TUTORIAL · JANUARY 16, 2024

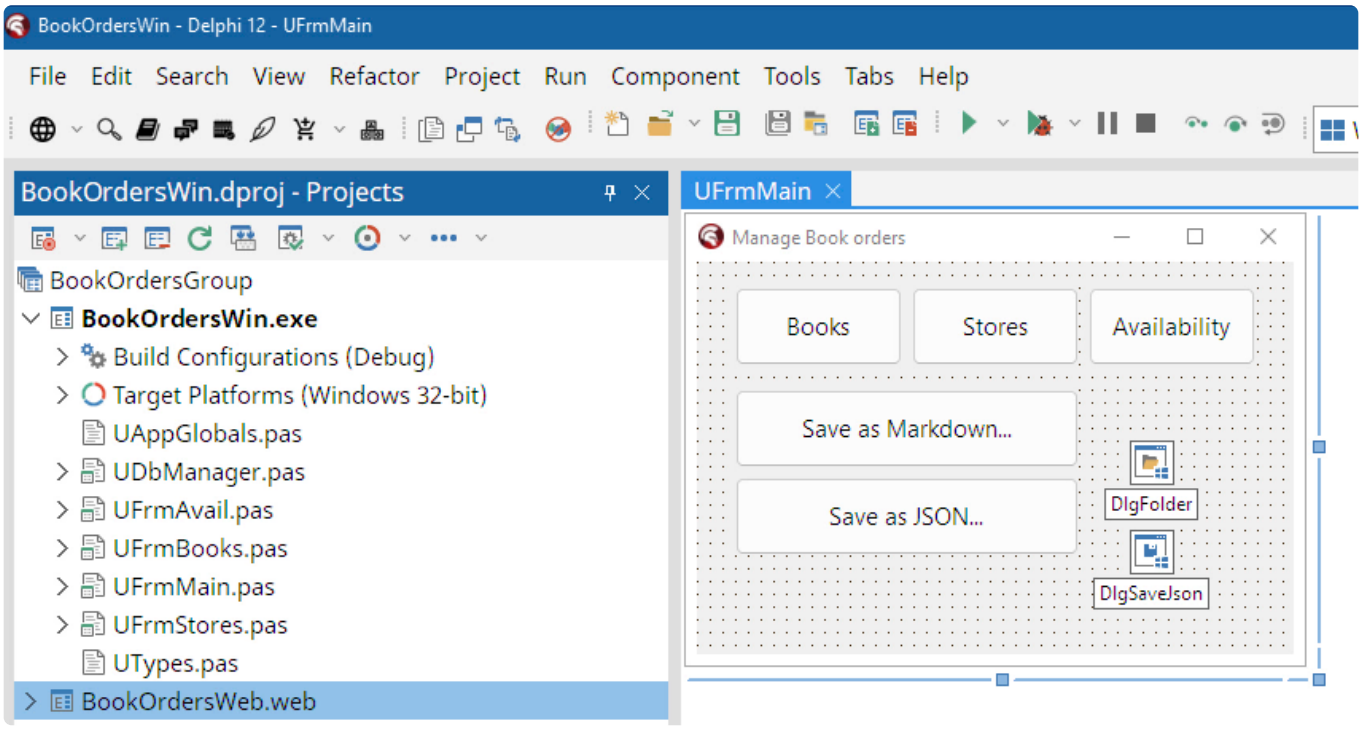
Modern Web development

Create a VCL database application with the Web in mind.

