Developer Week 2013

@developer_week #dwx13

Granuläres .NET-Web-Development

...mit Nancy- & Simple.Data-Frameworks

Timothée Bourguignon

MATHEMA Software GmbH

DWX

@developer_week #dwx13

First Date with Nancy

„Lightweight Web Framework for .NET“

nancyfx.org | #nancyfx

Microframework

- Serve up web content

- Lean API
- Extensible API
- Simple setup

- “Close to the metal”

What has Nancy to offer?

- Simplicity & Readability
- Modularity
- OpenSource
- "Close" to HTTP
- Very explicit routing
- Runs anywhere
- „Super Duper Happy Path“

„Hello Nancy“

```
public class MyModule : NancyModule
{
        public MyModule()
        {
                Get["/"] = _ => "Hello World";
        }
}
```

Route

- Composed of
  - Method (HTTP Methods: Get, Post, Put…)
  - Pattern
  - Action (+parameters & result object)
  - Condition (optional)

```
Get["/voteup/{id}"] = x => {
 return View["Voteup", x.id];
};
```

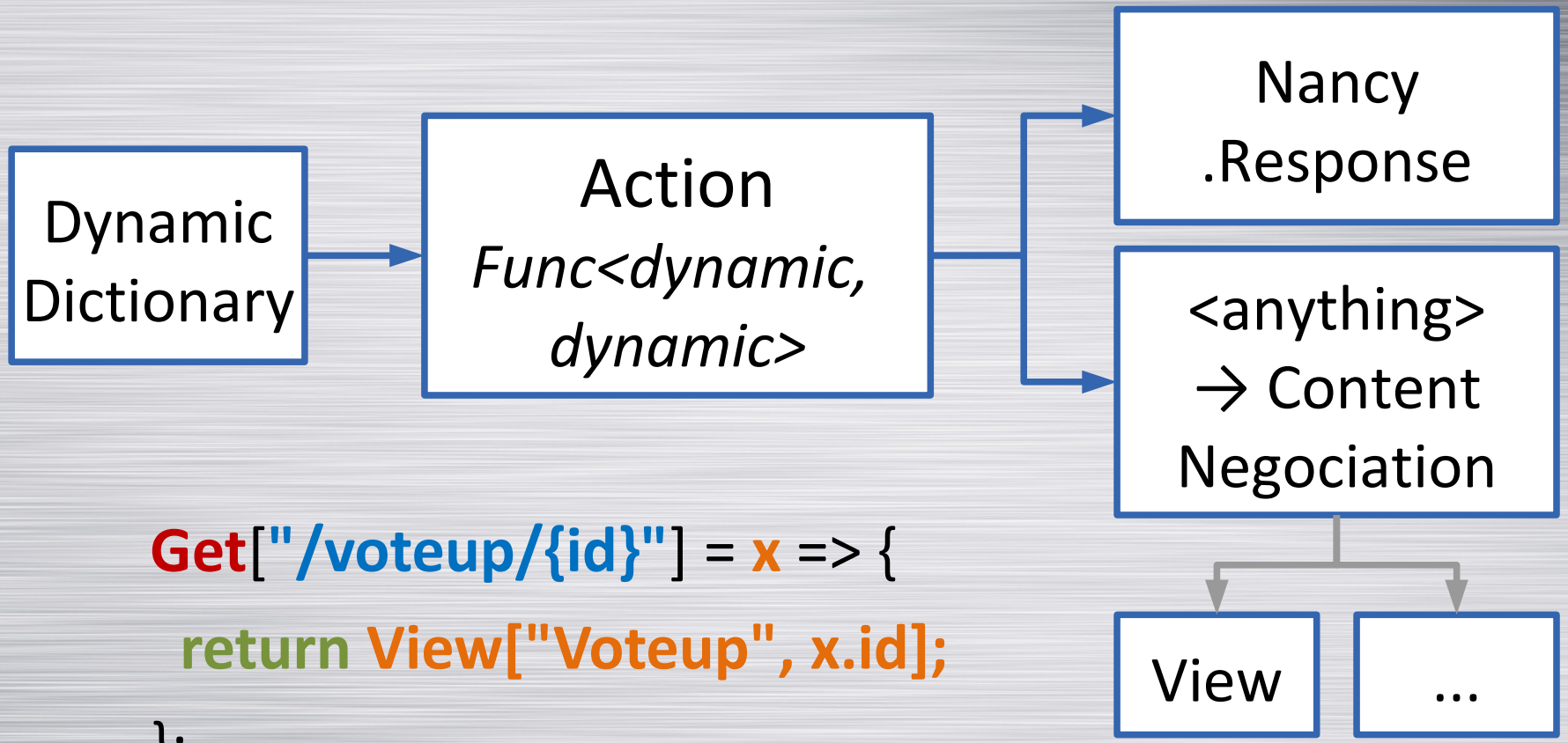
Pattern

- Literal segments: "/customer"
- Capture segments: "/{id}“
- Capture segments (Optional): "/{id?}“
- Capture segments (Optional/Default): "{name?unnamed}“
- Regex Segments: /(?<id>[\d]+)
- Greedy Regex Segments: ^(?<name>[a-z]{3,10}(?:/{1}))$
- Greedy Segments: "/{id*}“

```
Get["/voteup/{id}"] = x => {
  return View["Voteup", x.id];
};
```

Action

- Behavior invoked by a route




```
Get["/voteup/{id}"] = x => {
  return View["Voteup", x.id];
};
```

Nancy.Response

- Nancy.Response implicit casts
  - Int32 → HTTP Status Code
  - String → body of the response

- Response formatters:
  - As File, Image, Json, Xml & Redirect

Serving up Views

- Supported View Engines
  - SuperSimpleViewEngine
  - Razor, Spark, DotLiquid...
  - … any other IViewEngine
- View Engine is selected dynamically, based on the view's file extension
- Views are also discovered

```
Get["/products"] = _ => {
   return View["products.cshtml"];
};
```

Model Binding

- Module → View
  - Any object Type
  - dynamics per default
- View → Module
  - Query string
  - Captured parameters
  - Body of a request

```
Get["/products"] = _ =>
{
   return View["products",
                someModel];
};
```

```
Foo foo = this.Bind();
var foo = this.Bind<Foo>();
this.BindTo(foo);
```

DWX

@developer_week #dwx13

Hands on Nancy

HackerNews meet Nancy

Simple.Data

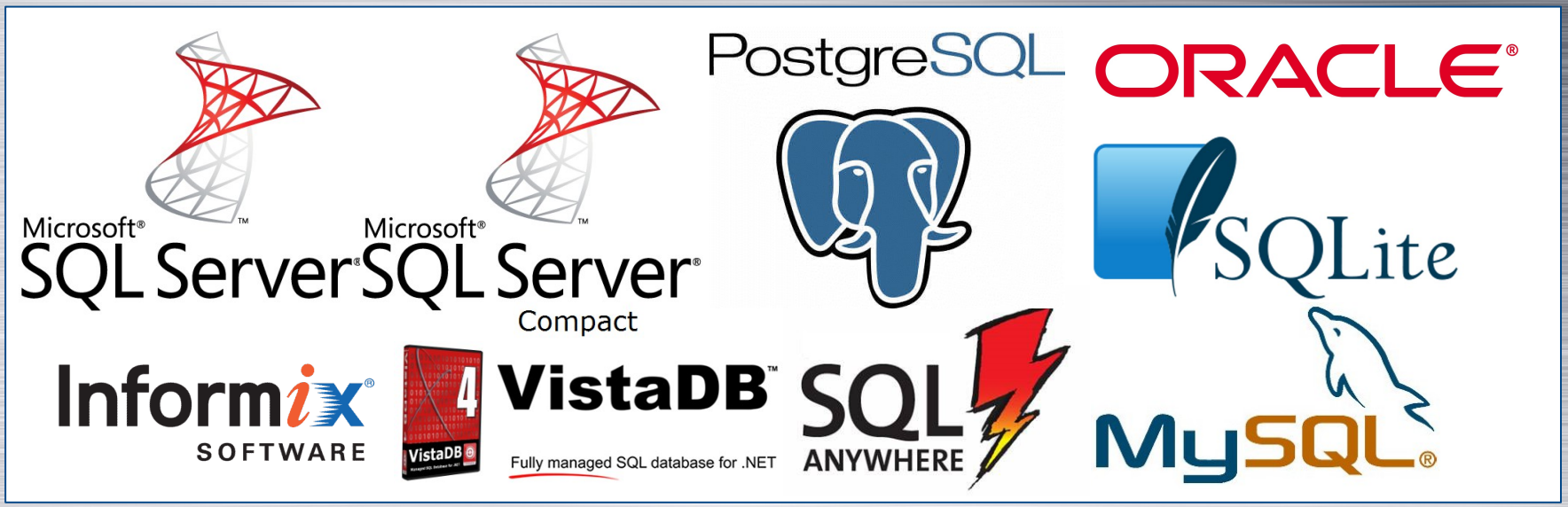
...an O/RM without O, R or M

What is Simple.Data?

