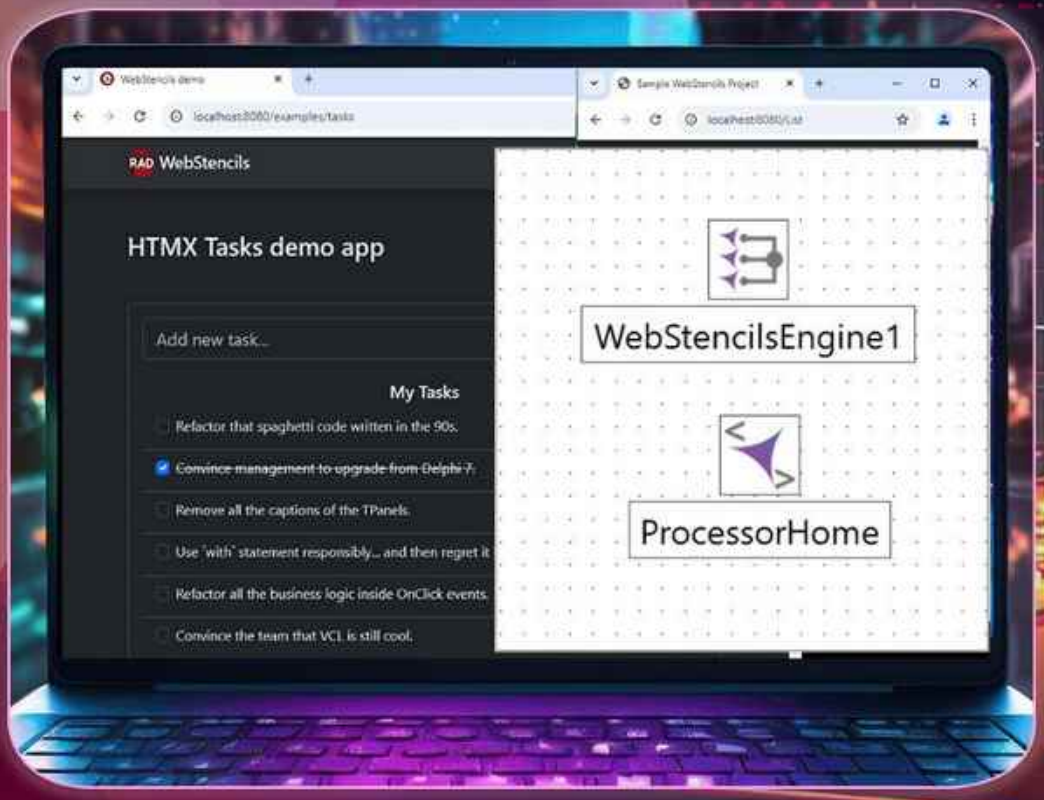
HTMX & WebStencils

RAD Studio (13.0)



Author
Antonio Zapater

embarcadero

HTMX WebStencils Web .

HTMX Web JavaScript Web
JavaScript
.

HTMX RAD Studio
.

WebStencils

.
.

HTMX WebStencils
Web . Web Web
RAD Studio Web .

RAD Studio
https://www.embarcadero.com/products/rad-studio

Web !

> *WebStencils HTMX*
> *GitHub .*
> *https://github.com/Embarcadero/WebStencilsDemos*
>
> ***:*** *https://wsdemo.embarcadero.com*

01

HTMX

HTMX?

HTMX HTML AJAX CSS WebSocket
HTML (SSE). JavaScript
HTML .

HTMX :

- : HTMX HTML .
- : HTMX .
- : .
- : HTMX .

JavaScript AJAX

Web JavaScript AJAX DOM.

.

HTMX :

- : JavaScript
  HTML .
- : AJAX div HTMX
  .
- : HTML .
- : HTML HTMX .

:

JavaScript/AJAX:

```
<button id="loadButton">Load Content</button>
<div id="content"></div>

<script>
document.getElementById('loadButton').addEventListener('click', function() {
    fetch('/some-content')
        .then(response => response.text())
        .then(data => {
            document.getElementById('content').innerHTML = data;
        });
});
</script>
```

HTMX:

```
<button hx-get="/some-content" hx-target="#content">Load Content</button>
<div id="content"></div>
```

HTMX HTML .

HTMX Web .

hx-get: GET

`hx-get` GET . hx-get
HTMX URL GET
.

:

```
<button hx-get="/api/user" hx-target="#user-info">
    Load User Info
</button>
<div id="user-info"></div>
```

HTMX "/api/user" GET id "user-info" div .

hx-target:

, `hx-target`
.

:

```
<button hx-get="/api/notification" hx-target="#notification-area">
    Check Notifications
</button>
<div id="notification-area"></div>
```

"/api/notification" id "notification-area" div .

hx-post: POST

`hx-get` , `hx-post` POST
.

:

```
<form hx-post="/api/submit" hx-target="#response">
    <input type="text" name="username">
    <button type="submit">Submit</button>
</form>
<div id="response"></div>
```

HTMX "/api/submit" POST #response div .

hx-put, hx-patch, hx-delete

`hx-get` `hx-post` . `hx-put`, `hx-patch` `hx-delete` PUT PATCH DELETE .

.

hx-swap:

`hx-swap` :

- `innerHTML` ( ): HTML
- `outerHTML`:
- `beforebegin`:
- `afterbegin`:
- `beforeend`:
- `afterend`:

:

```
<div id="list">
    <button hx-get="/api/item" hx-target="#list" hx-swap="beforeend">
        Add Item
    </button>
</div>
```

.

HTMX :

1. `hx-trigger`: AJAX Click Submit .
2. `hx-params`: .
3. `hx-headers`: AJAX .
4. `hx-vals`: .
5. `hx-boost`: AJAX .
6. `hx-push-url`: URL URL .

HTMX Web .
HTMX JavaScript
.

HTMX : https://htmx.org/docs

02

WebBroker

WebBroker?

WebBroker RAD Studio Web Web
RESTful SOAP .

WebBroker

1. : WebBroker Web Apache ISAPI…
.
2. : RAD Studio Delphi C++Builder .
3. : WebBroker .
4. : .

WebBroker WebStencils Web .

WebBroker HTTP

:

1. `TWebModule`: WebBroker WebBroker
   .
2. `TWebDispatcher`: HTTP .
3. `TWebActionItem`: URL URL
   URL .

WebBroker :

1. HTTP
2. `TWebDispatcher` TWebActionItem
3. `TWebActionItem`
4.

WebBroker

WebBroker :

1. “File/New/Other…/Web/Web Server Application”.
2. Linux .
3. Apache .
4. 8080 .

WebModule .

TWebModule TWebActionItems URL TWebModule
Actions.

RAD Studio .

TWebActionItem :

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content := '<html><body><h1>Hello, WebBroker!</h1></body></html>';
```

```
  Handled := True;
end;
```

URL HTML .

WebBroker HTTP :

- `TWebRequest` HTTP .
- `TWebResponse` .

:

```
procedure TWebModule1.HandleGetUser(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  UserId: string;
begin
  UserId := Request.QueryFields.Values['id'];
  // Fetch user data based on UserId
  Response.ContentType := 'application/json';
  Response.StatusCode := 200;
  Response.Content := '{"id": "' + UserId + '", "name": "John Doe"}';
  Handled := True;
end;
```

WebBroker :

1. : Web CLI .
2. **Apache/ISAPI** : IIS Apache Web .

Web .
ISAPI Apache Web
.

WebBroker

WebBroker WebBroker

.

CSRF .

:

1. :

.

2. :

.

3. :

.

4. : Redis

.

L :

```
unit SessionManagerU;

interface

uses
  System.Generics.Collections, System.SysUtils, Web.HTTPApp;

type
  TSessionData = class
  private
    FValues: TDictionary<string, string>;
  public
    constructor Create;
    destructor Destroy; override;
    property Values: TDictionary<string, string> read FValues;
  end;
```

```
  TSessionManager = class
  private
    FSessions: TDictionary<string, TSessionData>;
    function GenerateSessionId: string;
  public
    constructor Create;
    destructor Destroy; override;
    function GetSession(const SessionId: string): TSessionData;
    function CreateSession: string;
    procedure RemoveSession(const SessionId: string);
  end;

implementation

uses
  System.Hash;

{ TSessionData }

constructor TSessionData.Create;
begin
  inherited;
  FValues := TDictionary<string, string>.Create;
end;

destructor TSessionData.Destroy;
begin
  FValues.Free;
  inherited;
end;

{ TSessionManager }

constructor TSessionManager.Create;
begin
  inherited;
  FSessions := TDictionary<string, TSessionData>.Create;
end;

destructor TSessionManager.Destroy;
begin
  for var Session in FSessions.Values do
    Session.Free;
  FSessions.Free;
  inherited;
end;
```