By Dr. Holger Flick

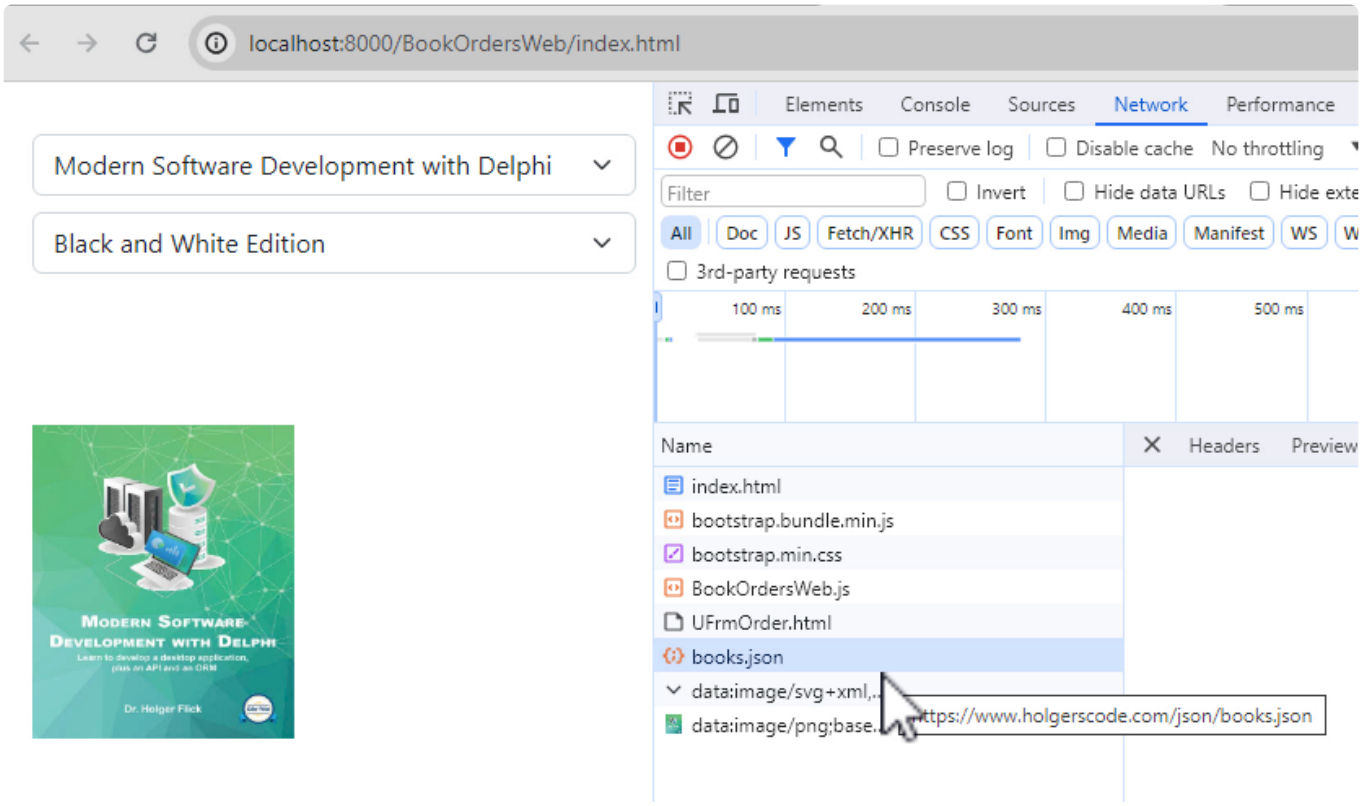
Back in May last year, I visited the town of Bruges in Belgium for TMS Days 2023. Aside from the sessions, I was able to talk to a lot of developers in person and also ask them about their feedback for *How it works with Holger*. one thing stood out: Developers love tips and tricks bundled with examples. However, it would be even more appreciated if the example spanned from the creation of the app to it being finished.

> **Source code**
> You find the source code for this example on GitHub: <https://github.com/holgerflick/hiw.bookorders>

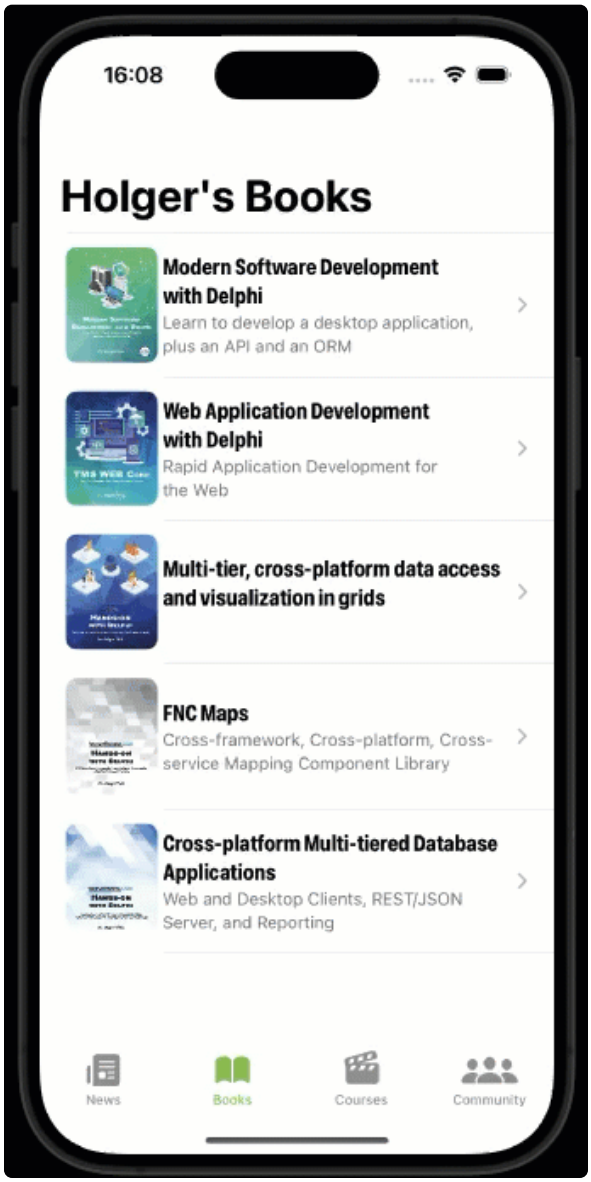


VCL application

The latest video series is the result of this feedback! It starts with creating the directories to store the new project files and ends with a modern VCL desktop application and a TMS WEB Core client that accesses data created with the application on the Web!



