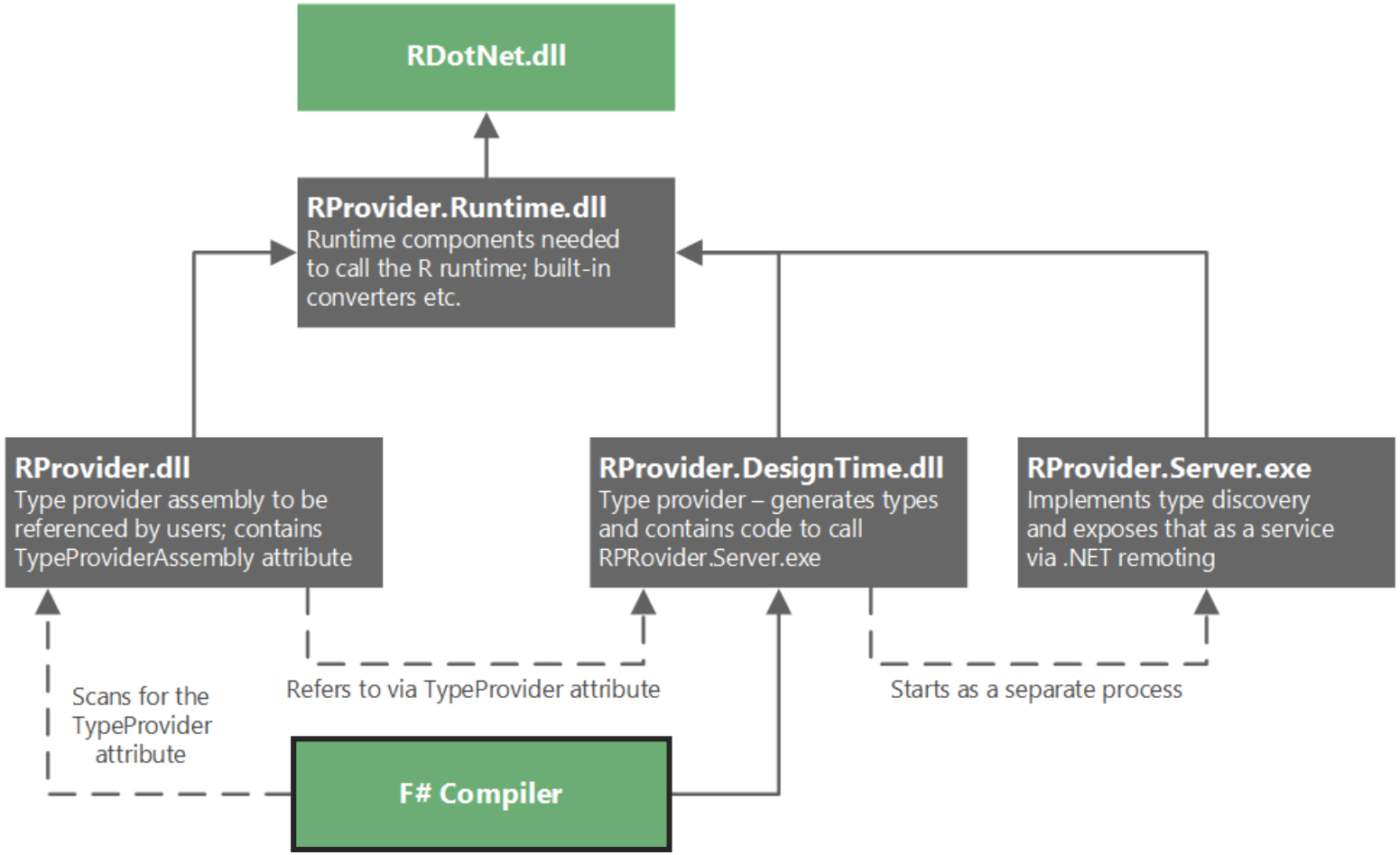
- Но провайдеры – сложный механизм с кучей подводных камней.
- Но бывает сложно понять, когда выполняется та или иная логика внутри провайдера
- Но провайдеры недостаточно хорошо специфицированы.

Но вообще-то. Программисты мы или погулять вышли?