```
function TSessionManager.GenerateSessionId: string;
begin
  var GUID := TGuid.NewGuid.ToString;
  Result := THashSHA2.GetHashString(GUID);
end;

function TSessionManager.GetSession(const SessionId: string): TSessionData;
begin
  if not FSessions.TryGetValue(SessionId, Result) then
    Result := nil;
end;

function TSessionManager.CreateSession: string;
var
  SessionId: string;
  SessionData: TSessionData;
begin
  SessionId := GenerateSessionId;
  SessionData := TSessionData.Create;
  SessionData.Values.AddOrSetValue('SessionId', SessionId);
  FSessions.Add(SessionId, SessionData);
  Result := SessionId;
end;

procedure TSessionManager.RemoveSession(const SessionId: string);
var
  SessionData: TSessionData;
begin
  if FSessions.TryGetValue(SessionId, SessionData) then
  begin
    SessionData.Free;
    FSessions.Remove(SessionId);
  end;
end;

end.
```

WebModule

WebBroker WebModule :

```
type
```

```
  TWebModuleWithSession = class(TWebModule)
    procedure WebModule1DefaultHandlerAction(Sender: TObject;
      Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
  private
    FSessionManager: TSessionManager;
    function GetSessionId(Request: TWebRequest; Response: TWebResponse): string;
    function GetSession(Request: TWebRequest; Response: TWebResponse): TSessionData;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  end;

var
  WebModuleClass: TComponentClass = TWebModuleWithSession;

implementation

uses
  DateUtils;

{%CLASSGROUP 'System.Classes.TPersistent'}

{$R *.dfm}

constructor TWebModuleWithSession.Create(AOwner: TComponent);
begin
  inherited;
  FSessionManager := TSessionManager.Create;
end;

destructor TWebModuleWithSession.Destroy;
begin
  FSessionManager.Free;
  inherited;
end;

function TWebModuleWithSession.GetSessionId(Request: TWebRequest; Response:
TWebResponse): string;
const
  SessionCookieName = 'SessionId';
var
  LCookie: TCookie;
begin
  Result := Request.CookieFields.Values[SessionCookieName];
  if Result = '' then
  begin
    Result := FSessionManager.CreateSession;
```

```
    LCookie := Response.Cookies.Add;
    LCookie.Name := SessionCookieName;
    LCookie.Value := Result;
    LCookie.Path := '/';
    // For demo purposes, the Cookie is only valid for 1 minute
    LCookie.Expires := IncMinute(Now, 1);
  end;
end;

function TWebModuleWithSession.GetSession(Request: TWebRequest; Response: TWebResponse):
TSessionData;
var
  SessionId: string;
begin
  SessionId := GetSessionId(Request, Response);
  Result := FSessionManager.GetSession(SessionId);
end;
```

WebModule :

```
procedure TWebModuleWithSession.WebModule1DefaultHandlerAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  LSession: TSessionData;
begin
  LSession := GetSession(Request, Response);
  // Use Session.Values to store or retrieve session data
  LSession.Values.AddOrSetValue('LastAccess', DateTimeToStr(Now));
  var HTMLResponse := '''
  <html>
  <head><title>Web Server Application</title></head>
  <body>Web Server Application</body>
  ''';
  HTMLResponse := HTMLResponse +
  '<p>Session ID: <strong>' + LSession.Values['SessionId'] + '</strong></p>' +
  '<p>Last Access: <strong>' + LSession.Values['LastAccess'] + '</strong></p>';
  HTMLResponse := HTMLResponse + '''
  </html>
  ''';
  Response.Content := HTMLResponse;
end;
```

> **Note** *SessionManager* . *TSessionData* *SessionManager* *(singleton)* .

CSRF

(CSRF) Web

CSRF:

1. : .
2. GET .

WebBroker :

```
unit CsrfProtection;

interface

uses
  System.SysUtils, Web.HTTPApp, SessionManager;

type
  TCsrfWebModule = class(TWebModule)
  private
    FSessionManager: TSessionManager;
    function GenerateCSRFToken: string;
  public
    procedure SendHTMLResponse(Sender: TObject;
      Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
    procedure ValidateCSRFToken(Request: TWebRequest);
    // The business logic related to session is the same as in the previous examples
    function GetSessionId(Request: TWebRequest; Response: TWebResponse): string;
    function GetSession(Request: TWebRequest; Response: TWebResponse): TSessionData;
  end;

implementation

uses
```

```
  System.Hash;

function TCsrfWebModule.GenerateCSRFToken: string;
begin
  var GUID := TGuid.NewGuid.ToString;
  Result := THashSHA2.GetHashString(GUID);
end;

procedure TCsrfWebModule.SendHTMLResponse(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  CSRFToken: string;
  Session: TSessionData;
begin
  Session := GetSession(Request, Response);
  CSRFToken := GenerateCSRFToken;
  LSession.Values.AddOrSetValue('CSRFToken', LCSRFToken);
  Response.Content :=
    '<form hx-post="/submit">' +
    '<input type="hidden" name="csrf_token" value="' + CSRFToken + '">' +
    // ... rest of your form ...
    '</form>';
  Handled := True;
end;

procedure TCsrfWebModule.ValidateCSRFToken(Request: TWebRequest);
var
  RequestToken, SessionToken: string;
  Session: TSessionData;
begin
  Session := GetSession(Request, Response);
  RequestToken := Request.ContentFields.Values['csrf_token'];
  SessionToken := Session.Values['CSRFToken'];

  if (RequestToken = '') or (SessionToken = '') or (RequestToken <> SessionToken) then
    raise Exception.Create('Invalid CSRF token');
end;

end.
```

:

1. CSRF .
2. .
3. .

.

:

```
procedure TWebModule1.ValidateUserInput(const Username: string);
begin
  if Length(Username) < 3 then
    raise Exception.Create('Username must be at least 3 characters long');

  if not TRegEx.IsMatch(Username, '^[a-zA-Z0-9_]+$') then
    raise Exception.Create('Username can only contain letters, numbers, and
underscores');
end;

procedure TWebModule1.HandleUserRegistration(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  Username: string;
begin
  Username := Request.ContentFields.Values['username'];
  try
    ValidateUserInput(Username);
    // Process registration...
    Response.Content := 'Registration successful';
  except
    on E: Exception do
      Response.Content := 'Registration failed: ' + E.Message;
  end;
  Handled := True;
end;
```

(XSS)

XSS HTML .
NetEncoding HTML .

:

```
function HTMLEncode(const S: string): string;
begin
  Result := TNetEncoding.HTML.Encode(S);
```

```
end;

procedure TWebModule1.DisplayUserComment(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  var UserComment := Request.QueryFields.Values['comment'];
  Response.Content := '<div class="comment">' + HTMLEncode(UserComment) + '</div>';
   Handled := True;
end;
```

1. **SQL** : .
2. : HTTPS .
3. : .

WebBroker .

.

DocWiki : WebBroker

03

WebBroker Web

WebBroker Web
“ Hello World ” .

“ Hello World ”

Hello World HTMX WebBroker .

1. RAD Studio WebBroker .
2. WebModule WebActionItem “ ActionHello ” .
3. ActionHello .

4. :

```
procedure TWebModule1.WebModule1ActionHelloAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content := '''
    <html>
      <head>
        <script src="https://unpkg.com/htmx.org@2.0.2"></script>
      </head>
      <body>
        <h1>Hello, WebBroker and HTMX!</h1>
        <button hx-get="/greet" hx-target="#greeting">Say Hello</button>
        <div id="greeting"></div>
      </body>
    </html>
    ''';
  Handled := True;
end;
```

5. "ActionGreet" WebActionItem :

```
procedure TWebModule1.WebModule1ActionGreetAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content := '<p>Hello from the server!</p>';
  Handled := True;
end;
```

6. “ Say Hello ”

.

WebBroker HTML HTMX .

.

1. WebBroker WebBroker .

2. :

```
unit TodoList;

interface

uses
  System.Generics.Collections;

type
  TTodoItem = record
    Id: Integer;
    Text: string;
  end;

  TTodoList = class
  private
    FItems: TList<TTodoItem>;
    FNextId: Integer;
  public
    constructor Create;
    destructor Destroy; override;
    function AddItem(const Text: string): Integer;
    function GetItems: TArray<TTodoItem>;
  end;

implementation

{ TTodoList }

constructor TTodoList.Create;
begin
  FItems := TList<TTodoItem>.Create;
  FNextId := 1;
end;

destructor TTodoList.Destroy;
begin
  FItems.Free;
  inherited;
end;

function TTodoList.AddItem(const Text: string): Integer;
var
  Item: TTodoItem;
begin
  Item.Id := FNextId;
  Item.Text := Text;
  FItems.Add(Item);
```

```
  Result := FNextId;
  Inc(FNextId);
end;

function TTodoList.GetItems: TArray<TTodoItem>;
begin
  Result := FItems.ToArray;
end;

end.
```

3. WebModule TodoList :

```
FTodoList: TTodoList;
```

WebModule .

4. WebActionItems
:

```
procedure TWebModule1.WebModule1ActionTodoListAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  Html: string;
  Item: TTodoItem;
Begin
  Html := '''
    <html>
      <head>
        <script src="https://unpkg.com/htmx.org@2.0.2"></script>
      </head>
      <body>
        <h1>Todo List</h1>
        <form hx-post="/add-todo" hx-target="#todo-list">
          <input type="text" name="todo-text" placeholder="New todo item">
          <button type="submit">Add</button>
        </form>
        <div id="todo-list">
    ''';

  for Item in FTodoList.GetItems do
  begin
```

```
    Html := Html + Format('<p>%d: %s</p>', [Item.Id, Item.Text]);
  end;