Web application



iOS application example

Thank you to the sponsor!

This video series is sponsored by TMS Software. Their enthusiasm for Delphi made it possible for me to meet other developers in Bruges, Belgium. Getting first-hand feedback is invaluable. It also enabled me to offer this video series on YouTube for the Delphi community to enjoy. <figure markdown> ![TMS Software-](https://www.tmssoftware.com) <figcaption></figcaption> </figure>

Outline of the series

In this **FREE 8 part tutorial series** (with **over 3 hours** of content!), I will present how to build a complete **VCL application with TMS WEB Core client** from scratch. The application is designed to organize the books I have published in recent years. I will use an **SQLite database and FireDAC** with **master-detail relationships** to

model the data. Standard Delphi database controls will be used to create the user interface. I will discuss options from TMS to build the user interface or backend with even more efficiency. As a key feature in addition to all CRUD operations, the data can be generated as **markdown documents** which can be used for static web site content generation with **modern web publishing** tools. Examples for these are MkDocs, Material for MkDocs, and Jekyll. `www.holgerscode.com` is automatically published using Material for MkDocs and will be shown as a hands-on example. However, the raw data can also be **serialized into JSON** to make it accessible for any device on any hardware. This document will be deployed to a web server and serve as the data for a TMS WEB Core client application. The Web application is designed to allow the end user to **query data interactively** in order to get more information for each book and **navigate to its order page** at Amazon.

Don't miss out on this 8 part in-depth tutorial series.

- Part 1 contains the introduction and sets the scope of the VCL application and describes the Web application.
<iframe width="560" height="315" src="https://www.youtube.com/embed/EF4re6Pm28g?si=xM-5y-2la2w-AEOm" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

- Part 2 builds the VCL application **from scratch**. Starting with the directory structure and looking at initial configuration parameters. A global application setting class as well as the database manager will be covered. Interesting implementation details that will be presented are that the former will **use class methods** and the latter will make use of the **singleton pattern**.
<iframe width="560" height="315" src="https://www.youtube.com/embed/P65pQ6xMN1Y?si=l7tq2qgja151zo-p" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

- Part 3 is focused on how to design and create an **SQLite database with FireDAC**. **It gives plenty of details for further investigation like how SQLite field types are mapped to FireDAC,** creating diagram diagrams with JetBrains DataGrip**, creating tables at run-time, and** embedding the SQLite database drivers** into your executable.
<iframe width="560" height="315" src="https://www.youtube.com/embed/EKCc9AT*sT0?si=z-jTCCS*ix8wNsHX" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

- Part 4 builds a form with all associated queries to manage books and their editions. Datasets will feature **master-detail relationships** created at design-time. Forms will be designed with access to design-time information. Fields sued will feature a mix of various data types including **images and memo fields**. Also, the massive image format and editing library **ImageEn will be introduced as a jackknife** for any image format requirements.
<iframe width="560" height="315" src="https://www.youtube.com/embed/C2-1ZC2nfj8?si=zYVrEC2qSmajlwmc" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

- Part 5 builds the two remaining forms of the VCL application. Stores that offer the books needs to be managed as well as the availability (or unavailability) of certain book editions in some countries. This will introduce you to cross-tables and how to bind you user interface using FireDAC commands.
<iframe width="560" height="315" src="https://www.youtube.com/embed/I-MIu5QumL8?si=IIi3BN7N2IpvALIS" ti-

tle="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

- In Part 6 you learn to create **Markdown documents** for database information that can be used to publish web pages. Image information will be prepared for this specific use case and referenced from Markdown. MkDocs and Material for MKDocs will be presented as an example for a Markdown dialect for technical texts.
<iframe width="560" height="315" src="https://www.youtube.com/embed/nVXb1b9MiWc?si=kv4TPRI6zeVUvHEW" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

- In order make the information in the SQLite database accessible from any client application, a **JSON document** is created in part 7. A web service is not needed in this case as the content does not change frequently. Holger does not publish books that frequently. Thus, using a static JSON file is a great solution to publish your information to the Internet without having the overhead of running another web server with services. Creation of the document will be done using the **Neon serialization framework**. **TMS FNC Core** will be presented as a cross-platform alternative.
<iframe width="560" height="315" src="https://www.youtube.com/embed/djzfeS9k4KU?si=Wjk3xDtMeh1mzsXs" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

- Finally, part 8 brings us full circle with adding a web application that can be accessed from any device anywhere with an online connection. The Web application will feature the **Bootstrap framework** to create a modern look and feel. Content will be presented in a matter that Delphi developers with VCL experience will be able to follow the creation of the Web application. It will become clear how **efficient development with TMS WEB Core** is if you have a code base in Delphi. Further, the advantages of having **type-safe, compile-time-checked, manageable Delphi source code** will be emphasized.
<iframe width="560" height="315" src="https://www.youtube.com/embed/HyX8Hl-Qe2w?si=WJaEcDTJUEbv77jN" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —