

3.– 6. September 2012
in Nürnberg



Herbstcampus

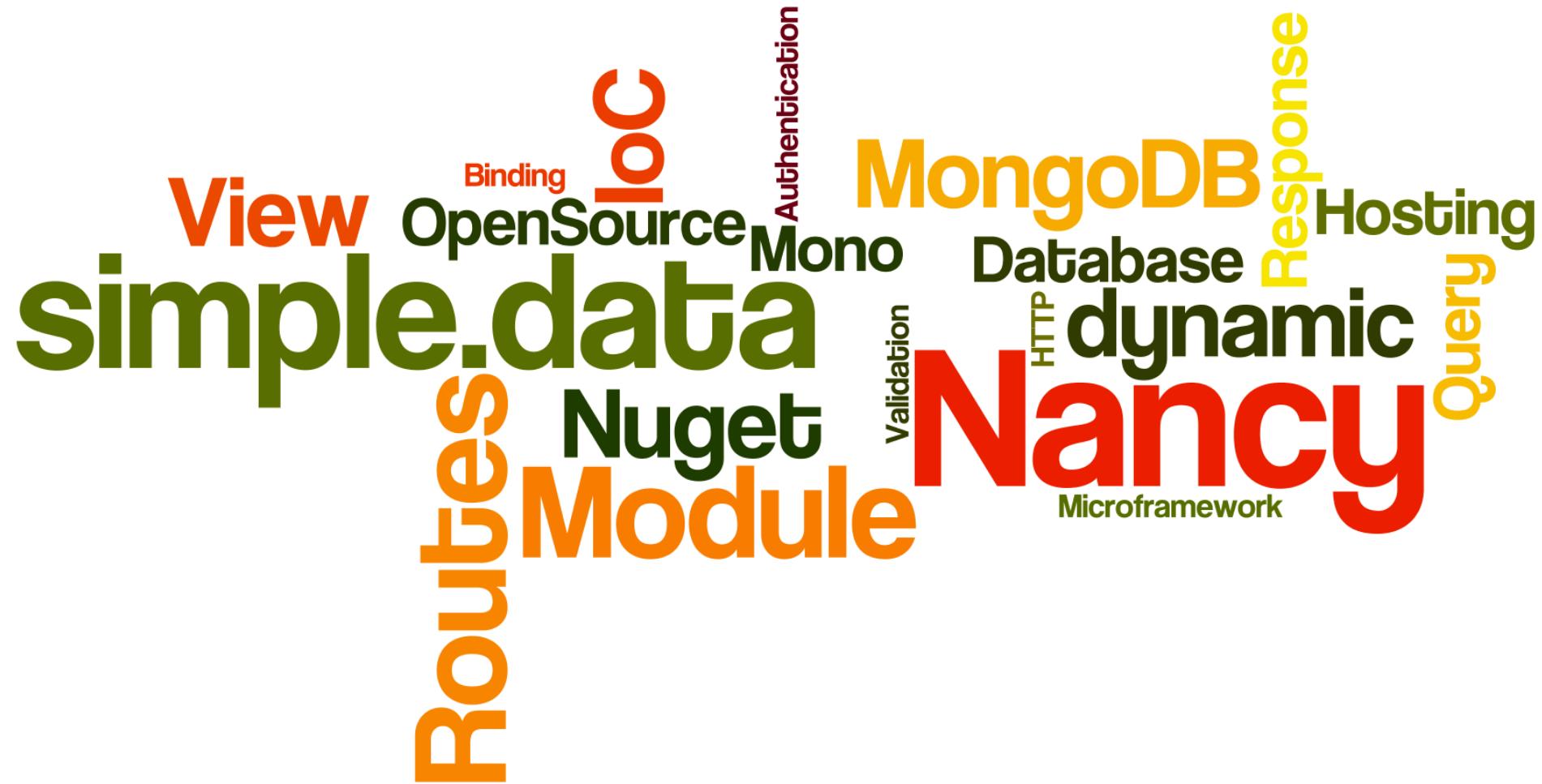
Wissenstransfer
par excellence

EINFACH SIMPEL

Granuläres .NET-Web-Development mit Nancy- und Simple.Data- Frameworks

Timothée Bourguignon

Mathema Software GmbH



Why?