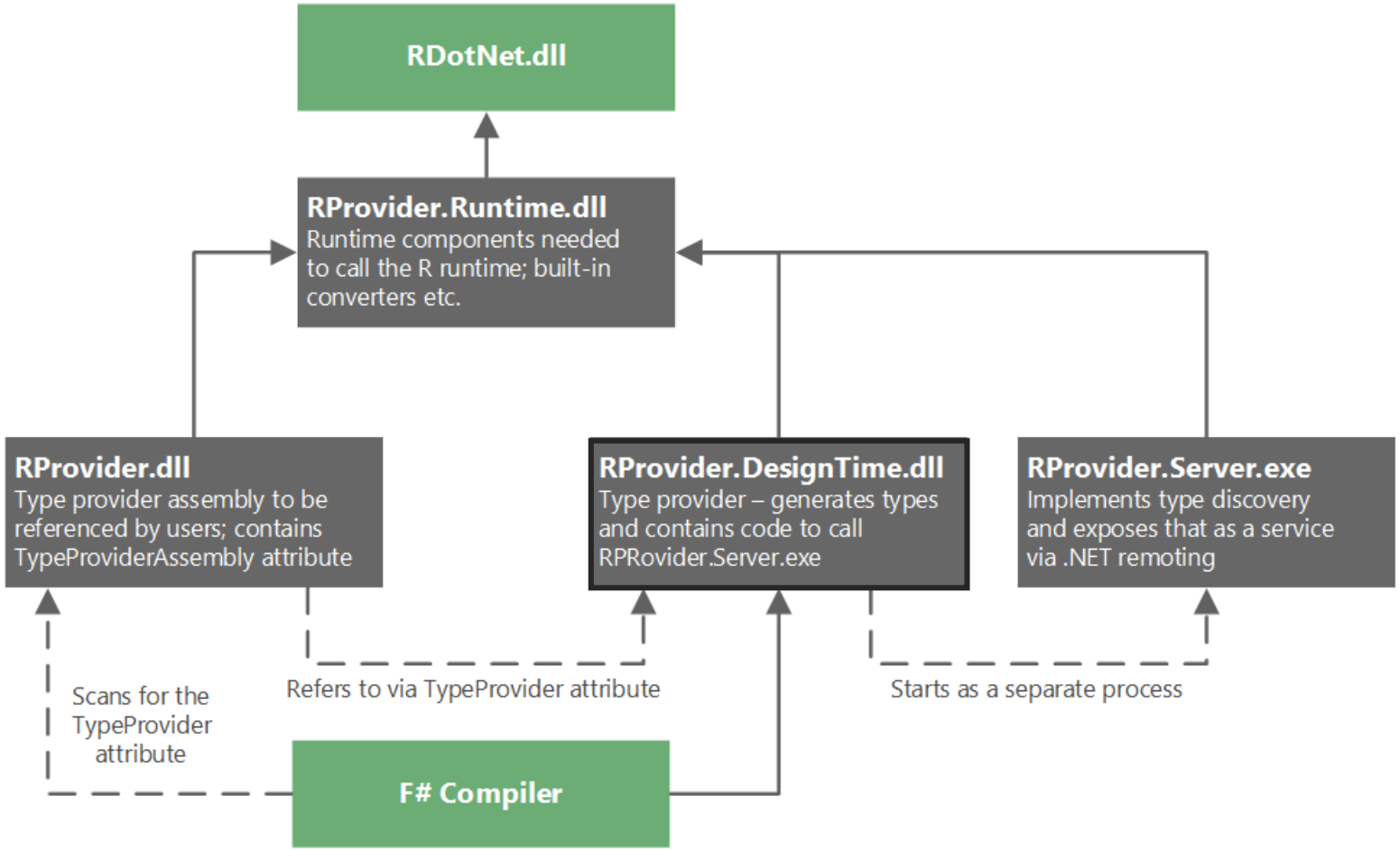
Высокоуровневая схема R Provider



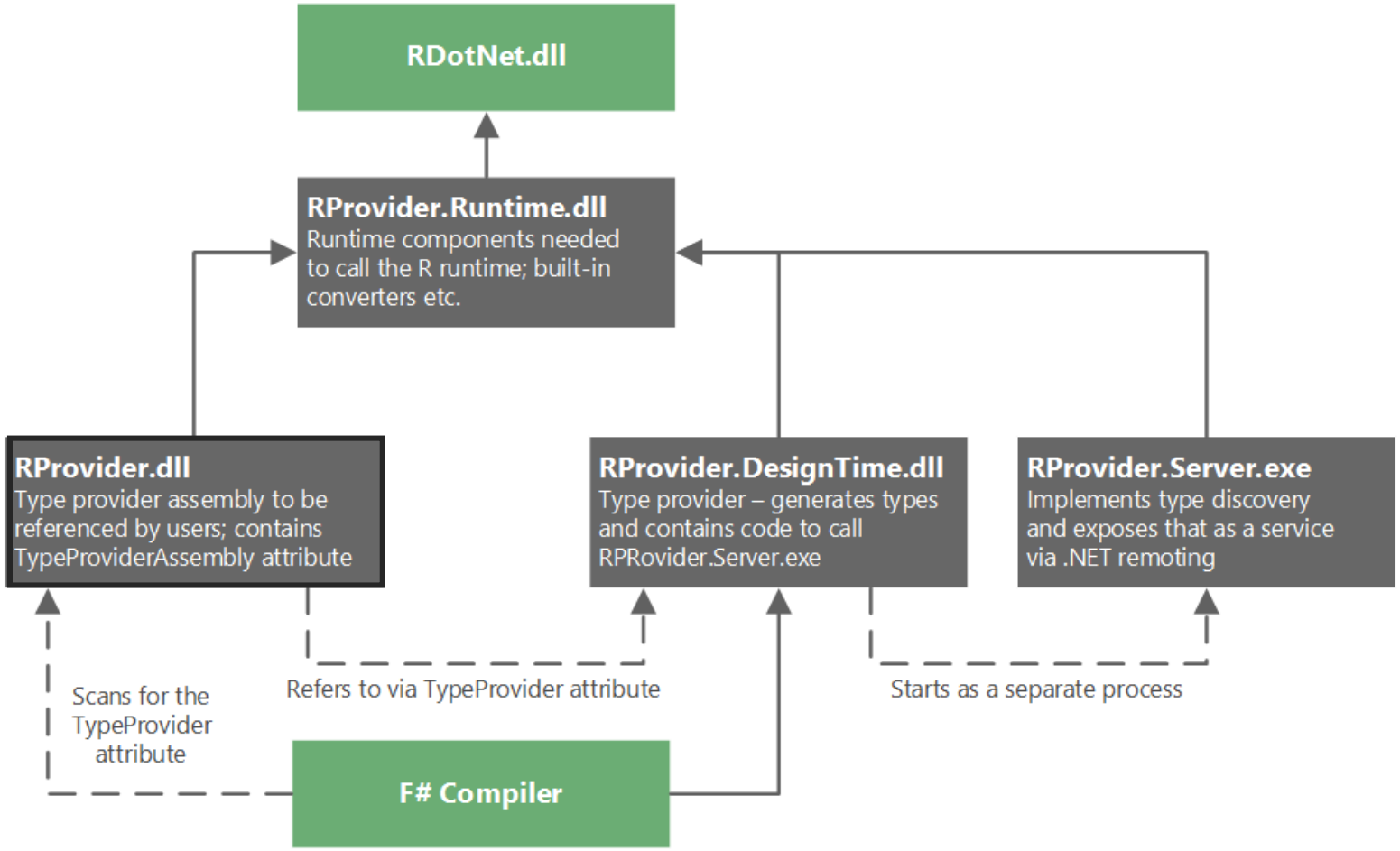
Высокоуровневая схема R Provider



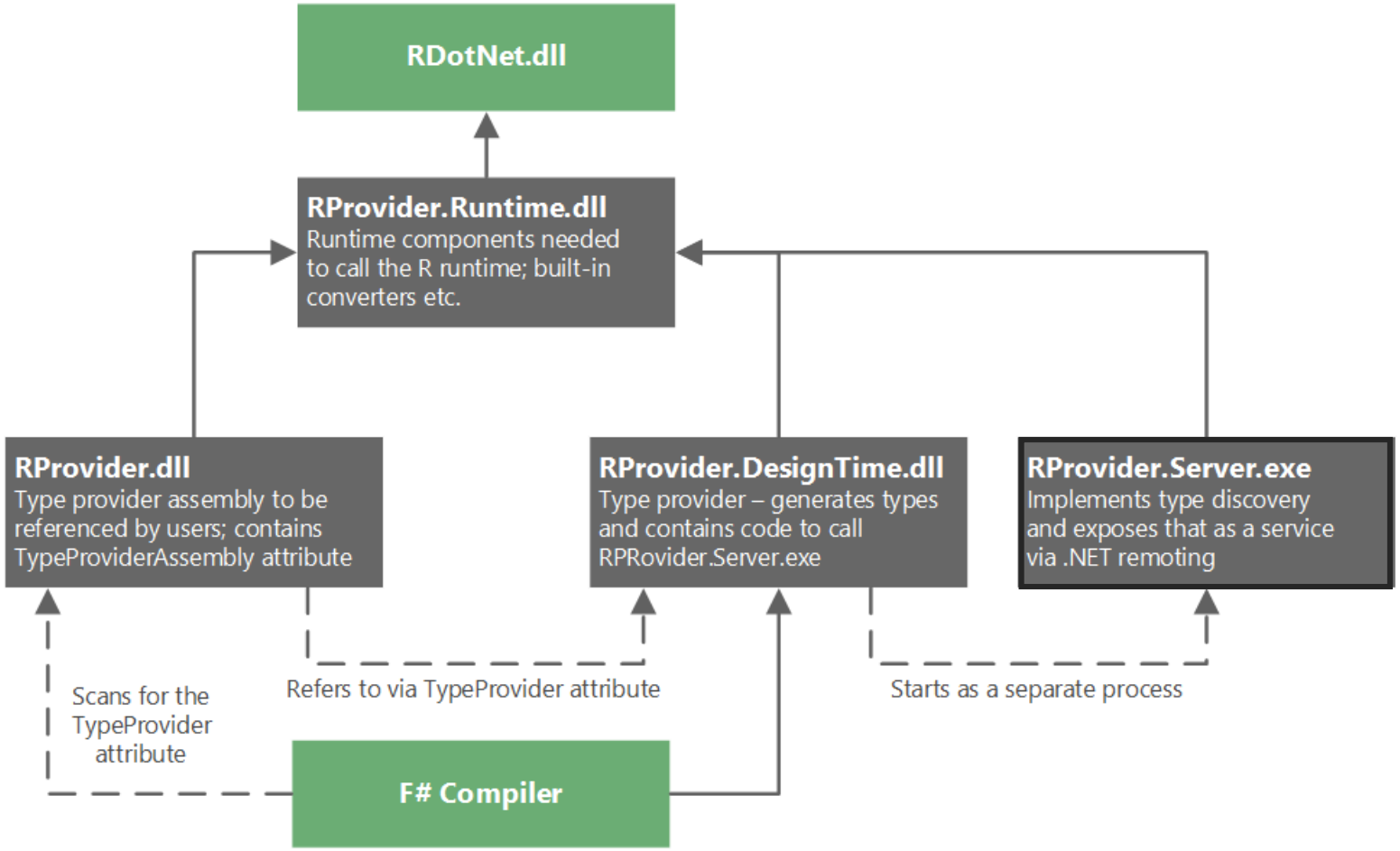
Высокоуровневая схема R Provider



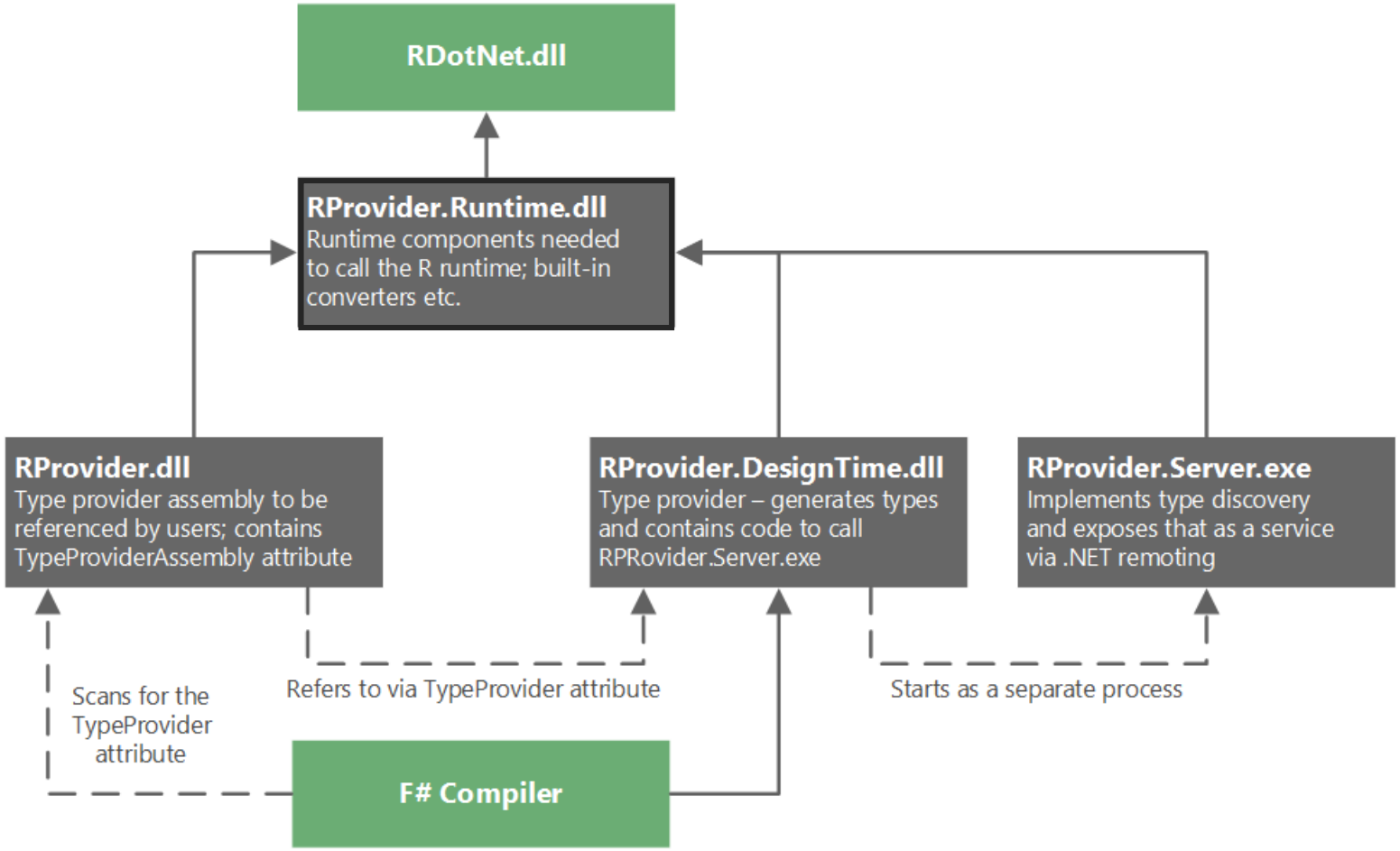
Высокоуровневая схема R Provider



