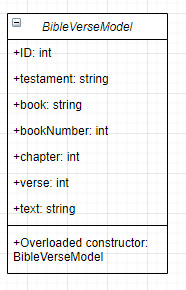
Carson Perry

Shad Sluiter CST-247  08/08/2021

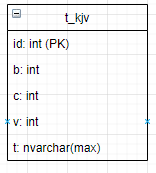
CST-247 Benchmark

I have only one model, which is the BibleVerseModel.



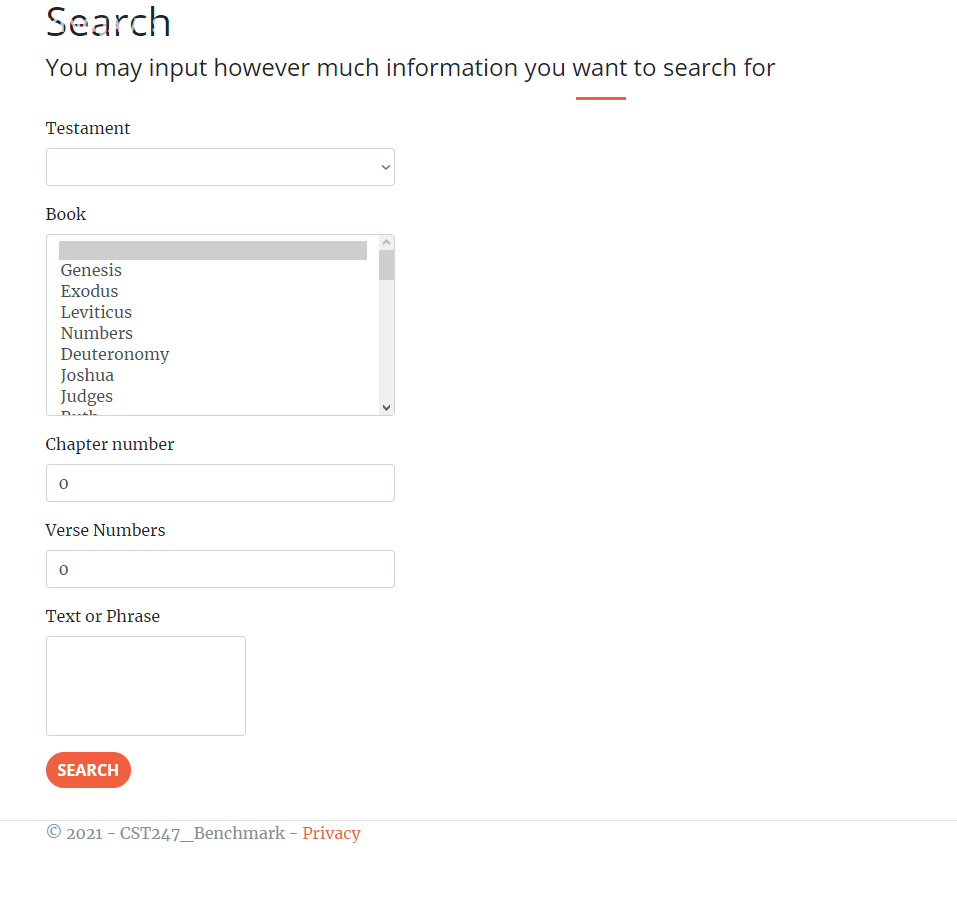
Within the overloaded constructor, it takes and ID int, book number int, chapter int, verse int, and a text string. It then sets the Testament based on the book number, as well as setting the book according to the book number. I created it this way to streamline the creation of the BibleVerseModel and requiring as little data as possible that can be extrapolated to fill out the last bit of info for a full model that’s ready to use. This way, the pages can view the verse with the name of the book, showcase which testament that verse is in, while still being able to retain book number to be used elsewhere if needed. Another fairly large technical decision, is that I decided instead of using a separate SQL search for every keyword/phrase search, I could instead use the other existing SQL statements, but then filter out the list of verses that results in those statements. It made for a much more compact piece of code with less redundancy. The final large decision is to have only one search form. This streamlines the experience for users, while still containing all functionality of searching for whatever the user wants. If the user didn’t enter in data for some of the fields, that’s fine and the processSearch method in the BibleController searches with the data they did provide to give the relevant information. For instance, the user can enter just the Old testament, and it will provide all the Old Testament books. Or they could enter in the book, a chapter number, and enter in a phrase to search for, and it will return any verses containing that key within that book and chapter.