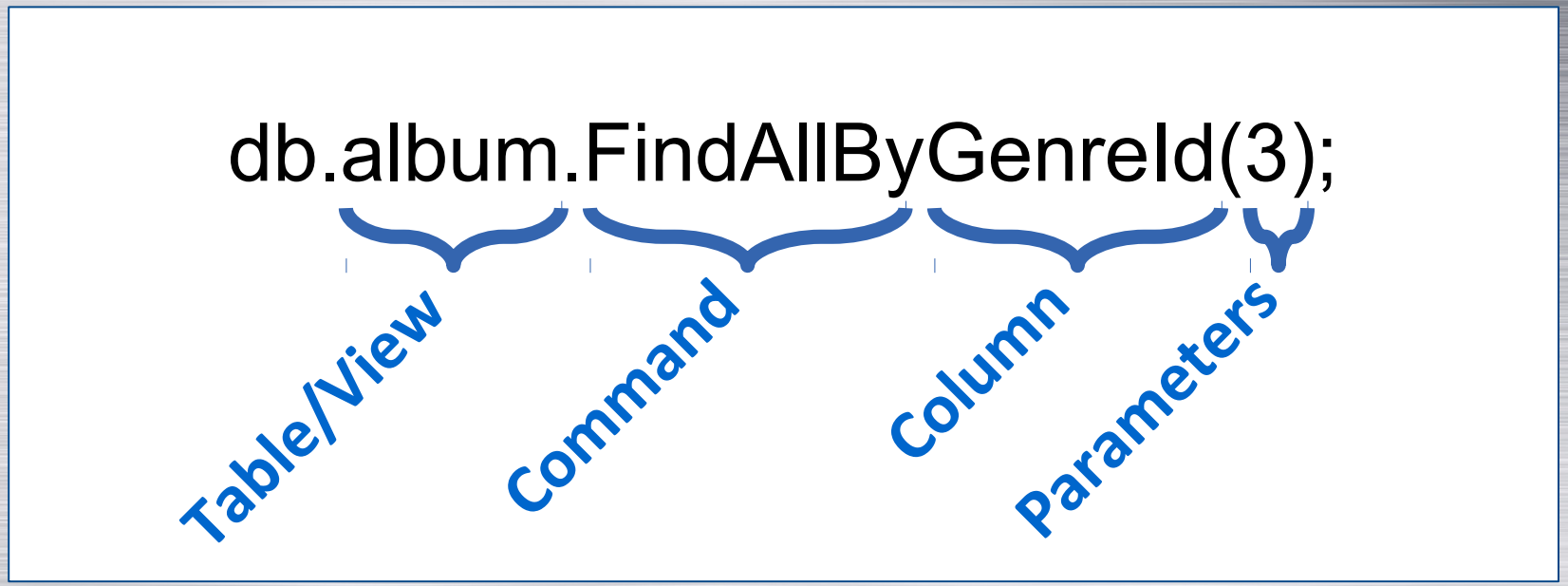
- Lightweight way of manipulating data
  - Based on .NET 4.0's „dynamic“ keyword
  - Interprets method and property names
  - Maps them to your underlying data-store
  - Prevents SQL Injection
  - Inspired by Ruby’s ActiveRecord and DataMappers
  - Open Source & runs on Mono
  - V1.0 rc3 released in Nov. 2012

```
PM>  Install-package Simple.Data.<TheProvider>
```

Database agnostic



Fluid Convention



„Hello Simple.Data“

```
public void GetCustomers()
{
        var conString = "...";
        dynamic db =
                Database.OpenConnection(conString);

        dynamic customer = db.Customers.FindById(1);

        Console.WriteLine("{0}, {1}!",
                customer.FirstName, customer.LastName);
}
```

Simple CRUD operations

```
public void CRUD()
{
        db.People.FindAllByName("Bob");
        db.People.FindByFirstNameAndLastName("Bob", "X");
        db.Users.All().OrderByJoinDateDescending();
        db.Users.All().OrderByJoinDate().ThenByNickname();
        db.People.Insert(Id: 1, FirstName: "Bob");
        db.People.Insert(new Person {Id = 1, Name = "Bob"});
        db.People.UpdateById(Id: 1, FirstName: "Robert");
        db.People.DeleteById(1);
}
```

Barely less simple operations

```
//Paging
db.Users.All().OrderByNickname().Skip(10).Take(10);

//Table joins
db.Customers.FindByCustomerId(1).WithOrders();

//Casting
Artist artist = db.Artists.Get(42);
IList<Artist> artists = db.Artists.All().ToList<Artist>();
```