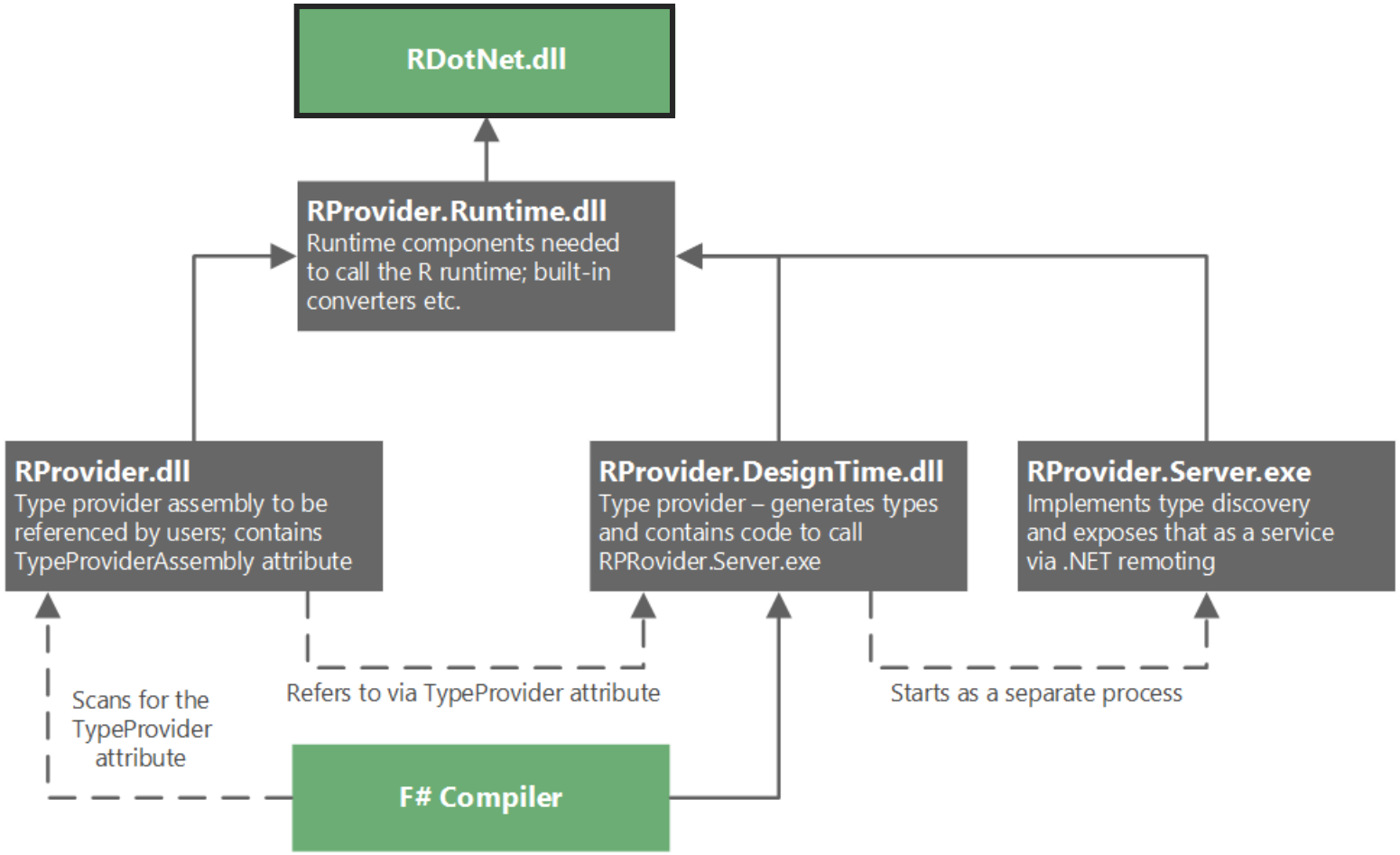
Высокоуровневая схема R Provider

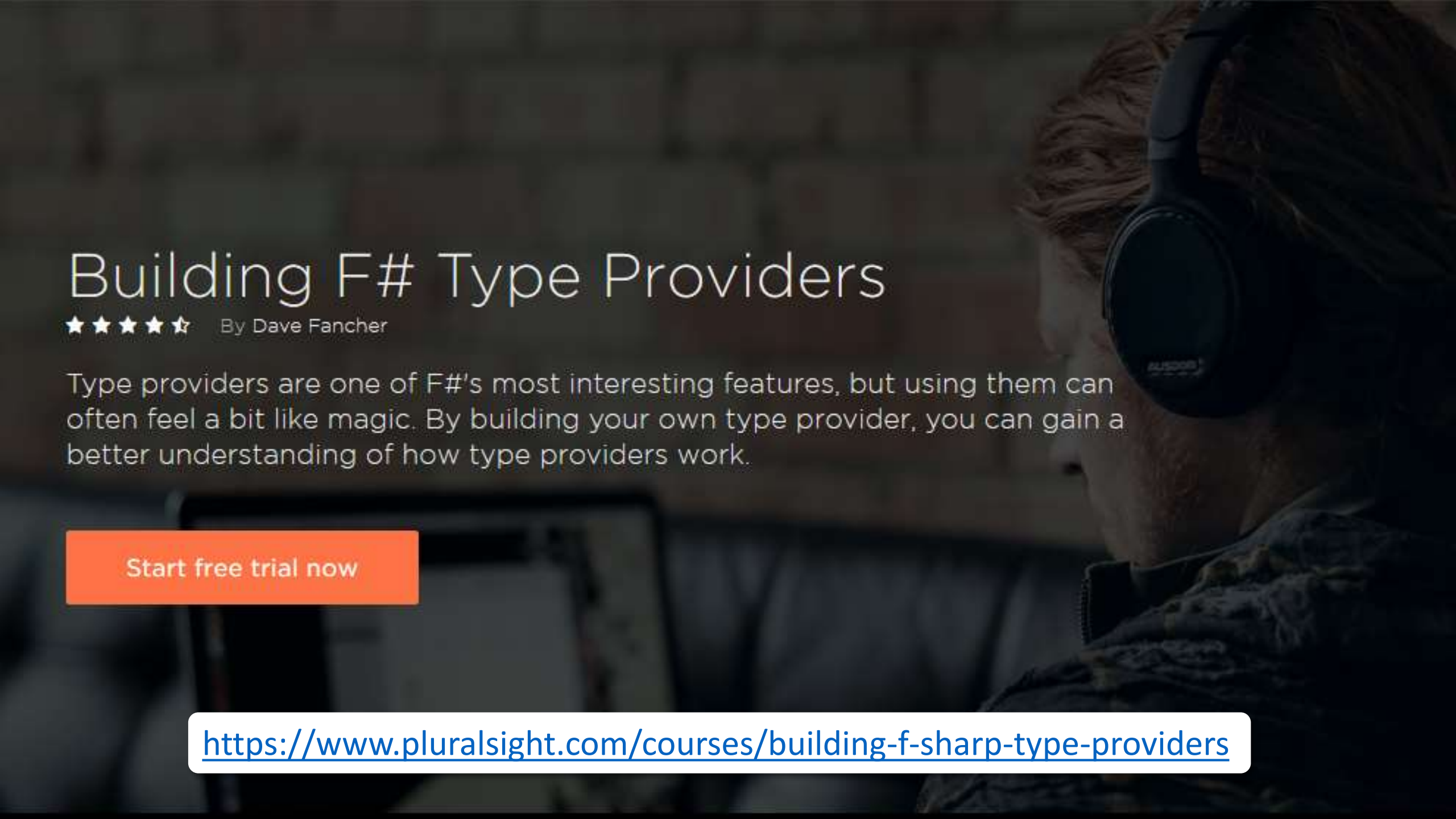


Высокоуровневая схема R Provider



Высокоуровневая схема R Provider





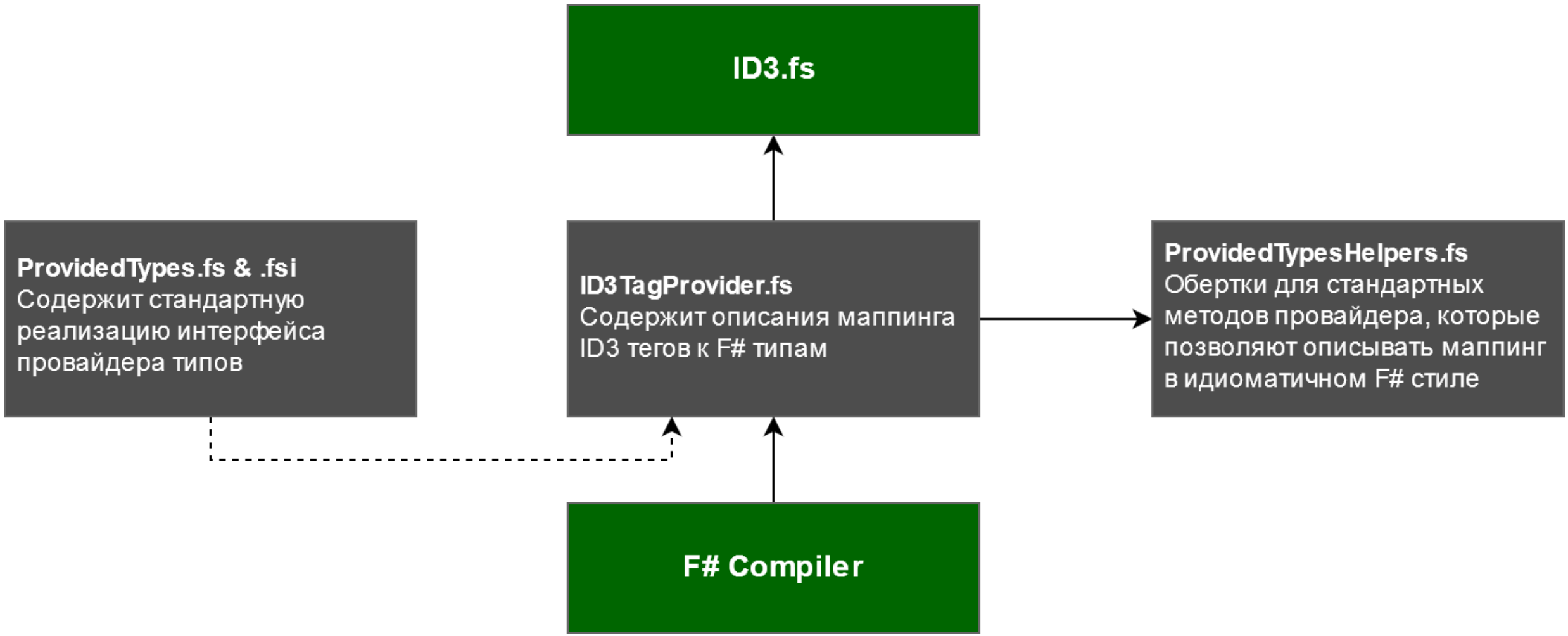
https://www.pluralsight.com/courses/building-f-sharp-type-providers

