Small Case Study:

https://www.basicdatasafe.com/isapi/VMCIsapi.dll/demo/libsearch.html

This project was built in 6 hours.

A client sent me a spreadsheet with data. I wrote a quick utility to scan the spreadsheet and to get max length of data in each column with column header name (TMS FlexCell - great Excel component) . I then created an EDB2 database with the table and imported the data.

I started using EWB from its first appearance, when there was no EWB Web Server module. So I wrote my own. I had been using RealThinClient (RTC) components since their first appearance too, and used them to write a backend Delphi Web Server. Out of the box RTC provides a fully functioning demo web server - so I build upon it. In my architecture/framework, I have built several database pool modules. These database pools are paired with the RTC server I am building for a project–giving me different database backends - DBISAM, EDB and MSSQL via DevArt.

My RTC webserver was extended with functions to handle the EWB TDataset JSON API. It handles table and query metadata retrieval, and getting data and saving data via the EWB JSON API.

I have built generic extensions into the backend (RTC Web Server) framework. These extensions include a more generic data retrieval mechanism, emailing and reporting.

Here is a data retrieval example used in the demo linked above:

My framework has a table to hold SQL queries I need in the app.

Table: SysQuery
Fields: QryName, SQLStmt, GroupByStmt, OrderByStmt

In EWB app I make a request with a TDataset. Set the DataSetName to the QryName field in table.

The SQL statement is retrieved and the Where statement is dynamically generated from the params sent with the request.

I have a param naming convention made of 4 parts:

Part 1 - 1 character in length - that tells me the datatype of the variable (I could look it up on the other end - but the time to open and query table to find datatype time is way more then send the datatype - I think)

I- Integer
F-float
S-string
U-string do uppercase compares
D-date
B-boolean

Part 2 - 2 characters - basically an operator

L1-Like %X
L2-Like X%
L3-Like %X%
EQ-equal
NE-not equal
LT-less than
LE- less than or equal
GT-greater than
GE-greater than or equal
PA-parameter in the query - no part 4 when using this operator
RS-range start - no part 4 when using this operator - EDB database only
RE-range end - no part 4 when using this operator - EDB database only

Part 4 - 5 characters - Boolean logic to apply

1 - A or O - and / or
2,3 - X or B - B bracket X nothing - 1 or 2 brackets before field
4,5 - X or B - B bracket X nothing - 1 or 2 brackets after value

Part 5 – Field Name

Here is the code used in the demo to do the search:

```
 Search.Close;
 Search.Params.Clear;

 // Title
 if frmSearch.edTitle.Text > '' then
   Search.Params.Add('SL3AXXXXTITLE=' + Uppercase(frmSearch.edTitle.Text));


 // Author
 if frmSearch.edAuthor.Text > '' then
 begin
   Search.Params.Add('SL3ABBBXAUTHOR=' + Uppercase(frmSearch.edAuthor.Text));
   Search.Params.Add('SL3OXBBXAUTHOR2=' + Uppercase(frmSearch.edAuthor.Text));
   Search.Params.Add('SL3OXBBBAUTHOR3=' + Uppercase(frmSearch.edAuthor.Text));
 end;
```

```
// Subjects
if frmSearch.edSubject.Text > '' then
  Search.Params.Add('SL3AXXXXSUBJECTS=' + Uppercase(frmSearch.edSubject.Text));

// Publisher
if frmSearch.cmbPub.Text > '' then
  Search.Params.Add('UEQAXXXXPUBLISHER=' + Uppercase(frmSearch.cmbPub.Text));

// Category
if frmSearch.cmbCat.Text > '' then
  Search.Params.Add('UEQAXXXXCATEGORY=' + Uppercase(frmSearch.cmbCat.Text));

// ResourceType
if frmSearch.cmbRT.Text > '' then
  Search.Params.Add('UEQAXXXXRESOURCETYPE=' + Uppercase(frmSearch.cmbRT.Text));

Database.LoadRows(Search);
```

If all the fields are queried the backend server will generate a where statement as follows:

```
Where  TITLE Like ‘%Value%’
and ((AUTHOR LIKE ‘%VALUE%’)
or (AUTHOR2 LIKE ‘%VALUE%’)
or (AUTHOR3 LIKE ‘%VALUE%’))
and SUBJECTS LIKE ‘%VALUE%’
and PUBLISHER = ‘VALUE’
and CATEGORY =  ‘VALUE’
and RESOURCETYPE = ‘VALUE’
```

What I like about this capability is that the SQL is hidden in a backend table, but its use, is dynamic, being determined by the EWB client app.

I have built a similar convention for accessing tables via TXXXTable components in database pool.

Notes on RealThinClient components.