DWX

@developer_week #dwx13

Nancy, meet Simple.Data

Simple.Data, meet Nancy

DWX

@developer_week #dwx13

Second date with Nancy

In case the SuperDuperHappyPath is not completely Super, Duper or Happy yet…

Bootstrapper

- ~DSL on top of the IoC container
  - Ships with TinyIoC
- Responsible for „putting the puzzle together“
- Override and extend
- Autoregister your dependencies

```
public class Home : NancyModule
{
        public Home(IMessageService service)
        {
                //If there is only one implementation
                // of ImessageService, TinyIoC will
                // resolve it and inject it
        }
}
```

Content Negociation

- Actions output:
  - ResponseObject or ContentNegociation
    - Response.AsXml(), Response.AsJson()...
- Negociating the format based on the:
  - URI Extension: .json, .xml …
  - Header information: „*Accept: application/xml*“
- Default ResponseProcessors: View, Xml, Json...

```
return Negotiate.WithModel(model)
                .WithView("MyView");
```

Authentication

```
public class MyBootstrapper : DefaultNancyBootstrapper
{
    protected override void InitialiseInternal(TinyIoCContainer container)
    {
        base.InitialiseInternal(container);
        FormsAuthentication.Enable(this,
            new FormsAuthenticationConfiguration
            {
                RedirectUrl = "~/login",
                UsernameMapper = container.Resolve<IUsernameMapper>()
            });
    }
}
public class MyModule : NancyModule
{
    public MyModule() : base("/secure")
    {
        this.RequiresAuthentication();
        Get["/"] = _ => "Secure!";
    }
}
```

Testing

```
[Test]
public void Should_redirect_to_login_with_error_details_incorrect()
{
        // Given
        var bootstrapper = new DefaultNancyBootstrapper();
        var browser = new Browser(bootstrapper);

        // When
        var response = browser.Post("/login/", (with) =>
        {
                with.HttpRequest();
                with.FormValue("Username", "username");
                with.FormValue("Password", "wrongpassword");
        });

        // Then
        response.ShouldHaveRedirectedTo(
                "/login?error=true&username=username");
}
```

```
PM>  Install-package Nancy.Testing
```

Nancy.Diagnostic

- localhost/_nancy
  - Information
  - Interactive diagnostic
  - Request Tracing
  - Settings

Version	v0.11.0.0
Caches Disabled	true
Traces Disabled	false
Case Sensitivity	Insensitive
Root Path	G:\WORKSPACE\.NetHN\.NetHN\
Hosting	Aspnet (v0.11.0.0)
Bootstrapper Container	TinyIoC
Located Bootstrapper	NetHN.FormsAuthBootstrapper
Loaded View Engines	SuperSimple
	Razor
Route Resolver	Nancy.Routing.DefaultRouteResolver
Route Pattern Matcher	Nancy.Routing.DefaultRoutePatternMatcher
Context Factory	Nancy.DefaultNancyContextFactory
Nancy Engine	Nancy.NancyEngine
Module Key Generator	Nancy.Bootstrapper.DefaultModuleKeyGenerator
Route Cache	Nancy.Routing.RouteCache
Route Cache Provider	Nancy.Routing.DefaultRouteCacheProvider
View Locator	Nancy.ViewEngines.DefaultViewLocator
...	

DWX

@developer_week #dwx13

.NetHackerNews on Steroids

Simple.Data & Nancy hand in hand

Wrap-up!

- Simple, readable & flexible frameworks
- Run everywhere

- Nancy
  - Easy to test
  - Customisable
  - Great for Webservices
- Simple.Data
  - Powerful
  - DB Agnostic
  - Compelling

- Excellent for prototypes & small projects
- „Super duper happy path“

Further Reading

- Github
  - https://github.com/markrendle/Simple.Data
  - https://github.com/NancyFx/Nancy
- GoogleGroups
  - https://groups.google.com/forum/?fromgroups=#!forum/simpledata
  - https://groups.google.com/forum/?fromgroups=#!forum/nancy-web-framework

Contacts

- Andreas Håkansson (NancyFx)
  - @TheCodeJunkie

- Steven Robbins (NancyFx, TinyIoC)
  - @Grumpydev
  - http://www.grumpydev.com/

- Mark Rendle (Simple.Data)
  - @MarkRendle
  - http://blog.markrendle.net/

DWX

@developer_week #dwx13

Fragen?

MATHEMA

tim.bourguignon@mathema.de
about.me/timbourguignon

DWX Developer Week 2013

Feedback & Kontakt: feedback@developer-week.de

Durchhänger?

Eine kleine Stärkung gibt es jetzt am MATHEMA-Stand!