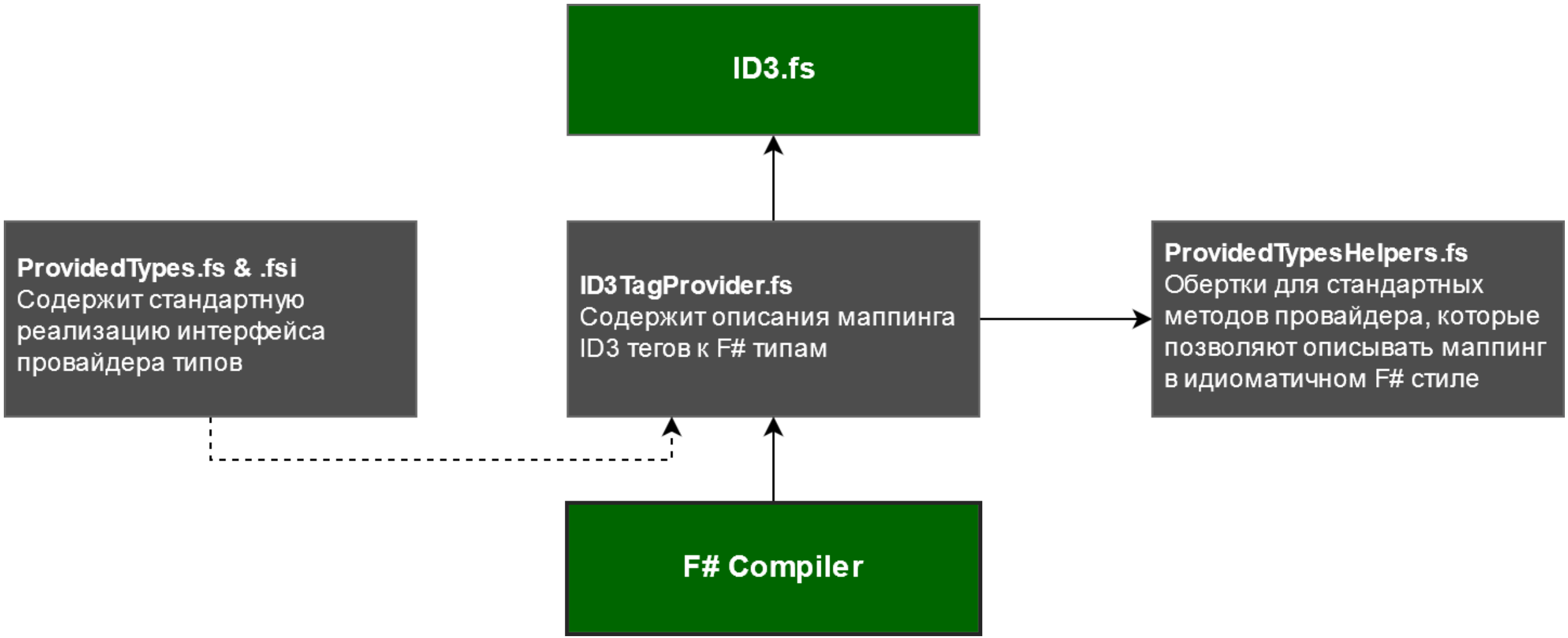
ID3 Provider : обзор задачи

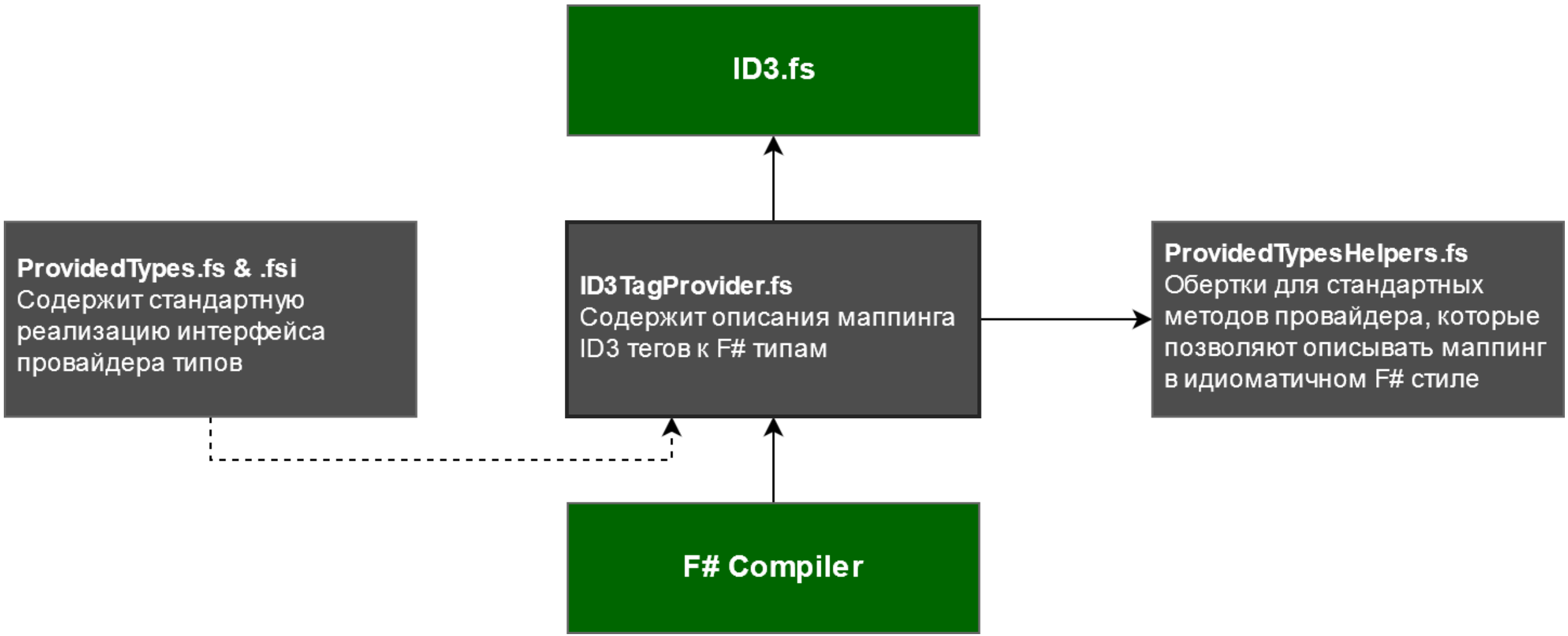
- Информация об MP3 файлах хранится в ID3 тегах
- Разные теги хранят в себе разные данные
- MP3 файл может содержать или не содержать любые ID3 теги

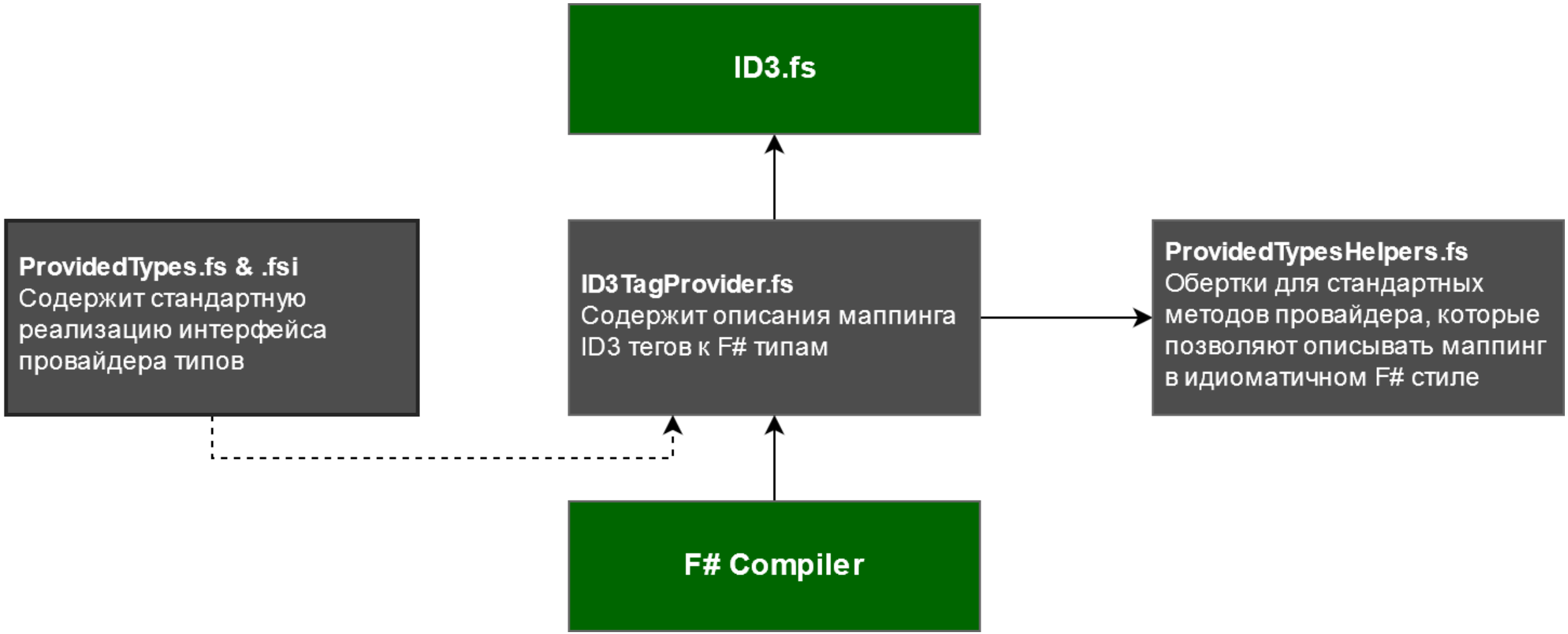
Это задача для провайдеров!

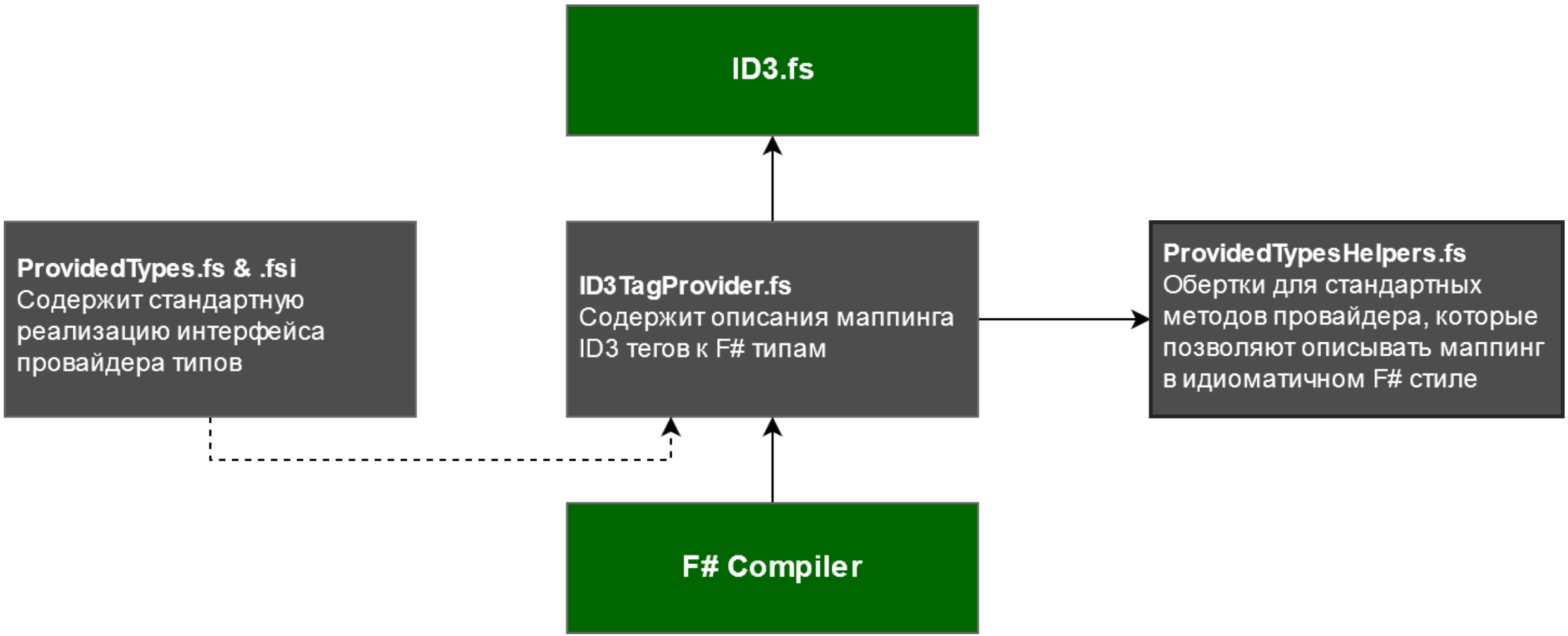

ID3.fs
ID3Provider.fs
ID3TagProvider.fsproj
ProvidedTypes.fs
ProvidedTypes.fsi
ProvidedTypesHelpers.fs

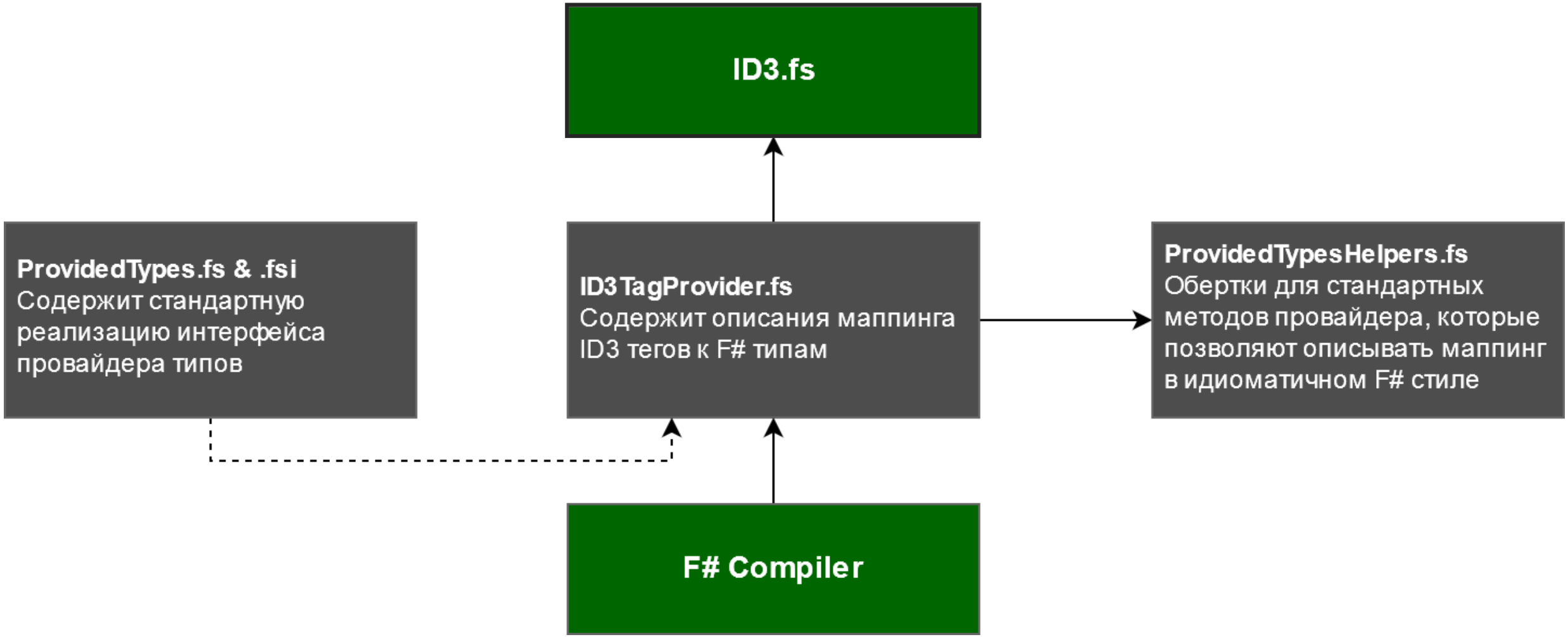

ID3.fs
ProvidedTypes.fs & .fsi
Содержит стандартную реализацию интерфейса провайдера типов
ID3TagProvider.fs
Содержит описания маппинга ID3 тегов к F# типам
ProvidedTypesHelpers.fs
Обертки для стандартных методов провайдера, которые позволяют описывать маппинг в идиоматичном F# стиле
F# Compiler


ID3.fs
ProvidedTypes.fs & .fsi
Содержит стандартную реализацию интерфейса провайдера типов
ID3TagProvider.fs
Содержит описания маппинга ID3 тегов к F# типам
ProvidedTypesHelpers.fs
Обертки для стандартных методов провайдера, которые позволяют описывать маппинг в идиоматичном F# стиле
F# Compiler


ID3.fs
ProvidedTypes.fs & .fsi
Содержит стандартную реализацию интерфейса провайдера типов
ID3TagProvider.fs
Содержит описания маппинга ID3 тегов к F# типам
ProvidedTypesHelpers.fs
Обертки для стандартных методов провайдера, которые позволяют описывать маппинг в идиоматичном F# стиле
F# Compiler


ID3.fs
ProvidedTypes.fs & .fsi
Содержит стандартную реализацию интерфейса провайдера типов
ID3TagProvider.fs
Содержит описания маппинга ID3 тегов к F# типам
ProvidedTypesHelpers.fs
Обертки для стандартных методов провайдера, которые позволяют описывать маппинг в идиоматичном F# стиле
F# Compiler


ID3.fs
ProvidedTypes.fs & .fsi
Содержит стандартную реализацию интерфейса провайдера типов
ID3TagProvider.fs
Содержит описания маппинга ID3 тегов к F# типам
ProvidedTypesHelpers.fs
Обертки для стандартных методов провайдера, которые позволяют описывать маппинг в идиоматичном F# стиле
F# Compiler


ID3.fs
ProvidedTypes.fs & .fsi
Содержит стандартную реализацию интерфейса провайдера типов
ID3TagProvider.fs
Содержит описания маппинга ID3 тегов к F# типам
ProvidedTypesHelpers.fs
Обертки для стандартных методов провайдера, которые позволяют описывать маппинг в идиоматичном F# стиле
F# Compiler

Демо: смотрим на код ID3 Tag Provider

Так все-таки.
Существуют на самом деле сгенерированные типы или нет?

Generative провайдеры

- Создает реальные типы на основании схемы
- Созданные типы хранятся в файле сборки
- Совместим с рефлексией и другими .NET языками
- Увеличивают размер сборки

Erased провайдеры

- Создает типы, о которых знает только F# компилятор
- Невозможно использовать рефлексию
- Несовместимы с другими .NET языками
- Умеют работать с бесконечными структурами данных

Code Quotations

```
let buildExpr tag =
  fun [tags] ->
      <@@ (
          ((%%tags:obj) :?> Dictionary<string, ID3Frame>).
            [tag]).GetContent() |> unbox
      @@>
```

Code Quotations

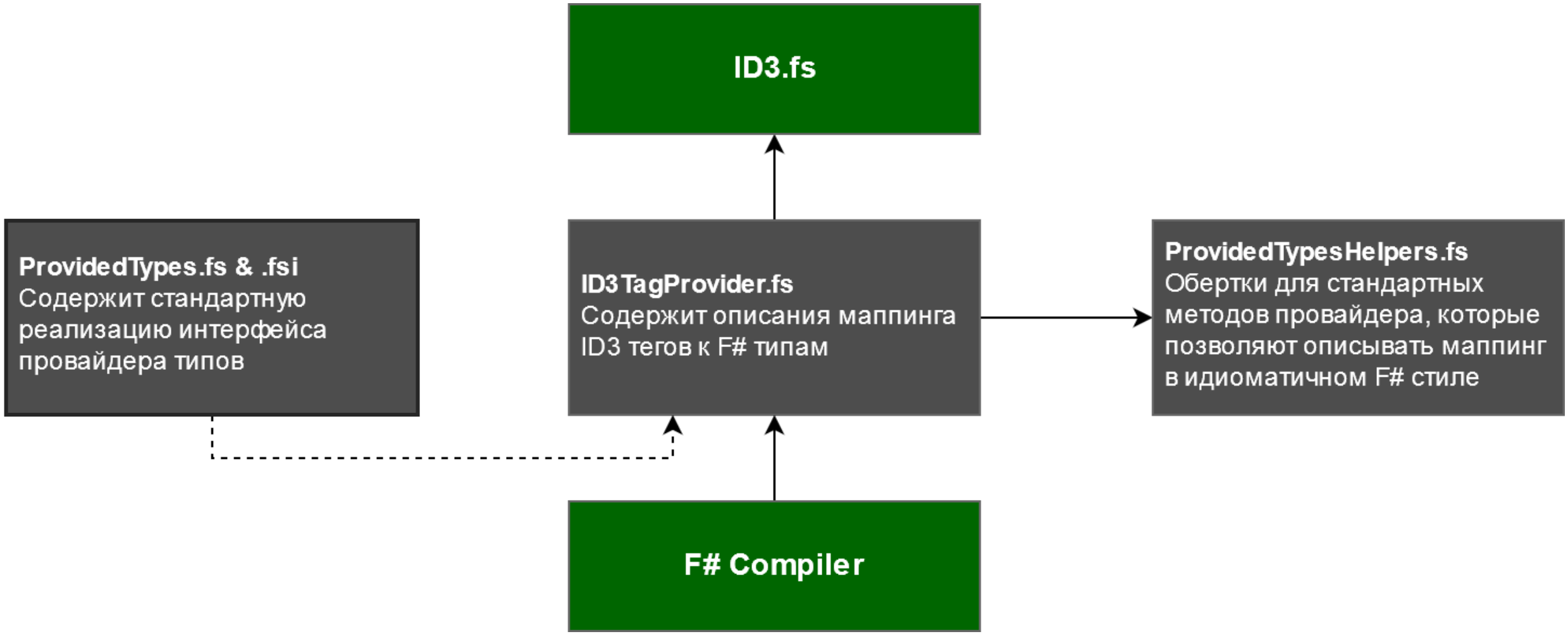
- Механизм метапрограммирования в F#, схожий с деревьями выражений
- Полностью валидируется компилятором
- Используется для создания «шаблонов» выражений

Code Quotations

```
let buildExpr tag =
  fun [tags] ->
      <@@ (
          ((%%tags:obj) :?> Dictionary<string, ID3Frame>).
            [tag]).GetContent() |> unbox
      @@>
```

Демо: декомпилируем провайдеры

Но как это все сгенерировалось?


ID3.fs
ProvidedTypes.fs & .fsi
Содержит стандартную реализацию интерфейса провайдера типов
ID3TagProvider.fs
Содержит описания маппинга ID3 тегов к F# типам
ProvidedTypesHelpers.fs
Обертки для стандартных методов провайдера, которые позволяют описывать маппинг в идиоматичном F# стиле
F# Compiler

TypeProviderForNamespaces

```
[<TypeProvider>]
type ID3TagProvider() as this =
  inherit TypeProviderForNamespaces()
```

TypeProviderForNamespaces

```
[<TypeProvider>]
type ID3TagProvider() as this =
  inherit TypeProviderForNamespaces()
```

TypeProviderForNamespaces

```
[<TypeProvider>]
type ID3TagProvider() as this =
  inherit TypeProviderForNamespaces()
```

```
/// A base type providing default implementations of type provider
  functionality when all provided
/// types are of type ProvidedTypeDefinition.
type TypeProviderForNamespaces =
```

TypeProviderForNamespaces

```
    /// A base type providing default implementations of type provider functionality when all provided
    /// types are of type ProvidedTypeDefinition.
    type TypeProviderForNamespaces =
        new: config: TypeProviderConfig * namespaceName:string * types: ProvidedTypeDefinition list * ?
          assemblyReplacementMap: (string * string) list -> TypeProviderForNamespaces
        new: config: TypeProviderConfig * ?assemblyReplacementMap: (string * string) list -> TypeProviderForNamespaces
        member AddNamespace: namespaceName:string * types: ProvidedTypeDefinition list -> unit
        member Namespaces: IProvidedNamespace[]
        member Invalidate: unit -> unit
        member GetStaticParametersForMethod: MethodBase -> ParameterInfo[]
        member ApplyStaticArgumentsForMethod: MethodBase * string * obj[] -> MethodBase
#if FX_NO_LOCAL_FILESYSTEM
        abstract ResolveAssembly: ResolveEventArgs -> Assembly
        default ResolveAssembly: ResolveEventArgs -> Assembly
        member RegisterProbingFolder: folder: string -> unit
        member RegisterRuntimeAssemblyLocationAsProbingFolder: config: TypeProviderConfig -> unit
#endif
#if !NO_GENERATIVE
        member RegisterGeneratedTargetAssembly: fileName: string -> Assembly
#endif
        [<CLIEvent>]
        member Disposing: IEvent<EventHandler,EventArgs>
        member TargetContext: ProvidedTypesContext

        interface ITypeProvider
```

TypeProviderForNamespaces

```
/// A base type providing default implementations of type provider functionality when all provided
/// types are of type ProvidedTypeDefinition.
type TypeProviderForNamespaces =
    new: config: TypeProviderConfig * namespaceName:string * types: ProvidedTypeDefinition list * ?
      assemblyReplacementMap: (string * string) list -> TypeProviderForNamespaces
    new: config: TypeProviderConfig * ?assemblyReplacementMap: (string * string) list -> TypeProviderForNamespaces
    member AddNamespace: namespaceName:string * types: ProvidedTypeDefinition list -> unit
    member Namespaces: IProvidedNamespace[]
    member Invalidate: unit -> unit
    member GetStaticParametersForMethod: MethodBase -> ParameterInfo[]
    member ApplyStaticArgumentsForMethod: MethodBase * string * obj[] -> MethodBase
#if FX_NO_LOCAL_FILESYSTEM
    abstract ResolveAssembly: ResolveEventArgs -> Assembly
    default ResolveAssembly: ResolveEventArgs -> Assembly
    member RegisterProbingFolder: folder: string -> unit
    member RegisterRuntimeAssemblyLocationAsProbingFolder: config: TypeProviderConfig -> unit
#endif
#if !NO_GENERATIVE
    member RegisterGeneratedTargetAssembly: fileName: string -> Assembly
#endif
    [<CLIEvent>]
    member Disposing: IEvent<EventHandler,EventArgs>
    member TargetContext: ProvidedTypesContext
```

interface ITypeProvider

Интерфейс `ITypeProvider`

«*Type providers implement this interface in order to be recognized by the compiler as an F# type provider. The implementation of this interface determines the public interface and behavior of the type provider*»

TypeProviderForNamespaces

```
/// Invoked by the type provider to add a namespace of provided types
  in the specification of the type provider.
member AddNamespace: namespaceName:string * types:
  ProvidedTypeDefinition list -> unit
```

TypeProviderForNamespaces

```
/// Invoked by the type provider to add a namespace of provided types
  in the specification of the type provider.
member AddNamespace: namespaceName:string * types:
  ProvidedTypeDefinition list -> unit

member __.AddNamespace (namespaceName, types) =
    namespacesT.Add (makeProvidedNamespace namespaceName types)

member __.Namespaces = namespacesT.ToArray()
```

Определение базового типа

```
let ns = "DidacticCode.TypeProviders"
let assy = Assembly.GetExecutingAssembly()

let audioFileType = ProvidedTypeDefinition(assy, ns, "AudioFile", None)
```

Определение базового типа

```
let ns = "DidacticCode.TypeProviders"
let assy = Assembly.GetExecutingAssembly()

let audioFileType = ProvidedTypeDefinition(assy, ns, "AudioFile", None)
```

Определение базового типа

```
let ns = "DidacticCode.TypeProviders"
let assy = Assembly.GetExecutingAssembly()

let audioFileType = ProvidedTypeDefinition(assy, ns, "AudioFile", None)


/// Represents a provided type definition.
[<Class>]
type ProvidedTypeDefinition =
    inherit TypeDelegator
```

Статические параметры

```
audioFileType.DefineStaticParameters(
  [ ProvidedStaticParameter("fileName", typeof<string>) ],
  instantiationFunction = (
    fun typeName [| :? string as fileName |] ->
```

Статические параметры

```
audioFileType.DefineStaticParameters(
  [ ProvidedStaticParameter("fileName", typeof<string>) ],
  instantiationFunction = (
    fun typeName [| :? string as fileName |] ->

/// Abstract a type to a parametric-type. Requires "formal parameters" and
  "instantiation function".
member __.DefineStaticParameters(parameters: ProvidedStaticParameter list,
  instantiationFunction: (string -> obj[] -> ProvidedTypeDefinition)) =
    if staticParamsDefined then failwithf "Static parameters have already
      been defined for this type. stacktrace = %A" Environment.StackTrace
    staticParamsDefined <- true
    staticParams      <- parameters
    staticParamsApply <- Some instantiationFunction
```

Статические параметры

```
audioFileType.DefineStaticParameters(
  [ ProvidedStaticParameter("fileName", typeof<string>) ],
  instantiationFunction = (
    fun typeName [| :? string as fileName |] ->

/// Abstract a type to a parametric-type. Requires "formal parameters" and
  "instantiation function".
member __.DefineStaticParameters(parameters: ProvidedStaticParameter list,
  instantiationFunction: (string -> obj[] -> ProvidedTypeDefinition)) =
    if staticParamsDefined then failwithf "Static parameters have already
      been defined for this type. stacktrace = %A" Environment.StackTrace
    staticParamsDefined <- true
    staticParams      <- parameters
    staticParamsApply <- Some instantiationFunction
```

Статические параметры

```
audioFileType.DefineStaticParameters(
  [ ProvidedStaticParameter("fileName", typeof<string>) ],
  instantiationFunction = (
    fun typeName [| :? string as fileName |] ->

/// Abstract a type to a parametric-type. Requires "formal parameters" and
  "instantiation function".
member __.DefineStaticParameters(parameters: ProvidedStaticParameter list,
  instantiationFunction: (string -> obj[] -> ProvidedTypeDefinition)) =
    if staticParamsDefined then failwithf "Static parameters have already
      been defined for this type. stacktrace = %A" Environment.StackTrace
    staticParamsDefined <- true
    staticParams      <- parameters
    staticParamsApply <- Some instantiationFunction
```

Статические параметры

```
audioFileType.DefineStaticParameters(
  [ ProvidedStaticParameter("fileName", typeof<string>) ],
  instantiationFunction = (
    fun typeName [| :? string as fileName |] ->

/// Abstract a type to a parametric-type. Requires "formal parameters" and
  "instantiation function".
member __.DefineStaticParameters(parameters: ProvidedStaticParameter list,
  instantiationFunction: (string -> obj[] -> ProvidedTypeDefinition)) =
    if staticParamsDefined then failwithf "Static parameters have already
      been defined for this type. stacktrace = %A" Environment.StackTrace
    staticParamsDefined <- true
    staticParams      <- parameters
    staticParamsApply <- Some instantiationFunction
```

Статические параметры

```
/// Represents a provided static parameter.
[<Class>]
type ProvidedStaticParameter =
    inherit ParameterInfo
    new: parameterName: string * parameterType: Type * ?parameterDefaultValue: obj ->
      ProvidedStaticParameter
    member AddXmlDoc: xmlDoc: string -> unit
    member AddXmlDocDelayed: xmlDocFunction: (unit -> string) -> unit
```

Статические параметры

```
/// Represents a provided static parameter.
[<Class>]
type ProvidedStaticParameter =
    inherit ParameterInfo
    new: parameterName: string * parameterType: Type * ?parameterDefaultValue: obj ->
      ProvidedStaticParameter
    member AddXmlDoc: xmlDoc: string -> unit
    member AddXmlDocDelayed: xmlDocFunction: (unit -> string) -> unit
```

Статические параметры

```
/// Represents a provided static parameter.
[<Class>]
type ProvidedStaticParameter =
    inherit ParameterInfo
    new: parameterName: string * parameterType: Type * ?parameterDefaultValue: obj ->
      ProvidedStaticParameter
    member AddXmlDoc: xmlDoc: string -> unit
    member AddXmlDocDelayed: xmlDocFunction: (unit -> string) -> unit
```

Статические параметры

```
/// Represents a provided static parameter.
[<Class>]
type ProvidedStaticParameter =
    inherit ParameterInfo
    new: parameterName: string * parameterType: Type * ?parameterDefaultValue: obj ->
      ProvidedStaticParameter
    member AddXmlDoc: xmlDoc: string -> unit
    member AddXmlDocDelayed: xmlDocFunction: (unit -> string) -> unit
```

Подклассы и параметры

```
let ty = ProvidedTypeDefinition(assy, ns, typeName, None)

makeProvidedConstructor
    [ ]
    (fun [] -> <@@ fileName |> ID3Reader.readID3Frames @@>)
|> addDelayedXmlComment "Creates a reader for the specified file."
|> ty.AddMember
```