- Simplicity & Readability
- Modularity
- OpenSource

- "Close" to HTTP
- Very explicit routing
- Runs anywhere

- „Super Duper Happy Path“

3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

First date with Nancy

„Lightweight Web Framework for .NET“

nancyfx.org | [#nancyfx](https://twitter.com/nancyfx)

Microframework

- Serve up web content
- Lean
- Extensible API
- Simple setup
- “Close to the metal”

„Hello Sinatra“

```
$ gem install sinatra  
$ ruby -rubygems app.rb
```

```
require 'sinatra'  
get '/hi' do  
  "Hello World!"  
end
```



„Hello Nancy“

```
namespace NancyDemo
{
    public class MainModule : NancyModule
    {
        public MainModule()
        {
            Get["/hi"] = _ => "Hello Nancy!";
        }
    }
}
```



NANCY

Setting up a Nancy project

- Create an empty web project
- Install Nancy via Nuget:

```
PM> Install-package Nancy.Hosting.Aspnet  
Or      Install-Package Nancy.Hosting.Owin  
Or      Install-Package Nancy.Hosting.Wcf
```

...

- Adds Nancy assembly references
 - Sets up web.config (if using IIS)
- Create a module and define a route



NANCY

Architectural Overview

- HTTP handler
 - Modules
 - Routes
 - Response
 - Views
 - View engines
- Bootstrapping (IoC)
 - Hosting
 - ASP.NET
 - WCF
 - Azure
 - Umbraco
 - Owin (~ Ruby's "Rack")
 - Self-hosted
 - Authentication
 - ...



NANCY

Module's Anatomy

```
public class MyModule: NancyModule
{
    Public MyModule()
    {
        Get[ "/" ] = _ => { ... };
        Get[ "/submit" ] = _ => { ... };
        Post[ "/submit" ] = _ => { ... };
        Get[ "/voteup/{id}" ] = parameters => { ... };
        Get[ "/login" ] = _ => { ... };
        Get[ "/logout" ] = _ => { ... };
        Post[ "/login" ] = _ => { ... };
    }
}
```



NANCY

Route

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

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



NANCY

Route pattern

- Literal segments: `"/voteup"`
- Variable segments: `"/voteup/{id}"`
- Literal segments + regular expression back-references: `"/voteup/(?<id>[\d]+)"`

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



NANCY

Route resolver

- Request method & Path matching
- Picks the first route among those which captures the most parameters

Lifecycle

- "Before" pipeline is executed (if present)
- Route's action is invoked
- "After" pipeline is executed (if present)

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



NANCY

Route action, parameters & Response

- Receives a .Net4.0 “*dynamic*” object packaging the request parameters

`x => x.id`

- Returns an “*Nancy.Response*” object

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



NANCY

Nancy.Response

- *Nancy.Response* implicit casts
 - Int32 → HTTP Status Code (ex: 404, No Found)
 - 'HttpStatusCode' enumerable value
 - String → body of the response
 - 'Action<Stream>'
- Response formatters:
 - As File, Image, Json, Xml & Redirect
- Views

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



Serving up views

- Views are discovered in the project
- Supported View Engines
 - SuperSimpleViewEngine (basic HTML & iteration syntax)
 - Razor
 - Spark
 - Django
 - DotLiquid
 - Any other implementation IViewEngine
 - Selected dynamically, based on the view's file extension



NANCY

View model & Model Binding

- Data passed via Query string, captured parameters on routes or body of a request
- Module → View
 - Supports any object Type
 - Uses “dynamic” per default
- Module ← View
 - Model binding for all ways

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



3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Demo: hands on Nancy

HackerNews meet Nancy

3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Simple.Data

A „/“, e.g. an O/RM without O, R or M

Simple.Data

- Inspired by Ruby's DataMapper & ActiveRecord
- Based on the .NET 4.0 “dynamic” keyword

- No SQL injection

- Easy
- Intuitive
- Flexible

- Database agnostic
- Convention over Configuration
- Not an O/RM

Hello Simple.Data

```
public void Greetings()  
{  
    var db = Simple.Data.Database.Open();  
  
    var hello = db.Hello.FindById(1);  
  
    Console.WriteLine("{0}, {1}!",  
        hello.Greeting, hello.Subject);  
}
```

Setting up a Simple.Data project

- Install the driver you need via Nuget

```
PM> Install-Package Simple.Data.MongoDB
```

- Create a “dynamic” Database object



NANCY

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

- Complex criteria

```
db.Customers.Find(db.Customers.MoneyOwing > 0);
```

- Paging

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

- Having

```
db.Posts.All().Having(db.Posts.Comments.CommentId  
                        .Count() == 0);
```

- Upsert (Update or Insert)

```
db.Users.Upsert(user);
```


Complex operations

- Eager loading with the “with” operator

```
db.Customers.WithOrders().Get(1);
```

```
db.Customers.FindAllById(1).WithOrders();
```

- Implicit / Explicit Join, OuterJoin...

Drivers & Supported DBs

- ADO-based access to relational databases:
 - SQL Server 2005 and later (including SQL Azure)
 - SQL Server Compact Edition 4.0
 - Oracle
 - MySQL 4.0 and later
 - SQLite
 - PostgreSQL
 - SQLAnywhere
 - Informix
- MongoDB
- OData

3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Demo: Nancy, meet Simple.Data

Simple.Data, meet Nancy

3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Second date with Nancy

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

Nancy's Bootstrapping (IoC)

- Uses IoC containers to bootstrap the framework
- Discover and load any implementation from `INancyBootstrapper`
- The `DefaultNancyBootstrapper` uses `TinyIOC`
- Nancy also includes bootstrappers for
 - Ninject
 - StructureMap
 - Unity
 - Windsor



Authentication

- Nancy.Authentication.Basic
 - Basic HTTP authentication
- Nancy.Authentication.Forms
 - Proper authentication
 - Facebook, OAuth etc. on their way

Authentication

```
public class MyBootstrapper : DefaultNancyBootstrapper
{
    protected override void InitialiseInternal(TinyIoC.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

- Nuget: PM> Install-Package Nancy.Testing

- „Browser“ class

```
var result = browser.Get("/", with => {  
    with.HttpRequest(); });
```

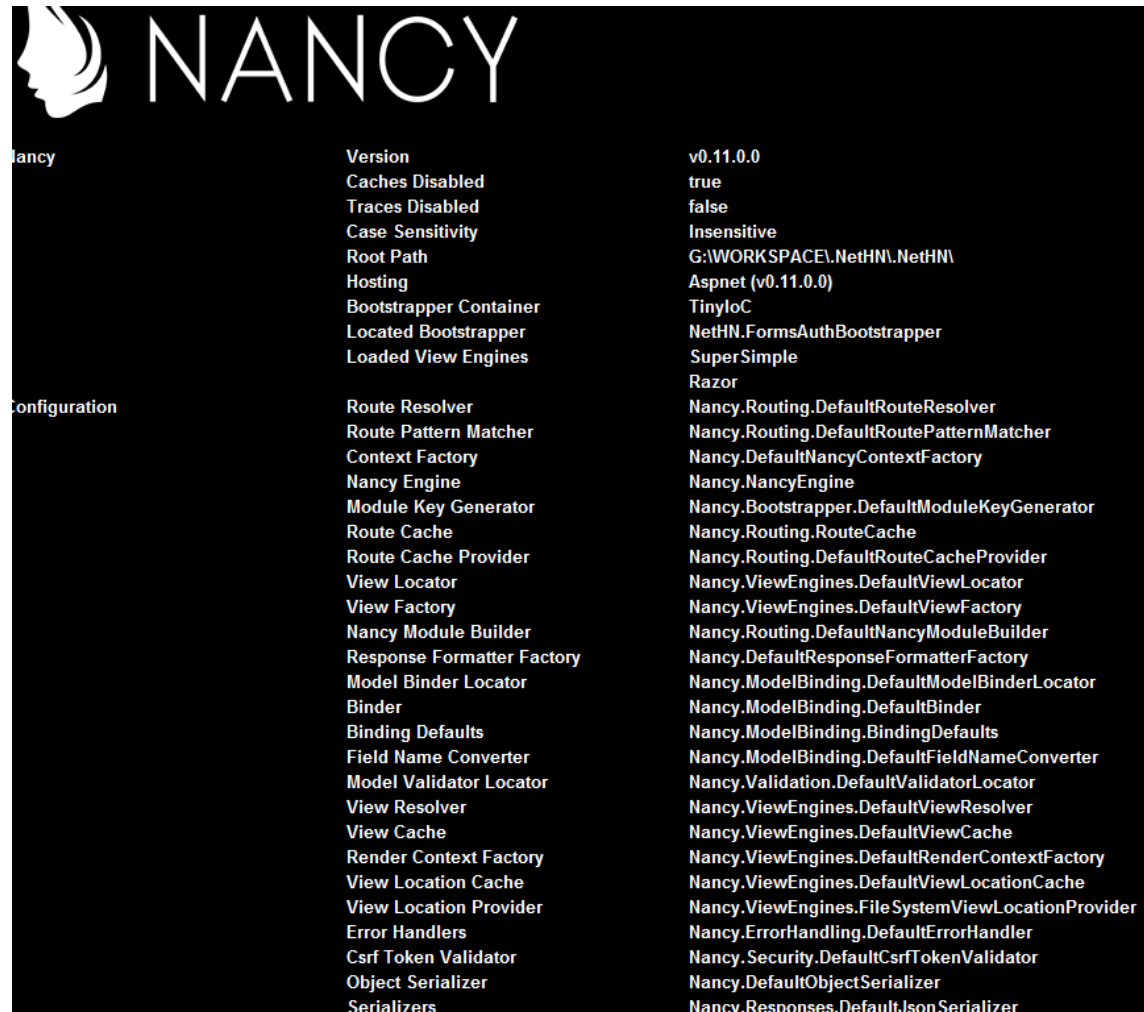
- Assert