For the database, it’s a very simple structure with a massive amount of data



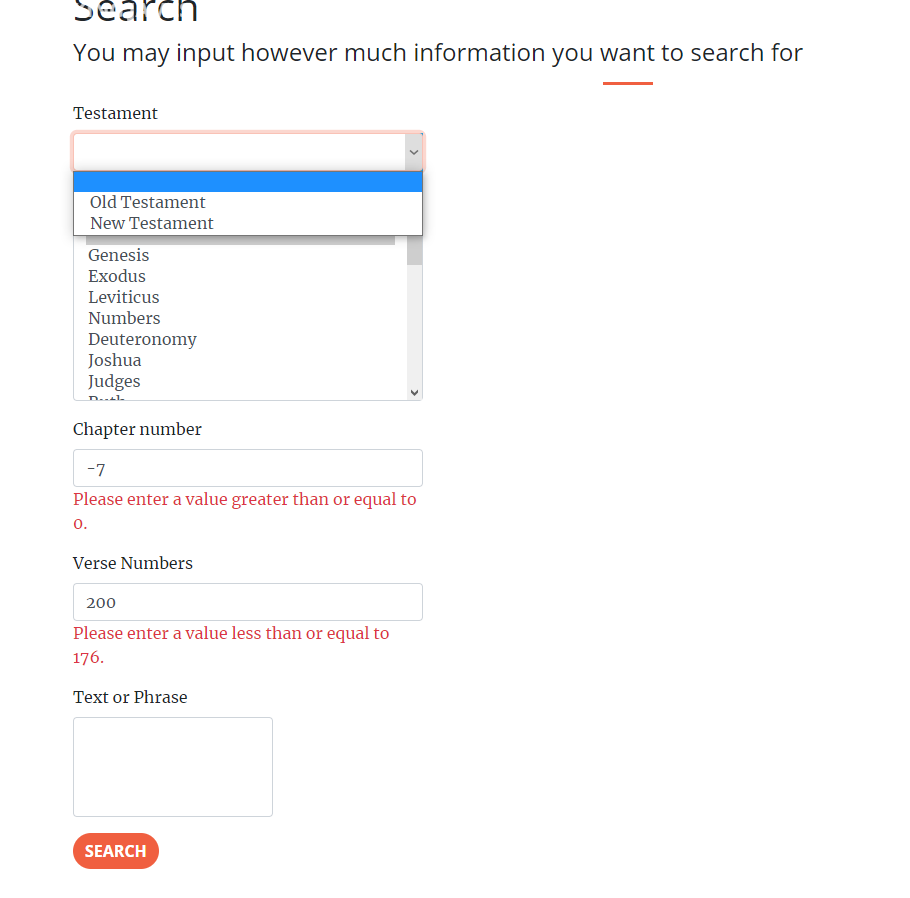
This was taken from <https://github.com/scrollmapper/bible_databases> and uses their structure. The id consists of 2 digits for the book number, followed by 3 numbers for the chapter, then 3 more numbers for the verse number instead of an auto increment id. That actually made me unable to view individual verses when clicking on them from a listed view, since ints won’t store the extra 0’s at the start of that id and it breaks the formatting. However the rest is b for book number, c for chapter, v for verse number, and t for text.