When writing your RTC server, you write a Provider unit that includes a Uses for one data provider unit. This gives access to the database. (I have written 3 different data provider units – but only use one per project). The Provider unit and the Database Pool's are Delphi TDataModule units that are reusable.

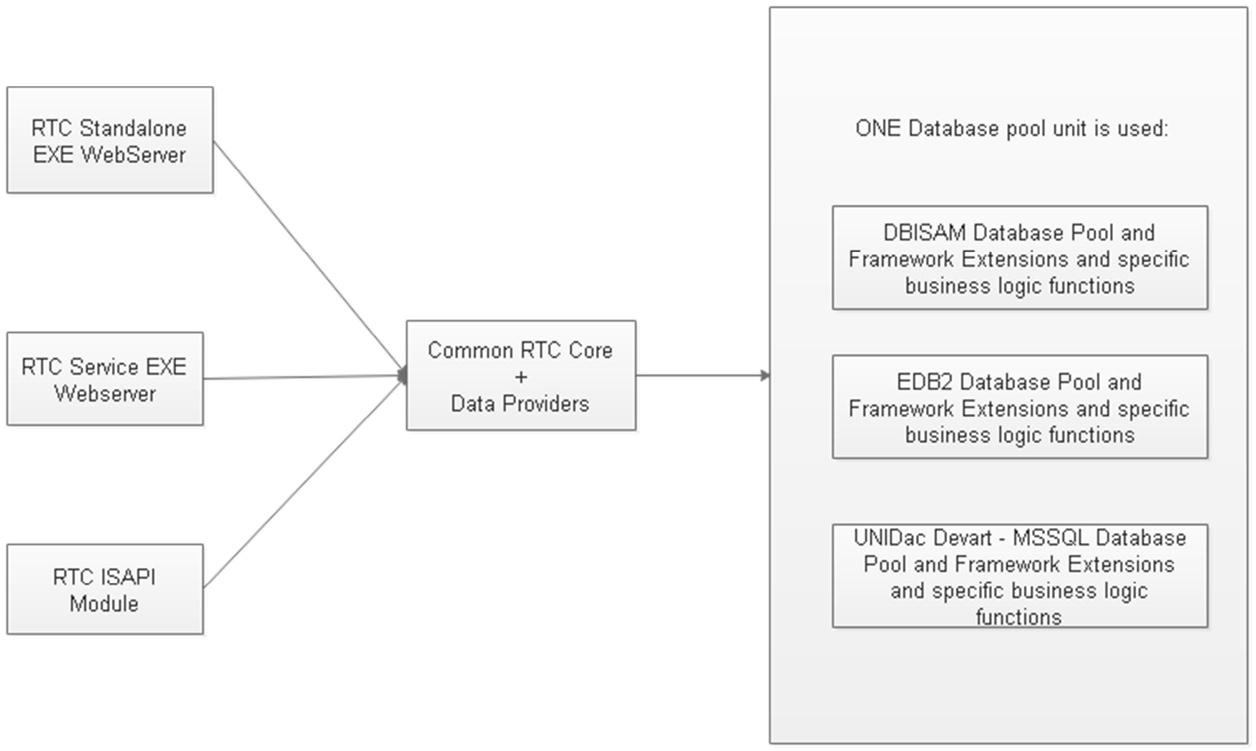
Finally – I have 3 "Servers" flavours for each project I do: a standalone, a service and an ISAPI dll. The RTC Provider (middle) is shared unchanged and reusable. Creating the 3 "servers" is quick and simple by just adding the Provider unit to the Uses. Literally about 50 lines of code per Standalone, Service or ISAPI projects – just to link to the common components in the Provider unit to the Server unit.

During development I run the standalone EXE with EWB IDE to develop, debug and test.

I can then deploy finished projects with a RTC Service exe running 24/7 or as an ISAPI dll to IIS or Apache.

With ISAPI you can deploy SSL https – if the IIS server has certificates.

RTC has a StreamSec plugin component (StreamSec components provide SSL https) which I have used to deploy the Service based Server with https access.


RTC Standalone EXE WebServer
RTC Service EXE Webserver
RTC ISAPI Module
Common RTC Core + Data Providers
ONE Database pool unit is used:
DBISAM Database Pool and Framework Extensions and specific business logic functions
EDB2 Database Pool and Framework Extensions and specific business logic functions
UNIDac Devart - MSSQL Database Pool and Framework Extensions and specific business logic functions

Elevate Web Builder

The backend RTC server was built once and is reused with small additions being added as necessary but the real workhorse is EWB. The elegance and simplicity of database handling is awesome, I would dare say unparalleled. Knit this with the responsive layout architecture of EWB and (I am bias) the grace and power of Object Pascal. This is absolutely RAD.