```
response.Body["#errorBox"]  
    .ShouldExistOnce()  
    .And.ShouldBeOfClass("floatingError")  
    .And.ShouldContain("invalid",  
        StringComparison.InvariantCultureIgnoreCase);
```


Nancy.Diagnostic

localhost/_nancy

- Information
- Interactive diagnostic
- Request Tracing
- Settings



The screenshot shows the Nancy Diagnostic tool interface. At the top, there is a logo of a woman's profile and the word "NANCY" in large white letters. Below this, the interface is divided into sections. The "nancy" section lists general settings, and the "configuration" section lists various components and their default values.

Section	Property	Value
nancy	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
	configuration	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
View Factory		Nancy.ViewEngines.DefaultViewFactory
Nancy Module Builder		Nancy.Routing.DefaultNancyModuleBuilder
Response Formatter Factory		Nancy.DefaultResponseFormatterFactory
Model Binder Locator		Nancy.ModelBinding.DefaultModelBinderLocator
Binder		Nancy.ModelBinding.DefaultBinder
Binding Defaults		Nancy.ModelBinding.BindingDefaults
Field Name Converter		Nancy.ModelBinding.DefaultFieldNameConverter
Model Validator Locator		Nancy.Validation.DefaultValidatorLocator
View Resolver		Nancy.ViewEngines.DefaultViewResolver
View Cache		Nancy.ViewEngines.DefaultViewCache
Render Context Factory		Nancy.ViewEngines.DefaultRenderContextFactory
View Location Cache		Nancy.ViewEngines.DefaultViewLocationCache
View Location Provider		Nancy.ViewEngines.FileSystemViewLocationProvider
Error Handlers		Nancy.ErrorHandling.DefaultErrorHandler
Csrf Token Validator		Nancy.Security.DefaultCsrfTokenValidator
Object Serializer		Nancy.DefaultObjectSerializer
Serializers		Nancy.Responses.DefaultJsonSerializer

My opinion

- Pros
 - It is incredibly simple, readable & flexible
 - It runs everywhere (Self hosted on Mobile?)
 - Great for Webservice or small / fast projects
- Cons
 - Hard to picture a large project with Nancy or Simple.Data
 - Flexibility via „*dynamic*“ can be double edged
 - Nancy & Simple.Data cannot do more or less than the others
... but they do it in a very elegant and efficient manner

Links & Contacts



- **Andreas Håkansson (NancyFx)**
 - @TheCodeJunkie
 - <http://elegantcode.com/>



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



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

3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Tim Bourguignon

about.me/timbourguignon

Additional sources

- „Simple.Data, .NET Database access made easier“, Mark Rendle, <http://www.slideshare.net/markrendle/simple-data>
- Nancy (.NET Micro Web Frameworks), Nicholas Cloud, <http://www.nicholascloud.com/2011/05/nancy-net-micro-web-frameworks-part-1/>