  Html := Html + '''
        </div>
      </body>
    </html>
    ''';

  Response.Content := Html;
  Handled := True;
end;

procedure TWebModule1.WebModule1ActionAddTodoAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  TodoText: string;
  Html: string;
  Item: TTodoItem;
begin
  TodoText := Request.ContentFields.Values['todo-text'];
  FTodoList.AddItem(TodoText);

  Html := '';
  for Item in FTodoList.GetItems do
  begin
    Html := Html + Format('<p>%d: %s</p>', [Item.Id, Item.Text]);
  end;

  Response.Content := Html;
  Handled := True;
end;
```

5.

.

WebBroker GET POST HTMX .

. WebBroker HTMX Web .

04

HTMX

HTMX HTMX WebBroker Web
.

HTMX AJAX DOM
:

hx-put hx-delete: PUT DELETE

`hx-get` `hx-post` , HTMX HTTP PUT DELETE.

`hx-put` :

```
<button hx-put="/api/user/1" hx-target="#user-info">
```

```
    Update User
</button>
```

`hx-delete`: :

```
<button hx-delete="/api/user/1" hx-target="#user-list">
    Delete User
</button>
```

WebBroker :

```
procedure TWebModule1.HandlePutRequest(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  if Request.MethodType = mtPUT then
  begin
    // Handle PUT request
    Response.Content := 'User updated';
    Handled := True;
  end;
end;

procedure TWebModule1.HandleDeleteRequest(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  if Request.MethodType = mtDELETE then
  begin
    // Handle DELETE request
    Response.Content := 'User deleted';
    Handled := True;
  end;
end;
```

hx-trigger:

`hx-trigger` AJAX 'Click' 'Submit'.

:

```
<input type="text"
       name="search"
       hx-get="/search"
       hx-trigger="keyup changed delay:500ms"
       hx-target="#search-results">
```

500 .

hx-select:

`hx-` .

Example:

```
<button hx-get="/api/user"
        hx-target="#user-name"
        hx-select="#name">
    Load User Name
</button>
```

id "name" .

HTML HTMX

HTMX .

HTML HTMX

HTML .

hx-include:

`hx-include` .

:

```
<form hx-post="/api/submit" hx-include="#extra-data">
    <input type="text" name="username">
    <button type="submit">Submit</button>
</form>
<input type="hidden" id="extra-data" name="extra" value="some-value">
```

"extra-dat"HTML .

hx-push-url: URL

`hx-push-url` URL

.

:

```
<button hx-get="/new-page"
        hx-push-url="true">
    Go to New Page
</button>
```

URL / .

05

WebStencils

WebStencils?

WebStencils HTML , RAD Studio 12.2 .
JavaScript RAD Studio
.

WebStencils , wikis.
ASP.NET Razor RAD Studio .

HTMX

WebStencils HTMX HTMX REST
Delphi Web REST API.

CSS JS

WebStencils JavaScript CSS .
.

WebStencils

WebStencils :

1. @
2. { }

@

@ WebStencils :

-
-
- @

:

```
@object.value
```

“ object ” “ value ” .
OnValue .

{ }

WebStencils .

WebStencils :

```
<h2>User Profile</h2>
<p>Name: @user.name</p>
<p>Email: @user.email</p>
```

WebStencils

WebStencils :

@page

`@page` k .

:

```
<p>Current page is: @page.pagename</p>
```

@query

`@query` HTTP .

:

```
<p>You searched for: @query.searchTerm</p>
```

searchTerm URL : `yourdomain.com?searchTerm=”mySearch”`

(@* .. *@)

WebStencil `@* *@` HTML .

:

```
@* This is a comment and will not appear in the output *@
<p>This will appear in the output</p>
```

@if @else

`@if` `@else`.

:

```
@if user.isLoggedIn {
    <p>Welcome, @user.name!</p>
}
```

```
@else {
    <p>Please log in to continue.</p>
}
```

@if not

`@if not`.

:

```
@if not cart.isEmpty {
    <p>You have @cart.itemCount items in your cart.</p>
}
@else {
    <p>Your cart is empty.</p>
}
```

@ForEach

`@ForEach` .

:

```
<ul>
@ForEach (var product in productList) {
    <li>@product.name - @product.price</li>
}
</ul>
```

WebStencils RAD Studio .
HTML . @ ,
.

WebStencils Web .
WebStencils
WebStencils .

06

WebStencils .
WebStencils Web .

WebStencils

WebStencils WebStencils WebStencils
HTML .

WebStencils

WebStencils :

1. **WebStencilsProcessor** :
.
2. : WebStencilsProcessor Web
.

`TWebStencilsEngine` :

- `Dispatcher`: (`IWebDispatch`) .
- `PathTemplates`: .
- `RootDirectory`: .
- `DefaultFileExt`: (“ .html ”).
- `AddVar`: .
- `AddModule`: `[WebStencilsVar]` .

WebStencils

WebStencils (HTML) .
WebStencils .

`TWebStencilsProcessor` :

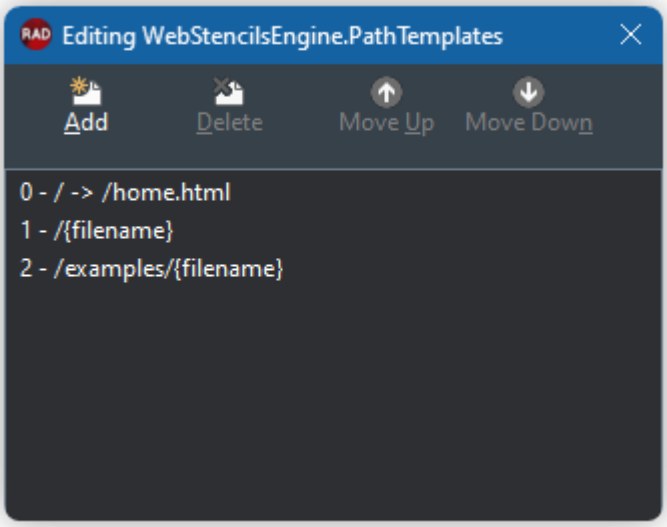
- `InputFilename`: .
- `InputLines`: .
- `Engine`: ( ).
- `Content`: .
- `AddVar`: .

TWebStencilsEngine WebBroker

WebStencils `TWebFileDispatcher` WebBroker .

PathTemplates .

:



:

`0-/->/home.html`: `home.html`

`1-/{filename}`: URI . https://localhost:8080/basics `basics.html` .

`2-/examples/{filename}`: `/example`

AddVar

`AddVar` Delphi WebStencils AddVar :

1. :

```
WebStencilsProcessor1.AddVar('user', UserObject);
```

2. :

```
WebStencilsProcessor1.AddVar('products',
  function: TObject
```

```
  begin
    Result := GetProductList;
  end);
```

3. : `[WebStencilsVar]` , `AddModule` :

```
type
  TMyDataModule = class(TDataModule)
    [WebStencilsVar]
    FDMemTable1: TFDMemTable;
    [WebStencilsVar]
    function GetCurrentUser: TUser;
  end;

// In your WebModule:
WebStencilsProcessor1.AddModule(DataModule1);
```

: `GetEnumerator` ( )
WebStencils (Record).

WebStencils Mustache Blade ERB Razor .
.

@RenderBody

`@RenderBody` d
HTML BaseTemplate.html:

```
<!-- This is the BaseTemplate.html -->
<!DOCTYPE html>
<html>
<head>
```

```
    <title>My Website</title>
</head>
<body>
    <header>
        <!-- Common header content -->
    </header>

    <main>
        @RenderBody
    </main>

    <footer>
        <!-- Common footer content -->
    </footer>
</body>
</html>
```

`@Renderbody` .

@LayoutPage

`@LayoutPage` :

```
<!-- BaseTemplate.html will be used as a base and the rest of the content included where
the @RenderBody tag is located -->
@LayoutPage BaseTemplate
<h2>Welcome to My Page</h2>
<p>This is the content of my page.</p>
```

BaseTemplate.html @LayoutPage @RenderBody .

@Import

`@Import` .

.

```
@Import Sidebar.html

@* Same Behavior *@
@Import Sidebar

@* Nested folder example *@
@Import folder/Sidebar
```

@ExtraHeader @RenderHeader

`@ExtraHeader` HTML
CSS JavaScript .

:

1. @ExtraHeader .
2. @RenderHeader .

:

(ProductPage.html):

```
@LayoutPage BaseTemplate
@ExtraHeader {
    <link rel="stylesheet" href="/css/product-page.css">
    <script src="/js/product-details.js"></script>
}

<h1>Product Details</h1>
<div id="product-info">
    <!-- Product information here -->
</div>
```

(BaseTemplate.html):

```
<!DOCTYPE html>
<html>
<head>
    <title>My Online Store</title>
```

```
    <link rel="stylesheet" href="/css/main.css">
    @RenderHeader
</head>
<body>
    <main>
        @RenderBody
    </main>
</body>
</html>
```

CSS JavaScript .
`@RenderHeader` HTML .

.

@ExtraHeader .
@RenderHeader .

@LayoutPage @RenderBody @ExtraHeader @RenderHeader
.

WebStencils .

@LayoutPage @RenderBody .
.

Header/Body/Footer

:

```
@Import Header.html

<main>
    <!-- Page-specific content here -->
</main>
```

```
@Import Footer.html
```

.

@Import .

:

```
<div class="product-list">
  @Import ProductList { @list = @ProductList }
</div>

<div class="tasks">
  @ForEach (var Task in Tasks.AllTasks) {
            @Import partials/tasks/item { @Task }
      }
</div>
```

UI

.

WebStencils Web .
WebStencils AddVar @LayoutPage
@RenderBody @Import Web

.

- / / -

. .

WebStencils

.

07

WebStencils

HTML

WebStencils WebStencils

.

WebStencils GitHub

.

HTML

HTML WebStencils .

.

`layout.html` :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://unpkg.com/htmx.org@@1.9.2"></script>
    @RenderHeader
</head>
<body>
    <div class="container">
        @RenderBody
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@@5.1.3/dist/js/bootstrap.bundle.min.js"></scr
ipt>
</body>
</html>
```

CDN
@@ @ WebStencils
.

todo-list.html:

```
@LayoutPage layout.html

<div id="todo-list">
    @ForEach (var todo in Todos) {
        <div class="card mb-2">
            <div class="card-body d-flex justify-content-between align-items-center">
                <span>@todo.Description</span>
                <div>
```

```
                    @if not todo.Completed {
                        <button class="btn btn-sm btn-success" hx-
post="/complete/@todo.Id" hx-target="#todo-list">Complete</button>
                    }
                    <button class="btn btn-sm btn-danger" hx-delete="/delete/@todo.Id"
hx-target="#todo-list">Delete</button>
                </div>
            </div>
        </div>
    }
</div>

<form hx-post="/add" hx-target="#todo-list" class="mt-4">
    <div class="input-group">
        <input type="text" name="description" class="form-control" placeholder="New todo
item" required>
        <button type="submit" class="btn btn-primary">Add</button>
    </div>
</form>
```

WebModule

WebModule TTodoList

Delete Complete GetAllItems .

Delphi GitHub .

```
unit TodoWebModule;

interface

uses
  System.SysUtils, System.Classes, Web.HTTPApp, TodoList, WebStencils;

type
  TTodoWebModule = class(TWebModule)
    procedure WebModuleCreate(Sender: TObject);
    procedure WebModuleDestroy(Sender: TObject);
    procedure WebModuleDefault(Sender: TObject; Request: TWebRequest;
      Response: TWebResponse; var Handled: Boolean);
  private
    FTodoList: TTodoList;
    FWebStencilsProcessor: TWebStencilsProcessor;
```

```
    procedure HandleGetTodoList(Response: TWebResponse);
    procedure HandleAddTodo(Request: TWebRequest; Response: TWebResponse);
    procedure HandleCompleteTodo(Request: TWebRequest; Response: TWebResponse);
    procedure HandleDeleteTodo(Request: TWebRequest; Response: TWebResponse);
  public
    { Public declarations }
  end;

var
  TodoWebModule: TTodoWebModule;

implementation

{%CLASSGROUP 'System.Classes.TPersistent'}

{$R *.dfm}

procedure TTodoWebModule.WebModuleCreate(Sender: TObject);
begin
  FTodoList:= TTodoList.Create;
  FWebStencilsProcessor := TWebStencilsProcessor.Create(Self);
  FWebStencilsProcessor.TemplateFolder := ExtractFilePath(ParamStr(0)) + 'templates\';
end;

procedure TTodoWebModule.WebModuleDestroy(Sender: TObject);
begin
  FTodoList.Free;
  FWebStencilsProcessor.Free;
end;

procedure TTodoWebModule.WebModuleDefault(Sender: TObject; Request: TWebRequest;
  Response: TWebResponse; var Handled: Boolean);
Begin
// This is a WebBroker action that handles all the requests. For better maintainability,
it should be split into different actions.
  if Request.PathInfo = '' then
    HandleGetTodoList(Response)
  else if (Request.PathInfo = '/add') and (Request.MethodType = mtPost) then
    HandleAddTodo(Request, Response)
  else if Request.PathInfo.StartsWith('/complete/') and (Request.MethodType = mtPost)
then
    HandleCompleteTodo(Request, Response)
  else if Request.PathInfo.StartsWith('/delete/') and (Request.MethodType = mtDelete)
then
    HandleDeleteTodo(Request, Response)
  else
  begin
```

```
    Response.Content := 'Not Found';
    Response.StatusCode := 404;
  end;

  Handled := True;
end;

procedure TTodoWebModule.HandleGetTodoList(Response: TWebResponse);
begin
  FWebStencilsProcessor.AddVar('Todos', FTodoList.GetAllItems);
  FWebStencilsProcessor.InputFileName := 'todo-list.html';
  Response.Content := FWebStencilsProcessor.Content;
end;

procedure TTodoWebModule.HandleAddTodo(Request: TWebRequest; Response: TWebResponse);
var
  Description: string;
begin
  Description := Request.ContentFields.Values['description'];
  FTodoList.AddItem(Description);
  HandleGetTodoList(Response);
end;

procedure TTodoWebModule.HandleCompleteTodo(Request: TWebRequest; Response:
TWebResponse);
var
  ItemId: Integer;
Begin
  // The ItemId is extracted from the PathInfo of the Request and converted to int
  ItemId := StrToIntDef(Request.PathInfo.Substring('/complete/'.Length), -1);
  if ItemId <> -1 then
    FTodoList.CompleteItem(ItemId);
  HandleGetTodoList(Response);
end;

procedure TTodoWebModule.HandleDeleteTodo(Request: TWebRequest; Response: TWebResponse);
var
  ItemId: Integer;
Begin
  ItemId := StrToIntDef(Request.PathInfo.Substring('/delete/'.Length), -1);
  if ItemId <> -1 then
    FTodoList.DeleteItem(ItemId);
  HandleGetTodoList(Response);
end;

end.
```

Actions.

GitHub *MVC* .

WebStencils

.

TTodoList TTodoItem :

```
TTodoItem = class
public
  Id: Integer;
  Description: string;
  Completed: Boolean;
  Category: string;
  constructor Create(AId: Integer; const ADescription, ACategory: string);
end;

constructor TTodoItem.Create(AId: Integer; const ADescription, ACategory: string);
begin
  Id := AId;
  Description := ADescription;
  Completed := False;
  Category := ACategory;
end;
```

todo-list.html :

WebModule HandleAddTodo :

```
@LayoutPage layout.html

<div id="todo-list">
    @ForEach (var todo in Todos) {
        <div class="card mb-2">
            <div class="card-body d-flex justify-content-between align-items-center">
                <div>
                    <span>@todo.Description</span>
                    <small class="text-muted ms-2">[@todo.Category]</small>
                </div>
                <div>
                    @if not todo.Completed {
                        <button class="btn btn-sm btn-success"
                                hx-post="/complete/@todo.Id"
                                hx-target="#todo-list">Complete</button>
                    }
                    <button class="btn btn-sm btn-danger"
                            hx-delete="/delete/@todo.Id"
                            hx-target="#todo-list">Delete</button>
                </div>
            </div>
        </div>
    }
</div>

<form hx-post="/add" hx-target="#todo-list" class="mt-4">
    <div class="input-group">
        <input type="text"
               name="description"
               class="form-control"
               placeholder="New todo item" required>
        <input type="text" name="category" class="form-control" placeholder="Category">
        <button type="submit" class="btn btn-primary">Add</button>
    </div>
</form>
```

```
procedure TTodoWebModule.HandleAddTodo(Request: TWebRequest; Response: TWebResponse);
var
  Description, Category: string;
begin
  Description := Request.ContentFields.Values['description'];
  Category := Request.ContentFields.Values['category'];
  FTodoList.AddItem(Description, Category);
  HandleGetTodoList(Response);
```

```
end;
```

`category-filter.html`:

```
<div class="mb-4">
    <h5>Filter by Category</h5>
    <div class="btn-group" role="group">
        <button class="btn btn-outline-primary"
                hx-get="/" hx-target="#todo-list">All</button>
        @ForEach (var category in Categories) {
            <button class="btn btn-outline-primary"
                    hx-get="/filter/@category" hx-target="#todo-list">@category</button>
        }
    </div>
</div>
```

todo-list.html :

```
@LayoutPage layout.html

@Import category-filter.html

<div id="todo-list">
    <!-- ... existing todo list content ... -->
</div>

<!-- ... existing form ... -->
```

WebModule :

```
procedure TTodoWebModule.HandleFilterTodos(Request: TWebRequest; Response: TWebResponse);
var
  Category: string;
  FilteredTodos: TArray<TTodoItem>;
begin
  Category := Request.PathInfo.Substring('/filter/'.Length);
```

```
  FilteredTodos := FTodoList.GetItemsByCategory(Category);
  FWebStencilsProcessor.AddVar('Todos', FilteredTodos);
  FWebStencilsProcessor.AddVar('Categories', FTodoList.GetAllCategories);
  FWebStencilsProcessor.InputFileName := 'todo-list.html';
  Response.Content := FWebStencilsProcessor.Content;
end;
```

WebModuleDefault :

```
procedure TTodoWebModule.WebModuleDefault(Sender: TObject; Request: TWebRequest;
  Response: TWebResponse; var Handled: Boolean);
begin
  if Request.PathInfo = '' then
    HandleGetTodoList(Response)
  else if Request.PathInfo.StartsWith('/filter/') then
    HandleFilterTodos(Request, Response)
  // ... existing routes ...
end;
```

WebStencils *RAD Studio*

WebStencils *Delphi* .