Screenshots:

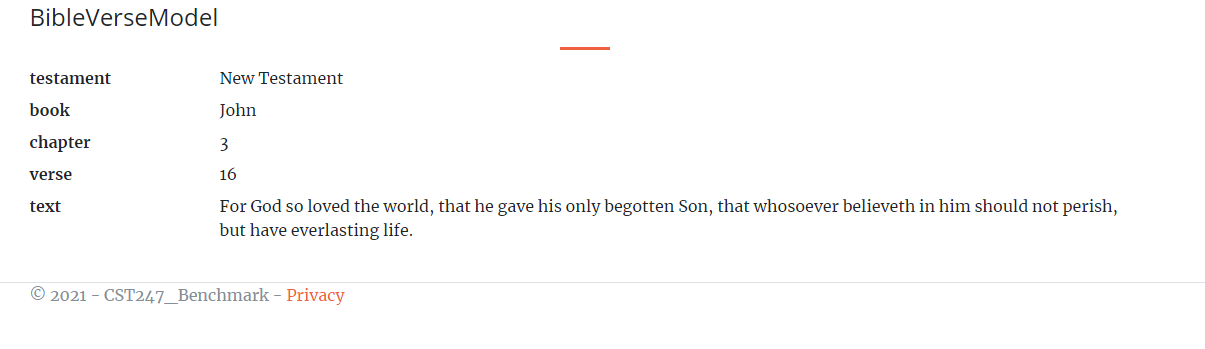


This is my one and only search form. From here, you can search in any reasonable form

This does have form validation as well:

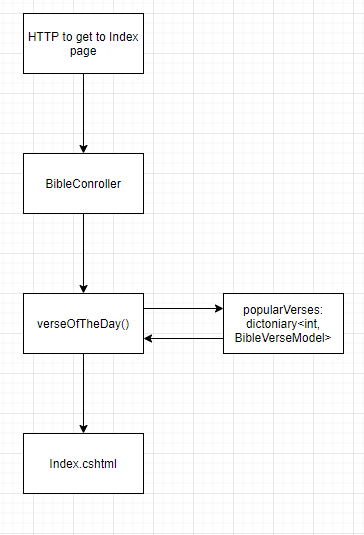


For testament and book, they’re both select fields. Testament has the default as a blank string, with Old and New Testament as options to choose if the user wishes. Then follows with the Book where there are 66 options all with their corresponding book number as the actual value. Min and max for Chapter is 0-150, and for Verse it’s 0-176. The max’s were found by doing a quick max function on the database to see the largest possible value. If the values don’t correlate with an actual verse, then the user gets sent to an error screen. The text or phrase is an input at the bottom for the user to enter in whatever string they wish to search/filter by. When searching for John 3:16, the user would choose John for the book, 3 for the chapter number, and 16 for the verse numbers and get this:



Bible Verse of the Day:

This could be implemented by manually inputting the information for the verse onto a page and change that every day. However, I am lazy, and like to automate things like that. I would create a page that retrieves the verse from a verse of the day method. This method would create a random int, that would then retrieve the verse that’s stored in a dictionary of popular verses. That way this page would never need to be updated again, and the only thing anyone would ever need to change is the dictionary it accesses, whether to add a popular verse or remove one for some reason. Another way to do this would be to incorporate BibleGateways Verse of the Day onto a page. I feel like the best place for the Verse of the day would be on the Index page before a user goes to the search page.



5. Not really sure what this prompt is wanting, but I will try my best with answering. Well, being able to provide a good product, especially those of humanitarian efforts, greatly shows the work ethic and respect of neighbors that the Bible instills in followers. Having a website that’s created poorly or doesn’t work as needed shows the sin of sloth, or laziness, and slacking off. Whether that’s the cause of the issue or not, that’s what comes to mind when obvious or prominent issues arise. Said in Colossians 3:23-24 "Whatever you do, work heartily, as for the Lord and not for men, knowing that from the Lord you will receive the inheritance as your reward. You are serving the Lord Christ." It’s against the Christian Worldview to work half-heartedly in anything you do.

Github: <https://github.com/arsonull/arsonull.github.io/tree/master/Sem4/C%23%20Remake/CST247-Benchmark>