



DelphiDay
italian conference

LE NOVITÀ DI WEBBROKER

Sessioni, FastCGI, REST, WiRL



LUCA MINUTI



luca.minuti.it



luca.minuti@gmail.com



dev.to/lminuti



github.com/lminuti



linkedin.com/in/luca.minuti/



DelphiDay
italian conference

19 Novembre 2025
Padova



wintech
italia

OPEN-SOURCE PROJECTS

github.com/lminuti

WiRL

github.com/delphi-blocks/WiRL

Delphi SAML

github.com/EtheaDev/Delphi-SAML

OpenSSL

github.com/lminuti/Delphi-OpenSSL

MCPCConnect

github.com/delphi-blocks/MCPCConnect



19 Novembre 2025
Padova





AGENDA

1. Introduzione WebBroker
2. Novità di Delphi 13
 - a. FastCGI
 - b. Logging
 - c. Sessioni
 - d. WebStencils
3. Integrazione con WiRL



WEBBROKER

1



PANORAMICA

- Prima tecnologia per applicazioni Web (Delphi 3)
- Permette di scrivere server HTTP usando un approccio RAD
- Multipiattaforma:
 - Linux, Windows, MacOS
- Server agnostic:
 - Apache, nginx, ISAPI, Standalone (CGI, FastCGI, Reverse Proxy)



INTRODUZIONE

- TWebModule: “entry point” dell’applicazione. Da qui è possibile smistare le chiamate HTTP tramite “WebActionItem”
- IWebDispatch: permette di collegare oggetti per gestire specifiche chiamate
- TPageProducer/TWebStencilProcessor: crea la risposta HTML tramite template

demo time

```
procedure TWebModule2.WebModule2DefaultHandlerAction(Sender: TObject;  
    Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);  
begin  
    Response.Content :=  
        '<html>' +  
        '<head><title>Web Server Application</title></head>' +  
        '<body>Web Server Application</body>' +  
        '</html>';  
end;
```


demo time





NOVITÀ

2



FASTCGI

- A differenza di CGI, FastCGI mantiene i processi dell'applicazione **sempre attivi** e li riutilizza per gestire richieste multiple
- La comunicazione tra WebServer e FastCGI avviene tramite socket (TCP o Unix) con un flusso di **dati binario**
- I processi sono pronti ad accettare nuove richieste immediatamente (attenzione ai threads!)



FLUSSO DEI DATI

- Arriva una richiesta HTTP al web server (es. Nginx, Apache)
- Il server identifica che serve un'applicazione FastCGI
- Inoltra la richiesta a un processo worker disponibile tramite protocollo FastCGI



FLUSSO DEI DATI

Wireshark · Segui flusso TCP (tcp.stream eq 1) · Adapter for loopback traffic capture

```
.....QUERY_STRING..REQUEST_METHODGET..CONTENT_TYPE..CONTENT_LENGTH..SCRIPT_NAME/fcgi..REQUEST_URI/fcgi..DOCUMENT_URI/
fcgi
.DOCUMENT_ROOTc:\nginx-1.29.3/html..SERVER_PROTOCOLHTTP/1.1..REQUEST_SCHEMEhttp..GATEWAY_INTERFACECGI/1.1..SERVER_SOFTWAREnginx/1.29.3.
REMOTE_ADDR127.0.0.1..REMOTE_PORT62604.      SERVER_ADDR127.0.0.1..SERVER_PORT1212.      SERVER_NAMElocalhost..REDIRECT_STATUS200..SCRIPT_
NAME/fcgi  .PATH_INFO..SCRIPT_FILENAMEdummy..PATH_TRANSLATEDc:\nginx-1.29.3/html      .HTTP_HOSTlocalhost:1212.
HTTP_CONNECTIONkeep-alive.      HTTP_CACHE_CONTROLmax-age=0.AHTTP_SEC_CH_UA"Chromium";v="142", "Google Chrome";v="142", "Not_A Brand";v="
99"..HTTP_SEC_CH_UA_MOBILE?0.      HTTP_SEC_CH_UA_PLATFORM"Windows"..HTTP_UPGRADE_INSECURE_REQUESTS1.oHTTP_USER_AGENTMozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36....HTTP_ACCEPTtext/html,application/xhtml+xml,app
lication/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7..HTTP_SEC_FETCH_SITENone..HTTP_SEC_FE
TCH_MODENavigate..HTTP_SEC_FETCH_USER?1..HTTP_SEC_FETCH_DESTdocument..HTTP_ACCEPT_ENCODINGgzip, deflate, br, zstd.#HTTP_ACCEPT_LANGUAGEit,
en;q=0.9,it-IT;q=0.8,en-US;q=0.7.....
.....&..[FastCGIDemo:info] Test web broker: "".....I..Status: 200 OK
Content-Type: text/html
Content-Length: 98
Content:

.....b..<html><head><title>Web Server Application</title></head><body>Web Server Application</body></html>.....
```

Pacchetto 6. 1 pacchetto client, 5 pacchetti server, 1 turno. Fai clic per selezionare.