HTML WebStencils .

. HTML

Delphi .

. WebStencils

.

08

WebStencils

WebStencils
Web .
.

(scaffolding) HTML OnValue
@ .

@()

@object.property

@() .

@() Delphi LiveBindings .

@()

:

RTL

.

```
<p>Format date: @(FormatDateTime('yyyy-mm-dd', order.Date))</p>
<p>UpperCase: @(UpperCase(customer.FirstName))</p>
<p>Round: @(Round(product.SalePrice))</p>

<!-- GetFullName() is a public method within customer object -->
<p>Public method class: @(customer.GetFullNume)</p>
```

:

```
<p>Price with VAT: $@(product.Price * product.VatPercentage)</p>
<p>Discount: $@(product.basePrice - product.salePrice)</p>
<p>Previous page: @(pagination.PageNumber - 1)</p>
```

:

```
<!-- TStringList -->
<p>First category: @(categories.Strings[0])</p>
<p>Selected color: @(colors.Strings[context.userChoice])</p>

<!-- TList<T> -->
<p>Product name: @(products.Items[2].Name)</p>

<!-- Classic static arrays (requires indexed property with getter) -->
<p>First item: @(data.ClassicArray[0])</p>
```

:

- `TStringList`, `.Strings[index]`
- `TList<T>`, `.Items[index]`
- (`array[0..N] of Type`), Delphi getter

@object.property *@()*
Delphi
.

@Scaffolding

@Scaffolding HTML
HTML HTML
.

@Import
.

:

```
<form>
  @Scaffolding User
</form>
```

WebStencilsProcessor OnScaffolding :

```
procedure TMyWebModule.ProcessorScaffolding(Sender: TObject;
  const AQualifClassName: string; var AReplaceText: string);
begin
  if SameText(AQualifClassName, 'User') then
  begin
    AReplaceText := '';
    // Generate form fields based on User properties
    AReplaceText := AReplaceText + '<input type="text" name="Username"
placeholder="Username">';
    AReplaceText := AReplaceText + '<input type="email" name="Email"
placeholder="Email">';
    // ... add more fields as needed
  end;
end;
```

Scaffolding AReplaceText .

HTML

HTML :

```
procedure TMyWebModule.ProcessorScaffolding(Sender: TObject;
  const AQualifClassName: string; var AReplaceText: string);
var
  LContext: TRttiContext;
  LType: TRttiType;
  LProp: TRttiProperty;
  LFieldHtml: string;
begin
  if SameText(AQualifClassName, 'User') then
  begin
    AReplaceText := '<div class="form-group">';

    LContext := TRttiContext.Create;
    try
      LType := LContext.FindType('TUser');
      if LType <> nil then
      begin
        for LProp in LType.GetProperties do
        begin
          // Skip internal properties
          if not (LProp.Visibility in [mvPublic, mvPublished]) then
            Continue;

          // Generate appropriate input based on property type
          case LProp.PropertyType.TypeKind of
            tkInteger:
              LFieldHtml := Format('<input type="number" name="%s" class="form-
control">', [LProp.Name]);
            tkString, tkUString, tkLString, tkWString:
              LFieldHtml := Format('<input type="text" name="%s" class="form-
control">', [LProp.Name]);
            tkEnumeration:
              if LProp.PropertyType.Handle = TypeInfo(Boolean) then
                LFieldHtml := Format('<input type="checkbox" name="%s" class="form-
check-input">', [LProp.Name])
              else
                Continue;
          else
            Continue;
          end;
```

```
          AReplaceText := AReplaceText + Format(
            '<div class="mb-3">' +
            '<label class="form-label">%s</label>' +
            '%s' +
            '</div>', [LProp.Name, LFieldHtml]);
        end;
      end;
    finally
      LContext.Free;
    end;

    AReplaceText := AReplaceText + '</div>';
  end;
end;
```

(RTTI)

.

OnValue

`OnValue` .

.

WebStencils RTTI

.

OnValue

```
procedure TMyWebModule.ProcessorOnValue(Sender: TObject;
  const ObjectName, FieldName: string; var ReplaceText: string;
  var Handled: Boolean);
begin
  if SameText(ObjectName, 'CurrentTime') then
  begin
    ReplaceText := FormatDateTime('dd-mm-yyyy hh:nn:ss', Now);
    Handled := True;
  end;
end;
```

:

```
<p>Current time: @CurrentTime</p>
```

OnValue

OnValue :

1. Computed

```
procedure TMyWebModule.ProcessorOnValue(Sender: TObject;
  const ObjectName, FieldName: string; var ReplaceText: string;
  var Handled: Boolean);
begin
  if SameText(ObjectName, 'stats') then
  begin
    Handled := True;
    if SameText(FieldName, 'TotalRevenue') then
      ReplaceText := FormatFloat('$#,##0.00', CalculateTotalRevenue)
    else if SameText(FieldName, 'ActiveUsers') then
      ReplaceText := IntToStr(GetActiveUserCount)
    else if SameText(FieldName, 'ConversionRate') then
      ReplaceText := FormatFloat('0.00%', CalculateConversionRate * 100)
    else
      Handled := False;
  end;
end;
```

:

```
<div class="dashboard">
  <div class="stat">Total Revenue: @stats.TotalRevenue</div>
  <div class="stat">Active Users: @stats.ActiveUsers</div>
  <div class="stat">Conversion Rate: @stats.ConversionRate</div>
</div>
```

2. /

```
procedure TMyWebModule.ProcessorOnValue(Sender: TObject;
  const ObjectName, FieldName: string; var ReplaceText: string;
  var Handled: Boolean);
```

```
begin
  if SameText(ObjectName, 'i18n') then
  begin
    ReplaceText := GetTranslation(FieldName, CurrentUserLanguage);
    Handled := True;
  end;
end;
```

:

```
<h1>@i18n.WelcomeMessage</h1>
<p>@i18n.IntroductionText</p>
<button>@i18n.GetStartedButton</button>
```

3.

```
procedure TMyWebModule.ProcessorOnValue(Sender: TObject;
  const ObjectName, FieldName: string; var ReplaceText: string;
  var Handled: Boolean);
begin
  if SameText(ObjectName, 'feature') then
  begin
    Handled := True;
    if SameText(FieldName, 'enabled') then
      ReplaceText := BoolToStr(IsFeatureEnabled(CurrentUser, FieldName), True)
    else if SameText(FieldName, 'available') then
      ReplaceText := BoolToStr(IsFeatureAvailable(CurrentSubscription, FieldName),
True)
    else
      Handled := False;
  end;
end;
```

:

```
@if (feature.enabled) {
  <div class="premium-feature">
    <h2>Premium Feature</h2>
    <p>This feature is available with your subscription.</p>
```

```
    </div>
  }
```

OnValue AddVar

OnValue AddVar

AddVar RTTI .

OnValue.

`OnValue` `WebStencils` `RTTI`
`OnValue`.

@LoginRequired : WebStencils @LoginRequired
RAD Studio 13.0 WebStencils
.

9
.

@LoginRequired
.

WebStencils . `Scaffolding`
HTML OnValue @()
.

. @if
@ForEach .

09

WebBroker Web .

.

.

RAD Studio 13.0
Web

.

ID.

WebBroker

.

3

:

TWebSessionManager

.

ID cookie

+IP .

TWebFormsAuthenticator

HTML

.

URL

.

TWebAuthorizer

“ ” “ ”

.

"admin" /admin "user"

/user .

.

WebModule :

1. `TWebSessionManager`
2. `TWebFormsAuthenticator`
3. `TWebAuthorizer`

:

```
// In the Object Inspector or code:
```

```
// Where to send unauthenticated users
WebFormsAuthenticator.LoginURL := '/login';

// Where to redirect after successful login
WebFormsAuthenticator.HomeURL := '/';

// Where to redirect after failed login
WebFormsAuthenticator.FailedURL := '/login?error=1';
```

WebModule .

OnAuthenticate :

```
procedure TWebModule1.WebFormsAuthenticatorAuthenticate(
  Sender: TCustomWebAuthenticator;
  Request: TWebRequest;
  const UserName, Password: string; var Roles: string; var Success: Boolean);
begin
  // Validate credentials against your user database
  Success := False;
  Roles := '';

  // Example: Check against database
  if ValidateUserCredentials(UserName, Password) then
  begin
    Success := True;
    Roles := GetUserRoles(UserName);  // e.g., 'user,admin'
  end;
end;
```

Roles 'user,moderator'
'user,admin' .

ValidateUserCredentials
.

(login.html):

```
@LayoutPage layouts/baseLayout

<div class="login-container">
  <h1>Sign In</h1>

  @if query.error {
    <div class="alert alert-danger">
      Invalid username or password. Please try again.
    </div>
  }

  <form method="post" action="/login">
    <div class="form-group">
      <label for="username">Username</label>
      <input type="text" id="username" name="username"
             class="form-control" required autofocus>
    </div>

    <div class="form-group">
      <label for="password">Password</label>
      <input type="password" id="password" name="password"
             class="form-control" required>
    </div>

    <button type="submit" class="btn btn-primary">Sign In</button>
  </form>
</div>
```

/login .

`LogoutURL` POST :

```
// In the Object Inspector or code:
WebFormsAuthenticator.LogoutURL := '/logout';
```

/logout POST , URL `HomeURL`.

.

:

```
<form method="post" action="/logout">
  <button type="submit" class="btn btn-secondary">Sign Out</button>
</form>
```

.

@session

@session

.

Session

`@session` :

:

- `@session.Authenticated` -
- `@session.UserName` -
- `@session.UserRoles` -

:

- `@session.SessionID` -
- `@session.Timeout` - ( )
- `@session.CreatedTime` -
- `@session.AccessedTime` -
- `@session.AccessedCount` -

:

- `@session.LastURL` - URL
- `@session.LastIP` - IP
- `@session.LastReferer` -
- `@session.LastUserAgent` -

@session

:

```
<!-- Navbar showing different content based on authentication -->
<nav class="navbar">
  <a href="/" class="nav-brand">My App</a>

  <div class="nav-menu">
    <a href="/home" class="nav-link">Home</a>
    <a href="/about" class="nav-link">About</a>

    @if session.Authenticated {
      <a href="/dashboard" class="nav-link">Dashboard</a>

      @if session.UserHasRole('admin') {
        <a href="/admin" class="nav-link">Admin</a>
      }

      @if session.UserHasRole('moderator') {
        <a href="/moderate" class="nav-link">Moderate</a>
      }

      <a href="/logout" class="nav-link">Sign Out (@session.UserName)</a>
    } @else {
      <a href="/login" class="nav-link">Sign In</a>
      <a href="/register" class="nav-link">Register</a>
    }
  </div>
</nav>
```

UserHasRole()

.

.

Zones :

```
procedure TWebModule1.WebModuleCreate(Sender: TObject);
var
  Zone: TWebAuthorizationZone;
begin
  // Protect all /admin/* URLs - require 'admin' role
  Zone := WebAuthorizer.Zones.Add;
  Zone.PathInfo := '/admin*';
  Zone.Kind := zkProtected;
  Zone.Roles := 'admin';

  // Allow public access to documentation - no authentication required
  Zone := WebAuthorizer.Zones.Add;
  Zone.PathInfo := '/docs*';
  Zone.Kind := zkFree;

  // Health check endpoint - bypass all authentication and session management
  Zone := WebAuthorizer.Zones.Add;
  Zone.PathInfo := '/health*';
  Zone.Kind := zkIgnore;

  // Protected user dashboard - require any authenticated user
  Zone := WebAuthorizer.Zones.Add;
  Zone.PathInfo := '/dashboard*';
  Zone.Kind := zkProtected;
  Zone.Roles := 'user,admin';  // Either role works
end;
```

(*) /admin* /admin /admin/users /admin/settings .

:

- `zkProtected` - ( )
- `zkFree` -
- `zkIgnore` -

> *zkIgnore*
> ( ).

zkProtected

.

`UnauthorizedURL`.

:

```
Zone.Roles := 'admin,moderator,support';  // Any of these roles grants access
```

( )

OR .

`TWebSessionManager` .

ID

IdLocation ID :

```
// Store in cookies (default and recommended)
WebSessionManager.IdLocation := ilCookie;

// Store in HTTP headers (useful for APIs)
WebSessionManager.IdLocation := ilHeader;

// Store in query parameters (least secure, use only if necessary)
WebSessionManager.IdLocation := ilQuery;
```

Cookie Web Cookie API
Header ID URL
.

Scope :

```
// New session for each request (no specific user binding)
WebSessionManager.Scope := ssUnlimited;

// Session tied to authenticated user (identified by username and roles)
WebSessionManager.Scope := ssUser;

// Session tied to user AND IP address (more secure but may break if IP changes)
WebSessionManager.Scope := ssUserAndIP;
```

ssUnlimited ID
.

ssUser ID HMAC-SHA2 SharedSecret
ID
.

`ssUserAndIP` IP HMAC IP

IP .

Timeout ( ) :

```
// Set timeout to 30 minutes (1800 seconds)
WebSessionManager.Timeout := 1800;

// Set timeout to 2 hours (7200 seconds)
WebSessionManager.Timeout := 7200;
```

3600 (1)
AccessedTime
.

ssUser ssUserAndIP

```
WebSessionManager.SharedSecret := 'your-secret-key-here';
```

ID HMAC-SHA2 .

32
.

RAD Studio Web

.

@session UI.

10

Web IDE

HTML ?

? .

RAD Studio 13.0 UI

FireDAC

.

TField

WebModule .

ebStencils FireDAC
FireDAC .

:

```
@forEach(var f in customers.fields) {
  <div>
    <label>@f.DisplayLabel</label>
    <input type="text" name="@f.FieldName" value="@f.Value">
  </div>
}
```

customers :

- `DisplayLabel` - FireDAC
- `FieldName` -
- `Value` -

.

WebStencils
:

TDataSet ( ):

- Active, FieldByName, First, Last, Next, Prior
- Bof, Eof, FieldCount, Fields, Found, RecordCount, RecNo

TStrings ( ):

- Contains, ContainsName, IndexOf, IndexOfName
- CommaText, Count, IsEmpty, DelimitedText
- Names, KeyNames, Values, ValueFromIndex, Strings, Text

TField (____):

TField TField

DisplayLabel FieldName Value

.

WebModule TField :

```
procedure TWebModule1.WebModuleCreate(Sender: TObject);
begin
  // Configure TField whitelist - REQUIRED for database-driven forms
  // Without this, templates cannot access field properties
  TWebStencilsProcessor.Whitelist.Configure(
    TField,
    ['DisplayText', 'Value', 'DisplayLabel', 'FieldName',
     'Required', 'LookupDataSet', 'LookupKeyFields',
     'Visible', 'DataType', 'Size', 'IsNull'],
    nil,   // No properties blocked
    False  // Don't inherit from parent class restrictions
  );
end;
```

`@field.DisplayLabel`

.

.

:

```
<form method="post">
  @forEach(var field in customers.fields) {
    @if(field.Visible) {
      <div class="form-group">
        <label for="@field.FieldName">
          @field.DisplayLabel
          @if(field.Required) {
            <span class="text-danger">*</span>
```

```
          }
        </label>

        <input
          type="text"
          name="@field.FieldName"
          value="@field.Value"
          @if(field.Required) { required }
          @if(field.Size > 0) { maxlength="@field.Size" }>
      </div>
    }
  }

  <button type="submit">Save</button>
</form>
```

:

- DisplayLabel
-
- HTML5
-
-

.

@switch

@switch (RAD Studio 13.0) field.DataType
HTML :

```
@forEach(var field in customers.fields) {
  @if(field.Visible) {
    <div class="form-group">
      <label>@field.DisplayLabel</label>

      @switch(field.DataType) {
        @case "ftString" {
          <input type="text"
                 name="@field.FieldName"
```

```
                value="@field.Value"
                @if(field.Size > 0) { maxlength="@field.Size" }>
        }
        @case "ftInteger" {
          <input type="number"
                name="@field.FieldName"
                value="@field.Value"
                step="1">
        }
        @case "ftFloat" {
          <input type="number"
                name="@field.FieldName"
                value="@field.Value"
                step="0.01">
        }
        @case "ftDate" {
          <input type="date"
                name="@field.FieldName"
                value="@field.DisplayText">
        }
        @case "ftDateTime" {
          <input type="datetime-local"
                name="@field.FieldName"
                value="@(FormatDateTime('yyyy-mm-dd''T''hh:nn', field.Value))">
        }
        @case "ftBoolean" {
          <input type="hidden" name="@field.FieldName" value="False">
          <input type="checkbox"
                name="@field.FieldName"
                value="True"
                @if(field.Value) { checked }>
        }
        @case "ftMemo" {
          <textarea name="@field.FieldName" rows="3">@field.Value</textarea>
        }
        @default {
          <input type="text" name="@field.FieldName" value="@field.Value">
        }
      }
    </div>
  }
}
```

HTML5 <input type="number"> <input type="date"> .

HTML

False

.

/ .

dynamicInput.html :

```
@if(field.Visible) {
<div class="form-group">
  <label for="@field.FieldName">
    @field.DisplayLabel
    @if(field.Required) { <span class="text-danger">*</span> }
  </label>

  @switch(field.DataType) {
    @case "ftInteger" {
      <input type="number"
             class="form-control"
             name="@field.FieldName"
             value="@field.Value"
             @if(field.Required) { required }>
    }
    @case "ftDate" {
      <input type="date"
             class="form-control"
             name="@field.FieldName"
             value="@field.DisplayText"
             @if(field.Required) { required }>
    }
    @case "ftBoolean" {
      <div class="form-check">
        <input type="hidden" name="@field.FieldName" value="False">
        <input type="checkbox"
               class="form-check-input"
               name="@field.FieldName"
               value="True"
               @if(field.Value) { checked }>
        <label class="form-check-label">@field.DisplayLabel</label>
```

```
      </div>
    }
    @default {
      <input type="text"
             class="form-control"
             name="@field.FieldName"
             value="@field.Value"
             @if(field.Required) { required }
             @if(field.Size > 0) { maxlength="@field.Size" }>
    }
  }
</div>
} @else {
  <input type="hidden" name="@field.FieldName" value="@field.Value">
}
```

:

```
<form method="post">
  @forEach(var f in customers.fields) {
    @import partials/dynamicInput { @field = @f }
  }
  <button type="submit">Save</button>
</form>
```

.

