# F# FÜR DAS WEB

CARSTEN KÖNIG

GITHUB

# F# FÜR DAS WEB

# WEBSHARPER

- Webapplikationen komplett in F#
- ASP.NET / Web.API Integration
- VisualStudio, teilweise Xamarin/MonoDevelop
- F# -> Javascript Compiler
- einfachste Client/Server Kommunikation
- HTML Templating
- Sitelets/Pagelets/...

# SITELETS

# SITELETS

MVC Framework (Serverseite)

Typsichere Links, Routen und Content parametrisiert über einen *Aktions*datentyp.

# AKTIONSTYP

Entspricht den Seiten/Aktionen:

```
type Aktion =
    | Home
    | Goto of SharpShortener.Id
    ...
```

# ROUTER

Bildet *Aktionen* auf URLs und URLs auf *Aktionen* ab.

```
let router : Router<Aktion> =
    [
        [
            Aktion.Home, "/"
            ...
        ] |> Router.Table
        Router.New
            (fun request -> ... )
            (fun aktion -> ... )
    ] |> Router.Sum
```

# CONTROLLER

Ordnet *Aktionen* **Content** zu.

```
let controller : Controller<Aktion> =
    {
        Handle = function
            | Aktion.Home    -> homeContent
            ...
    }
```

# CONTENT

Erzeugt aus einem `Context` (Request, Resourcen, ...) eine HTTP Antwort.

```
let homeContent =
    Skin.WithTemplate "#-Shortener" <| fun ctx ->
        [
            Div [new Controls.Shorten()]
            A [HRef (ctx.Link Aktion.Stats)] -< [Text "Statistiken"]
        ]
```

# HTML TEMPLATES

# TEMPLATE

```
<!DOCTYPE html>
<head>
    <title>${title}</title>
</head>
<body>
    <div class="templateContent">
        <div data-replace="body"></div>
    </div>
</body>
```

# HOLES

- `${Name}`: Platzhalter der mit Text ersetzt wird
- `data-replace="Name"`: Platzhalter-Element das komplett ersetzt wird
- `data-hole="Name"`: Platzhalter-Element, wird mit Kindelementen gefüllt.

# SKIN

Verarbeitet ein Template und wandelt es in Content um.

```
type Page =
    {
        Title : string
        Body : list<Content.HtmlElement>
    }

let MainTemplate =
    Content.Template<Page>("~/Main.html")
        .With("title", fun x -> x.Title)
        .With("body", fun x -> x.Body)
```

# HILFSFUNKTIONEN

```
let WithTemplate title body : Content<Aktion> =
    Content.WithTemplate MainTemplate <| fun context ->
        {
            Title = title
            Body = body context
        }
```

damit

```
let homeContent =
    Skin.WithTemplate "#-Shortener" <| fun ctx ->
        [
            Div [new Controls.Shorten()]
            A [HRef (ctx.Link Aktion.Stats)] -< [Text "Statistiken"]
        ]
```

# PAGELETS

# FUNKTIONSWEISE

Werden auf dem Server erstellt, an den Client übertragen und dort gerendert bzw. in die Seite eingefügt.

# HTML KOMBINATOREN

Für die meisten HTML Elemente gibt es entsprechende F# Funktionen/Kombinatoren.

```
Div [Width "200px"] -< [
    H1 [Text "Hallo DWX"]
]
```

# HTML EVENTS

Einige Ereignisse werden direkt von den Kombinatoren unterstützt und können mit

```
let ( |>! ) x f = f x; x
```

einfach angehängt werden

```
Div [
    Input [Attr.Type "button"; Attr.Value "drück mal!"]
    |>! OnClick (fun element eventArguments ->
        element.Value <- ";)")
]
```

# HTML EVENTS

*low-level* Handler

Können über die `Dom`-Eigenschaft angehängt werden:

```
btn.Dom.AddEventListener(
    "click",
    (fun () -> JavaScript.Alert("Click")), false)
```

normalerweise sollte auf `OnAfterRender` gewartet werden:

```
let btn =
    Button [Text "Test"]
    |>! OnAfterRender (fun el ->
        el.Dom.AddEventListener(
            "click",
            (fun () -> JavaScript.Alert("Click")),
            false))
```

# FORMLETS

# FORMLETS

Erstellen von typsichere interaktiver *Forms* und Benutzerschnittstellen.

```
let InputForm =
    Controls.Input ""
    |> Enhance.WithTextLabel "Nachricht"
    |> Enhance.WithSubmitButton
    |> Enhance.WithFormContainer
```

# LINKS

# LINKS

- WebSharper
- WebSharper UI next
- Material des Vortrags