Conversazione intera (1505 bytes) Mostra come ASCII Nessun delta dei temp Flusso 1

Trova: ☐ Distingui maiuscole



VANTAGGI

- Performance: Nessun overhead di avvio/terminazione processi
- Scalabilità: Pool di processi configurabile
- Isolamento: Processi separati dal web server
- Indipendenza dal WebServer: come CGI ma a differenza di ISAPI o moduli apache
- Protocollo binario ottimizzato (rispetto a Reverse Proxy)

Configurazione nginx

```
location ~ ^/fcgi(.*)$ {  
    include fastcgi_params;  
  
    fastcgi_param SCRIPT_NAME /fcgi;  
    fastcgi_param PATH_INFO    $1;  
    fastcgi_param SCRIPT_FILENAME dummy;  
    fastcgi_param PATH_TRANSLATED $document_root$1;  
    fastcgi_keep_conn on;  
  
    fastcgi_pass 127.0.0.1:9000;  
}
```

demo time





LOGGING

- Nuovo metodo `TWebApplication.Log` con severity, messaggio, e parametri
 - Apache: log file di Apache, default `logs\error.log`
 - ISAPI: Windows Event Log
 - CGI: file definito da `TCGIApplication.LogFileName`
 - FastCGI: HTTP (Apache, nginx) log file del server, default `logs\error.log`

Logging

uses

```
System.SysUtils, System.Classes, Web.HTTPApp,  
Web.WebBroker;
```

```
procedure TWebModule2.WebModule2DefaultHandlerAction(Sender: TObject;  
    Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);  
begin  
    Application.Log(lsInfo, Request, 'Log: "%s"', [Request.PathInfo]);  
  
    /// ...  
end;
```

demo time





SESSIONI

- Nuovo componente TWebSessionManager, che gestisce una lista di TWebSession
- Ogni sessione, oltre al nome utente e ruoli ha la possibilità di avere dei dati custom (DataVars) visibili da WebStencil
- Di default supporta basic auth o form auth
- L'autorizzazione avviene tramite TWebAuthorizer ed è basata sui ruoli



SESSION MANAGER

- Componente non visuale
- Lista delle sessioni attive
- Configurazione principale
 - Location: cookie, header, query
 - Name: nome dell'id di session
 - Timeout: durata della sessione (default: 3600)



AUTENTICAZIONE

- L'autenticazione può essere fatta manualmente
(Request.Session.Login)
- O tramite i componenti:
 - TWebBasicAuthenticator: chiamata che usa HTTP Basic Auth
 - TWebFormsAuthenticator: POST con utente e password nel body
 - IWebAuthenticator: autenticazione custom