( )

.

Tag

Tag 1

Tag .

:

```
<form method="post">
  <!-- Custom header section -->
  <div class="form-header">
    <h2>Customer Information</h2>
    <p>Please provide accurate contact details.</p>
  </div>

  <!-- Dynamic generation for fields with tags -->
  <div class="group-1">
    @forEach(var f in customers.fields) {
      @if(f.Visible and (f.Tag = 1)) {
        @import partials/dynamicInput { @field = @f }
      }
    }
  </div>
  <div class="group-2">
    @forEach(var f in customers.fields) {
      @if(f.Visible and (f.Tag = 2)) {
        @import partials/dynamicInput { @field = @f }
      }
    }
  </div>

  <!-- Custom section for special field handling -->
</form>
```

FireDAC

.

.

FireDAC DisplayLabel Required DataType Value

.

UI Web

FireDAC WebStencils .

:

1. -
   CRUD .
2. -
   @customers.fields.EMAIL.DisplayLabel .
   @switch
   .

11

CSS

WebStencils CSS JavaScript
”WebStencils ”
HTML .

CSS JavaScript Bootstrap
Tailwind Bulma BeerCSS CSS .

Bootstrap 5 WebStencils
CSS Tailwind CSS Tailwind CSS
.

CSS

UI

.

WebStencils HTML
HTML .

:

- .
  .
- . CSS WebStencils
  CDN .
- . Bootstrap CSS
  Tailwind Bootstrap .
- . CSS
  .

( )

CSS WebStencils :

Bootstrap

Bootstrap CSS
.

Bootstrap UI
JavaScript.

Bulma

Bulma CSS CSS JavaScript Flexbox
Bootstrap Bulma .

Bulma
.

PicoCSS

PicoCSS HTML CSS
HTML `<button>` “class="btn btn-primary"

HTML .

BeerCSS

BeerCSS Google
.

DaisyUI

DaisyUI Tailwind CSS Tailwind
.

CDN :

```
<!-- baseLayout.html -->
<!DOCTYPE html>
<html>
<head>
  <title>@page.title</title>

  <!-- Bootstrap -->
  <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bootstrap@@5.3.8/dist/css/bootstrap.min.css">

  <!-- Or Bulma -->
  <link rel="stylesheet"
```

```
        href="https://cdn.jsdelivr.net/npm/bulma@@1.0.4/css/bulma.min.css">

  <!-- Or PicoCSS -->
  <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/@@picocss/pico@@2/css/pico.min.css">

  @RenderHeader
</head>
<body>
  @RenderBody
</body>
</html>
```

> *@@* *WebStencils* *CDN URL* *@*
> .

CDN

WebStencils CSS.

Tailwind

Tailwind CSS CDN
Tailwind .

Tailwind

Tailwind CSS
:

```
<!-- Traditional CSS approach -->
<button class="btn btn-primary">Click me</button>

<!-- Tailwind utility approach -->
<button class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600">
  Click me
</button>
```

CSS
Tailwind CSS 3-4MB
.

Tailwind HTML
CSS ( ) CSS 5-50KB
.

Tailwind :

- Node.js
- npm (Node Package Manager)
- Vite Webpack
- package.json
- npm install npm run build

Web RAD Studio
JavaScript CSS.

RAD CLI

Tailwind CLI Node.js
Node.js npm .

RAD Studio Tailwind Node.js npm package.json
…:

1. CLI ( )
2. Delphi
3.

.

CSS :

:

- Tailwind CDN
-
-

-

:

- CSS
-
- (5-50KB)
- RAD Studio

CSS WebStencils :

```
<!-- baseLayout.html -->
<head>
  <title>@page.title</title>

  @if (env.debug) {
    <!-- Development: Full Tailwind via CDN -->
    <script src="https://cdn.jsdelivr.net/npm/@@tailwindcss/browser@@4"></script>
  } @else {
    <!-- Production: Optimized generated CSS -->
    <link rel="stylesheet" href="/static/css/output.css">
  }

  @RenderHeader
</head>
```

env.debug Delphi :

```
// In WebModule initialization
procedure TWebModule1.SetupEnvironmentVariables;
begin
  // env.debug is automatically available in templates
  WebStencilsEngine.AddVar('env', nil,
    function(AVar: TWebStencilsDataVar; const APropName: string;
      var AValue: string): Boolean
    begin
      if SameText(APropName, 'debug') then
      begin
        AValue := {$IFDEF DEBUG}'True'{$ELSE}'False'{$ENDIF};
        Result := True;
      end;
    end);
end;
```

> *OnValue* .

CSS Delphi Release Tailwind CLI:

:

```
tools\tailwindcss.exe -i src\input.css -o templates\static\css\output.css --minify
--content "templates\**\*.html"
```

:

- src/input.css (Tailwind)
- templates/ HTML
- CSS
- `templates/static/css/output.css`

CDN.

:

src/input.css (Tailwind):

```
@import "tailwindcss";
```

tools/tailwind.config.js (Tailwind):

```
module.exports = {
```

```
    content: ["templates/**/*.html"],
    safelist: [
      // Add any dynamic classes generated in Delphi code
      'bg-red-500',
      'text-green-600'
    ]
  }
```

safelist Delphi Tailwind
HTML .

GitHub .

CSS
Bootstrap Tailwind .

WebStencils
WebStencils
.

HTML Delphi WebStencils
.

ebStencils CSS
.

Tailwind WebStencils
CDN/CLI Delphi
.

12

Docker

WebStencils

?

.

WebBroker

WebBroker

.

NGINX FastCGI

Docker IIS Apache Web .

.

WebBroker :

( )

Web .exe (Windows) (Linux)
HTTP .

NGINX FastCGI(RAD Studio 13.0)
FastCGI NGINX HTTP

.

Apache IIS/ISAPI
Apache .so IIS .dll Web
.

Docker Container

.

:

- (Standalone, FastCGI, Docker):
  .
- (Apache module, IIS/ISAPI): Web Web
  .

WebStencils .

WebBroker HTTP
Web .

RAD Studio WebBroker :

- : Linux Windows .

- : / Windows

.

HTTP (Indy TIdHTTPWebBrokerBridge) .

:

```
procedure RunServer(APort: Integer);
var
  LServer: TIdHTTPWebBrokerBridge;
begin
  LServer := TIdHTTPWebBrokerBridge.Create(nil);
  try
    LServer.DefaultPort := APort;

    // Increase connection limits for better concurrency
    LServer.MaxConnections := 0;  // 0 = unlimited (default is limited)
    LServer.ListenQueue := 500;   // Backlog queue size (default is 15)
    LServer.KeepAlive := False;   // Disable HTTP Keep-Alive for simplicity

    LServer.Active := True;
    WriteLn('Server started on port ', APort);
    WriteLn('Press Enter to stop...');
    ReadLn;

    LServer.Active := False;
  finally
    LServer.Free;
  end;
end;
```

WebBroker

MaxConnections ListenQueue KeepAlive.

HTTP

Apache IIS .

" ?"

WebBroker

.

NGINX SSL/TLS NGINX
SSL HTTP Delphi
.

” ”
.

Windows Linux WebBroker
64 (Linux PAServer)
.

(HTML CSS JavaScript)
.

Linux Windows
.

FastCGI NGINX (RAD Studio 13.0+)

RAD Studio 13.0 FastCGI NGINX WebBroker .

FastCGI Web
( ) FastCGI HTTP NGINX .

(9000) NGINX
HTTP FastCGI
WebBroker FastCGI WebBroker NGINX HTTP
.

NGINX

FastCGI NGINX :

```
# Define upstream FastCGI server
upstream fastcgi_backend {
    server 127.0.0.1:9000;
    keepalive 32;
}

server {
    listen 80;
    server_name myapp.example.com;

    location / {
        fastcgi_pass fastcgi_backend;

        # Required FastCGI parameters
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param QUERY_STRING $query_string;
        fastcgi_param REQUEST_METHOD $request_method;
        fastcgi_param CONTENT_TYPE $content_type;
        fastcgi_param CONTENT_LENGTH $content_length;
        fastcgi_param PATH_INFO $uri;
        fastcgi_param REQUEST_URI $request_uri;
        fastcgi_param SERVER_PROTOCOL $server_protocol;
        fastcgi_param REMOTE_ADDR $remote_addr;
        fastcgi_param SERVER_NAME $server_name;
        fastcgi_param SERVER_PORT $server_port;
    }
}
```

NGINX FastCGI
HTTP NGINX
.

FastCGI Windows NGINX Apache Linux
Windows NGINX
Linux IIS.

Web

Apache IIS/ISAPI Web Delphi
Web .

Apache IIS/ISAPI DLL
Web Web .

Web :

- -
- -
- (IIS) (Apache)

:

- :
- :
- :

WebStencils (TWebSessionManager)
FastCGI Apache/IIS .

