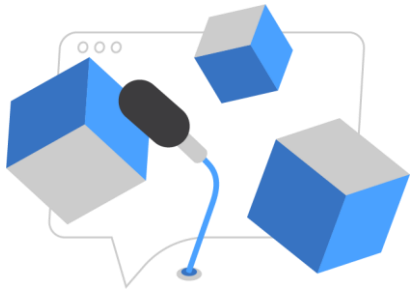

DOTVVM
VIRTUAL CONFERENCE
JUNE 21-22

New features in data-binding

Andrej Čižmárik

Update NOW 2021

riganti

Lambda functions

```
(input-parameters) => expression
```

- Lambda functions must be assignable to a known delegate type
- Supported delegates: **Action**s, **Func**s, **Predicate**s

Type inference in lambda functions (1)

- In many cases, DotVVM can **infer lambda parameter types** from context they are used in

- `Customers.Where((Customer c) => c.Id > 0)`
  - DotVVM 3.0 without type-inference

- `Customers.Where(c => c.Id > 0)`
  - DotVVM 3.1 with type inference

DOTVVM

Type inference in lambda functions (2)

- **What is our context?**
  - **Called method**: `Enumerable.Where(IEnumerable<T>, Func<T,bool>)`
  - **Target expression**: `List<Customer> Customers { get; set; }`

- **`T` is a `Customer`!**
- We have successfully inferred type of parameter named `` `c` ``

- `Customers.Where(c => c.Id > 0)`
  - DotVVM 3.1 with type inference

DOTVVM

Type inference in lambda functions (3)

- ```
  <dot:NamedCommand
          Name=MyCommand
          Command="{staticCommand: arg => DoSomething(arg)}" />
  ```
- **Error**: Could not infer type of parameter **arg**. (DotvvmCompilationException)

- Type inference sometimes does not have enough information to infer types
- DotVVM currently supports a subset of general C# type inference capabilities

DOTVVM
  ```

Client-side collections filtering

Client-side filtering (1)

- Evaluate expressions in **value bindings**
- Use **JavaScript translations** and **lambda functions** to filter within a value-binding

```
• {value: Customers.Where(c => c.Id > 0)}
• {value: Customers.OrderBy(c => c.DateJoined)}
```

- What methods can we use in DotVVM?

Client-side filtering (2): LINQ methods

- We introduced a lot of new **JS translations**
- The goal is to **capture .NET's behavior** as close as possible. Although some differences are possible

- Supported methods: **`All`, `Any`, `Concat`, `Distinct`, `FirstOrDefault`, `LastOrDefault`, `Max`, `Min`, `OrderBy`, `OrderByDescending`, `Select`, `Skip`, `Take`, `Where`**

DEMO
(client-side filtering)

New features for collections

Support for List<T>

- Supported methods: **`Add`**, **`AddRange`**, **`Clear`**, **`Insert`**, **`InsertRange`**, **`RemoveAt`**, **`RemoveAll`**, **`RemoveRange`**, **`Reverse`**

- DotVVM custom extensions:
  - **`AddOrUpdate`**`(element, matcher, updater)`
  - **`RemoveFirst`**`(predicate)`
  - **`RemoveLast`**`(predicate)`

DOTVVM

Support for Dictionary<K,V>

- Dictionaries are now supported in viewModels

- **Indexer** is supported – both **`get`** and **`set`** methods
- Other supported methods:
  - **`.Clear`**`()`
  - **`.ContainsKey`**`(key)`
  - **`.Remove`**`(key)`

DOTVVM

DEMO
(lists and dictionaries)

Variables

Declaring variables

- Variables in bindings are of **S**tatic **S**ingle **A**ssignment form (**SSA**), which implies following rules:
  - Declare before first use
  - Variables are only assigned once
- Declare using **var** keyword (`var variable = expr`)

- Note: variables are **statically-typed** and we use compiler to infer the correct type of given expression

Using variables

DotHTML:

```
{staticCommand:
    var result = GetNumber();
    Prop1 = result;
    Prop2 = result;
}
```

ViewModel:

```
public int Prop1 { get;set; }
public int Prop2 { get;set; }

public int GetNumber()
    => 42;
```

DOTVVM

Working with strings

String interpolation

- DotVVM supports interpolated **expressions** with **formatting component** (optional)
- Syntax sugar for string's **Format method**
- See supported formats in documentation

String interpolation

- DotVVM supports interpolated **expressions** with **formatting component** (optional)
- Syntax sugar for string's **Format method**
- See supported formats in documentation

```
•  {value: $"Hello {NameProperty}!"}
•  {value: $"Today's formatted date: {DateTimeProperty:dd/MM}"}
•  {value: $"Complex expression: {Property ?? "Unknown"}"}
```

Support for string methods

- Apart from existing support, DotVVM 3.1 introduces more JavaScript translations for string methods

- Supported methods: **`Contains`**, **`EndsWith`**, **`IndexOf`**, **`IsNullOrEmpty`**, **`Join`**, **`LastIndexOf`**, **`Replace`**, **`Split`**, **`StartsWith`**, **`ToUpper`**, **`ToLower`**, **`ToUpperInvariant`**, **`ToLowerInvariant`**

Other binding improvements

Extension methods

- Use **@import** directive to specify what namespaces should be searched for extension methods
- Compiler prefers non-extension methods

- **`System.Linq`** was added to default imports alongside with namespace that provides DotVVM extensions
  - Use @import directive for your own namespaces

More method types supported

- Variable number of arguments for methods
  - DotVVM supports methods with **params** keyword
- Improved support for **generic methods**
  - DotVVM improved its resolving and matching routines that were used for compiling **method invocations** and **registering JS translations**

Support for Math methods

- Supported methods: **`Abs`, `Acos`, `Asin`, `Atan`, `Atan2`, `Ceiling`, `Cos`, `Cosh`, `Exp`, `Floor`, `Log`, `Log10`, `Max`, `Min`, `Pow`, `Round`, `Sign`, `Sin`, `Sinh`, `Sqrt`, `Tan`, `Tanh`, `Trunc`**

Conclusion: do not forget to upgrade ☺

- Lambda functions (3.0) and type inference (**3.1**)
- Enumerable method translations (**3.1**)
- List method translations (**3.1**)
- Dictionary support (**3.1**) and method translations (**3.1**)
- Variables (3.0)
- String interpolation (**3.1**) and method translations (**3.1**)
- Params support (3.0) and generics improvements (**3.1**)
- Extension methods (3.0 – **3.1**)
- Math method translations (**3.1**)