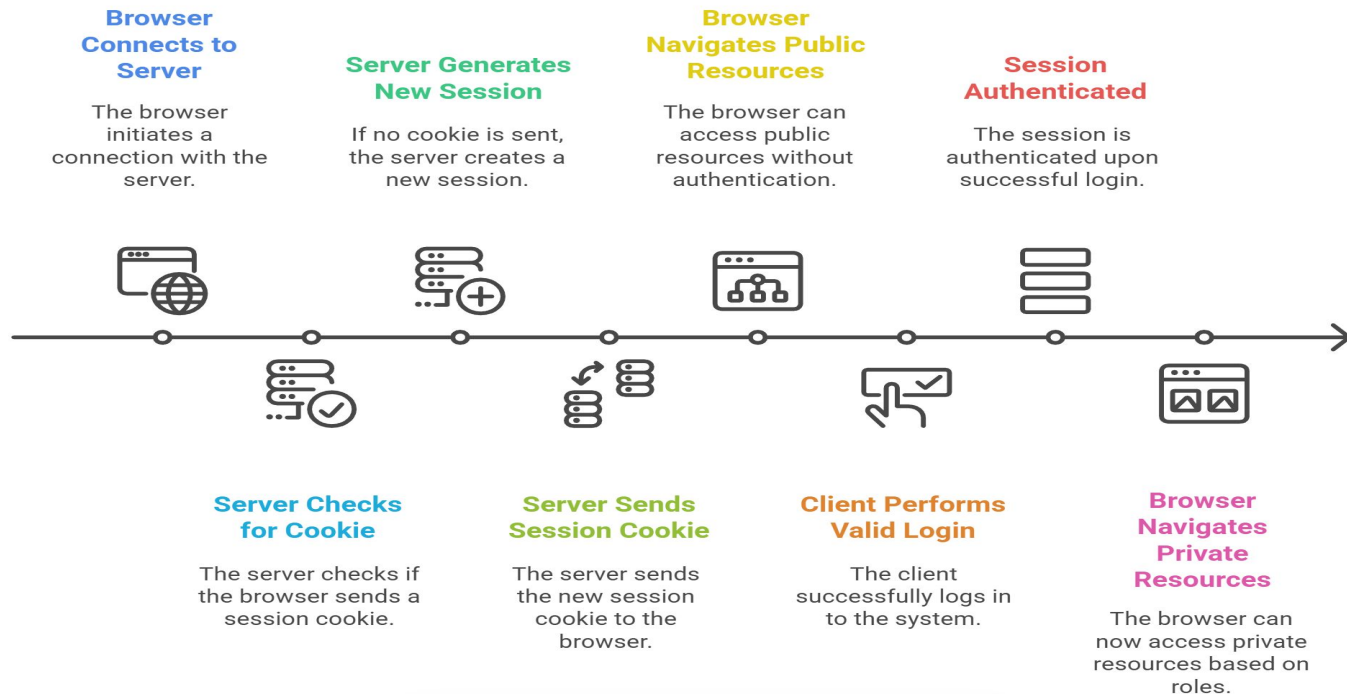


AUTORIZZAZIONE

- L'autorizzazione viene gestita da TWebAuthorizer
 - Gestione delle autorizzazioni basata ruoli definiti per “zone”
 - È possibile usare l'evento OnAuthorize per definire una logica più complessa
 - Quanto l'autorizzazione fallisce viene effettuato un redirect (UnauthorizedURL) o restituito un errore 403(forbidden)



AUTORIZZAZIONE





SESSIONE

- Alla prima chiamata viene sempre generata una sessione
- Le proprietà principali sono:
 - Authenticated
 - UserName, UserRoles, CreatedTime, AccessedTime, ...
 - DataVars: proprietà custom
 - UserHasRole()
 - Login()



RICHIESTA

- All'oggetto TWebRequest sono stati aggiunti diverse proprietà per la gestione della sessione:
 - AuthUserName: nome utente autenticato
 - AuthMethod: metodo di autenticazione usato (Basic/Form)
 - Session: la sessione corrente

demo time





WEBSTENCIL

3



NUOVI TEMPLATE

- Con l'utilizzo delle sessione adesso è possibile usare:
 - `@session.xxx`: accesso ad una proprietà della sessione
 - `@session.DataVars`: accesso alle proprietà custom
 - `@session.UserHasRole(<string>)`: verifica se l'utente ha un determinato ruolo
 - `@<n>`: accesso diretto alla DataVars con chiave `<n>`.

WebStencil

```
<tr>
  <td><strong>Session ID:</ strong></td>
  <td><code>@session.Id</ code></td>
</tr>
<tr>
  <td><strong>Authenticated:</ strong></td>
  <td><code>@session.Authenticated</ code></td>
</tr>
<tr>
  <td><strong>User Name:</ strong></td>
  <td><code>@session.UserName</ code></td>
</tr>
<tr>
  <td><strong>User Roles:</ strong></td>
  <td><code>@session.UserRoles</ code></td>
</tr>
<tr>
  <td><strong>Created Time:</ strong></td>
  <td><code>@session.CreatedTime</ code></td>
</tr>
```

WebStencil

```
@if (session.UserHasRole('admin')) {  
  <li><a class="dropdown-item" href="@env.resource/bigtable" >  
    <i class="bi bi-table me-2" ></i>Big Table Demo  
  </a></li>  
  <li><a class="dropdown-item" href="@env.resource/customers" >  
    <i class="bi bi-list-ol me-2" ></i>Customers Demo  
  </a></li>  
  <li><hr class="dropdown-divider" ></li>  
}
```

demo time





WiRL

2



MasterClass



INTEGRAZIONE

- Con l'ultima versione di WiRL è possibile integrare WiRL con WebBroker.
 - Supporto moduli Apache, IIS, CGI, FastCGI, ...
 - Integrazione con applicazioni già esistenti
 - Supporto per le sessioni WebBroker
 - Logging di WebBroker





IMPLEMENTAZIONE

- Nuovo componente TWiRLDispatcher (implementa IWebDispatch)
- Collegamento a WiRLServer
- È possibile tramite [Context] injection leggere TWebRequest, TWebResponse di WebBroker delle risorse ReST.



WIRL

```
// Configure WiRL
FWiRLServer := TWiRLServer.Create(nil);
...

// Create the dispatcher for WebBroker
FDispatcher := TWiRLDispatcher.Create(Self);
FDispatcher.Server := FWiRLServer;
```

demo time





THANK YOU