Apache IIS/ISAPI :

- : Redis /
  .
- : Web
  .

Web
FastCGI( ) .

> *Redis* .

Apache IIS :

: /login http://yourserver:8080/login Windows ISAPI
http://yourserver/WebApp.dll/login URL .

: Web
.

: RAD Studio

.

CGI

WebBroker CGI(Common Gateway Interface) FastCGI .

CGI FastCGI CGI
CGI
FastCGI.

Docker

Docker
.

WebStencils Docker :

- :
- : `docker run`
- : Linux
- :

Docker

Delphi Docker Delphi
Linux
.

WebStencils Dockerfile:

```
FROM debian:13-slim
```

```
WORKDIR /app

# Copy your compiled Linux executable and resources
COPY Linux64/Release/WebStencilsDemo /app/
COPY resources /app/resources

RUN chmod +x /app/WebStencilsDemo

EXPOSE 8080

CMD ["/app/WebStencilsDemo"]
```

.

Docker

Docker GitHub

Dockerfile.

```
FROM debian:13-slim

# Install security updates and clean up in single layer
RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# Create non-root user
RUN useradd -m -u 1001 appuser && \
    mkdir -p /app/logs /app/data /app/backup && \
    chown -R appuser:appuser /app

WORKDIR /app

# Copy application and set permissions
COPY --chown=appuser:appuser Linux64/Release/WebStencilsDemo /app/WebStencilsDemo
COPY --chown=appuser:appuser ../resources /app/resources

RUN chmod +x /app/WebStencilsDemo

# Switch to non-root user
```

```
USER appuser

# Environment variables
ENV APP_LOG_PATH=/app/logs/app.log \
    APP_RESOURCES_PATH=/app/resources \
    DOCKER_CONTAINER=1 \
    DEMO_MODE=true \
    DEMO_RESET_INTERVAL=900

EXPOSE 8080

VOLUME ["/app/logs", "/app/data"]

# Health check for web service (using bash built-in /dev/tcp - no dependencies
needed)
HEALTHCHECK --interval=30s --timeout=3s --start-period=10s --retries=3 \
    CMD bash -c 'exec 3<>/dev/tcp/localhost/8080 && echo -e "GET /health
HTTP/1.1\r\nHost: localhost\r\nConnection: close\r\n\r\n" >&3 && cat <&3 | grep -q
"healthy" || exit 1'

CMD ["/app/WebStencilsDemo"]
```

:

-
- **root** (root)
-
-
- Docker/Kubernetes

Docker/Kubernetes Dockerfile
Delphi
.

RAD Studio Docker :

1. **Docker** - Docker
2. - PowerShell
3. - Docker

Docker Ctrl+Shift+F9 Delphi
Docker .

:

- WSL2 Windows
- WSL2 Docker CLI Docker Desktop
- Delphi Linux
- PAServer

:

```
# Basic run
docker run -d -p 8080:8080 --name=myapp myapp:latest

# With persistent storage
docker run -d -p 8080:8080 \
  -v /host/logs:/app/logs \
  -v /host/data:/app/data \
  --name=myapp myapp:latest
```

-v .

> *Docker*
> *Dockerfile*
> .

WebStencils Docker
:

GitHub : github.com/Embarcadero/WebStencilsDemos

:

- Dockerfile
- PowerShell

-
-
-

.

:

SSL/TLS

HTTP SSL/TLS

.

: NGINX SSL .

```
server {
    listen 443 ssl;
    server_name myapp.example.com;

    ssl_certificate /etc/letsencrypt/live/myapp.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/myapp.example.com/privkey.pem;

    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Let's Encrypt certbot .

FastCGI: NGINX SSL .

Docker: NGINX SSL .

> *NGINX* *Swag* *NPM* *NGINX* .

.

RAD Studio

LoggerPro, DataLogger, Spring4D logging QuickLogger.

Docker .

.

```
// Read from environment variables
DatabaseHost := GetEnvironmentVariable('DB_HOST');
DatabasePassword := GetEnvironmentVariable('DB_PASSWORD');
ApiKey := GetEnvironmentVariable('API_KEY');

// Or from config file
Config := TIniFile.Create(
  TPath.Combine(AppPath, 'config.ini')
);
```

.

WebBroker
FastCGI Docker WebStencils
/ .

NGINX SSL
.

Docker FastCGI Docker
FastCGI NGINX HTTP .

Web Apache IIS
.

13

RAD WebStencils

RAD Server WebStencils . RAD Server Web RAD Server.

WebStencils RAD

RAD Server WebStencils

:

WebStencils

( ) HTML RAD

.

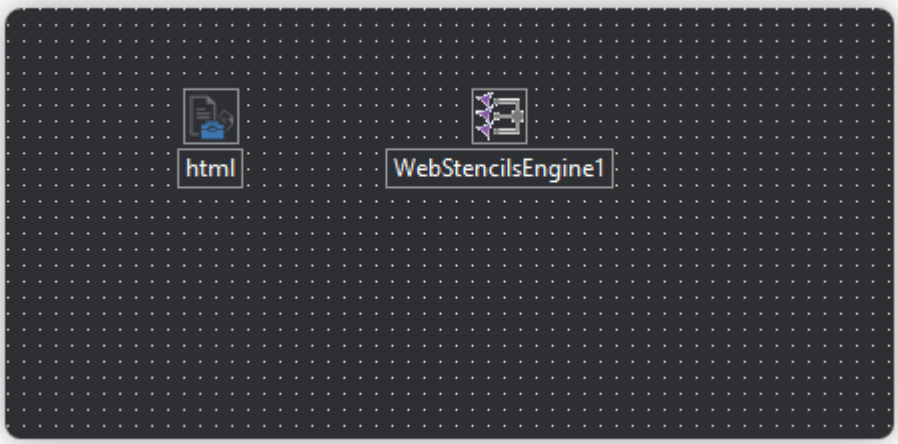
```
type
  [ResourceName('testfile')]
  TTestResource = class(TDataModule)
    [ResourceSuffix('get', './')]
    [EndpointProduce('get', 'text/html')]
    procedure Get(const AContext: TEndpointContext; const ARequest: TEndpointRequest;
const AResponse: TEndpointResponse);

...

procedure TTestResource.Get(const AContext: TEndpointContext; const ARequest:
TEndpointRequest; const AResponse: TEndpointResponse);
var
  LTemplateFile, LHTMLContent: string;
Begin
  // replace this variable with the real path to the template
  LTemplateFile := 'C:\path\to\your\file.html';
  WebStencilsProcessor.InputFileName := LTemplateFile;
  LHTMLContent := WebStencilsProcessor.Content;
  AResponse.Body.SetString(LHTMLContent);
end;
```

RAD Server URL http://localhost:8080/testfile LTemplateFile .

> *RAD* *Apache* *IIS* *RAD* .

WebStencils

TEMSFileResource TWebStencilsEngine .
HTTP . WebStencils Engine Dispatcher .



TEMSFileResource PathTemplate .

```
C:\path\to\your\templates\{filename}
```

{filename} .

{fileName} PathsTemplates.

FileResource

:

```
type
  [ResourceName('testfile')]
  TTestfileResource1 = class(TDataModule)
    [ResourceSuffix('./')]
    [ResourceSuffix('get', './{filename}')]
    [EndpointProduce('get', 'text/html')]
    html: TEMSFileResource;
```

:

```
http://localhost:8080/testfile/{filename}
```

{filename} PathTemplate .

TWebStencilsEngine *TEMSFileResource* *AddProcessor* *TEMSFileResource* .

:

```
AddProcessor(FileResourceResource, WebStencilsEngine1);
```

RAD

RAD WebBroker REST .

GitHub WebBroker MVC RAD :

WebBroker RAD Server ( ) InterBase .

TTasks TFDConnection . RAD DataModule TFDConnection TFDManager .

WebBroker RAD .

(Actions) (EndPoint)

WebBroker Actions RAD URI .

RAD JSON .

1. **HTMX** **JSON-enc**: JSON ( ) . <script> , hx-ext="json-enc" hx-post hx-put .
2. : RAD JSON JSON .

JS CSS

TEMSFileResource (JS CSS) .

RAD Server URL - https://localhost:8080/web.

WebBroker RAD WebStencils .

14

:

WebStencils

WebBroker Docker _____ wsdemo.embarcadero.com.

.

Embarcadero

WebStencils WebStencils

.

_____

HTMX (HTMX.org)

HTMX HTMX

.

htmx.org .

- _____
- ____
- ____
-

RAD

RAD

.

_____

MVC HTMX (HTMX.org)

HTMX MVC Web .
HTMX Web MVC . Delphi
MVC .

_____

WebStencils (DocWiki)

DocWiki WebStencils .

_____

HTMX

HTML JavaScript
JavaScript
HTML CSS .

JS HTML
WebStencils .

AlpineJS

AlpineJS JavaScript HTML
DOM JavaScript Vue React
HTML AlpineJS
JavaScript .

- .
- DOM .

: AlpineJS Docs

Hyperscript

Hyperscript HTML JavaScript
Hyperscript HTML
JavaScript .

- .
- .

: Hyperscript Docs

RAD Studio!

!

www.embarcadero.com