Подклассы и параметры

```
let ty = ProvidedTypeDefinition(assy, ns, typeName, None)

makeProvidedConstructor
    [ ]
    (fun [] -> <@@ fileName |> ID3Reader.readID3Frames @@>)
|> addDelayedXmlComment "Creates a reader for the specified file."
|> ty.AddMember

let inline makeProvidedConstructor parameters invokeCode =
  ProvidedConstructor(parameters, InvokeCode = invokeCode)
```

Подклассы и параметры

```
let ty = ProvidedTypeDefinition(assy, ns, typeName, None)

makeProvidedConstructor
    [ ]
    (fun [] -> <@@ fileName |> ID3Reader.readID3Frames @@>)
|> addDelayedXmlComment "Creates a reader for the specified file."
|> ty.AddMember

let inline makeProvidedConstructor parameters invokeCode =
  ProvidedConstructor(parameters, InvokeCode = invokeCode)

"AttachedPicture"
|> makeReadOnlyProvidedProperty<AttachedPicture> (buildExpr tag)
|> addDelayedXmlComment "Gets the album art attached to the file.
  Corresponds to the APIC tag.")
```

Подклассы и параметры

```
/// Represents an erased provided constructor.
[<Class>]
type ProvidedConstructor =
    inherit ConstructorInfo

    new: parameters: ProvidedParameter list * invokeCode: (Expr list -> Expr) ->
      ProvidedConstructor
```

Подклассы и параметры

```
[<Class>]
type ProvidedProperty =
    inherit PropertyInfo
    new:
        propertyName: string
        * propertyType: Type
        * ?getterCode: (Expr list -> Expr)
        * ?setterCode: (Expr list -> Expr)
        * ?isStatic: bool
        * ?indexParameters: ProvidedParameter list -> ProvidedProperty
    member AddObsoleteAttribute: message: string * ?isError: bool -> unit
    member AddXmlDoc: xmlDoc: string -> unit
    member AddXmlDocDelayed: xmlDocFunction: (unit -> string) -> unit
    member AddXmlDocComputed: xmlDocFunction: (unit -> string) -> unit
    member IsStatic: bool
    member AddDefinitionLocation: line:int * column:int * filePath:string -> unit
    member AddCustomAttribute: CustomAttributeData -> unit
```

Подклассы и параметры

```
[<Class>]
type ProvidedProperty =
    inherit PropertyInfo
    new:
        propertyName: string
        * propertyType: Type
        * ?getterCode: (Expr list -> Expr)
        * ?setterCode: (Expr list -> Expr)
        * ?isStatic: bool
        * ?indexParameters: ProvidedParameter list -> ProvidedProperty
    member AddObsoleteAttribute: message: string * ?isError: bool -> unit
    member AddXmlDoc: xmlDoc: string -> unit
    member AddXmlDocDelayed: xmlDocFunction: (unit -> string) -> unit
    member AddXmlDocComputed: xmlDocFunction: (unit -> string) -> unit
    member IsStatic: bool
    member AddDefinitionLocation: line:int * column:int * filePath:string -> unit
    member AddCustomAttribute: CustomAttributeData -> unit
```

Добавление всего сгенеренного

```
let ty = ProvidedTypeDefinition(assy, ns, typeName, None)

makeProvidedConstructor
    [ ]
    (fun [] -> <@@ fileName |> ID3Reader.readID3Frames @@>)
|> addDelayedXmlComment "Creates a reader for the specified file."
|> ty.AddMember
```

Добавление всего сгенеренного

```
let ty = ProvidedTypeDefinition(assy, ns, typeName, None)

makeProvidedConstructor
    [ ]
    (fun [] -> <@@ fileName |> ID3Reader.readID3Frames @@>)
|> addDelayedXmlComment "Creates a reader for the specified file."
|> ty.AddMember
```

Добавление всего сгенеренного

```
member this.AddMembers(memberInfos:list<#MemberInfo>) =
    memberInfos |> List.iter this.PatchDeclaringTypeOfMember
    membersQueue.Add (fun () -> memberInfos |> List.toArray |>
      Array.map (fun x -> x :> MemberInfo ))
```

Добавление всего сгенеренного

```
member this.AddMembers(memberInfos:list<#MemberInfo>) =
    memberInfos |> List.iter this.PatchDeclaringTypeOfMember
    membersQueue.Add (fun () -> memberInfos |> List.toArray |>
      Array.map (fun x -> x :> MemberInfo ))
```

Добавление всего сгенеренного

```
let elems = membersQueue |> Seq.toArray
membersQueue.Clear()
members.Add m
```

Добавление всего сгенеренного

```
let elems = membersQueue |> Seq.toArray
membersQueue.Clear()
members.Add m
```

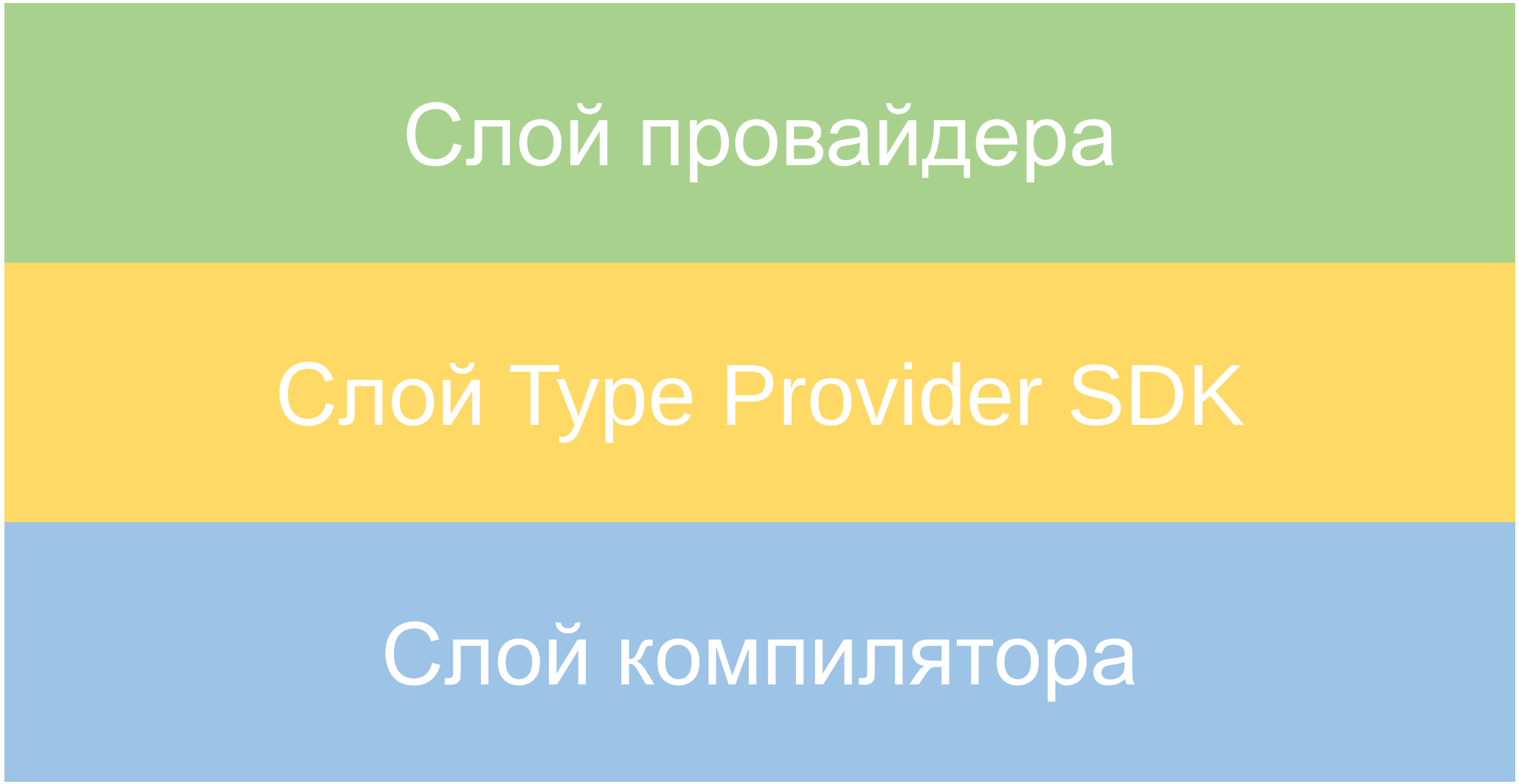
Добавление всего сгенеренного

```
match m with
| :? ProvidedProperty    as p ->
    if not p.BelongsToTargetModel then
        if p.CanRead then members.Add (p.GetGetMethod true)
        if p.CanWrite then members.Add (p.GetSetMethod true)
| :? ProvidedEvent       as e ->
    if not e.BelongsToTargetModel then
        members.Add (e.GetAddMethod true)
        members.Add (e.GetRemoveMethod true)
| _ -> ()
```

Слой провайдера

Слой провайдера

Слой Type Provider SDK


Слой провайдера
Слой Type Provider SDK
Слой компилятора

Спасибо!


nevoroman@gmail.com


nevoroman