Web Components for Database Developers

Andreas Schmidt*‡ and Tobias Münch†
* University of Applied Sciences
Karlsruhe, Germany
Email: andreas.schmidt@h-ka.de
‡ Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: andreas.schmidt@kit.edu
† Münch Ges. für IT Solutions mbH, Germany
Email: to.muench@muench-its.de

***Abstract*—We present a number of web components, which allow the presentation and modification of database content in a browser. The components include an element for displaying complete tables, or a portion thereof, a component for representing Structured Query Language (SQL) queries, and components that offer forms for creating and editing data records. In addition to presenting the functionality of the components, we also show how the components can be embedded in websites.**

***Index Terms*—Web Component; Relational-Database; Interface; Prototyping**

I. Introduction

The main purpose of the components presented here is the visual representation and manipulation of dynamic database content within websites.

Nowadays, web frameworks, such as angular or react are usually used for this purpose. These frameworks provide the experienced developer with a wide range of tools for realizing complex applications. Advantages include shorter development times, a more consistent code base, sophisticated security features and various scaling options [1].

But there are also a number of disadvantages, you have to consider. On the one hand, this concerns the complexity of the frameworks, which require a long familiarization period before they can be used productively. Due to the complexity, developers usually limit themselves to one framework, which makes them dependent to a certain extent on the continued existence of the framework [1].

These frameworks are often simply oversized when it comes to simple applications in the scientific and technical field. Here, it is often a matter of visualizing data in tabular form, searching and possibly manipulating it. In such applications, web components [2] can be a good alternative to web frameworks. Web components are a recommendation of the W3C and are now supported by all common browsers.

The structure of the abstract is as follows: First, web components are introduced in Section II, afterwards the overall architecture is presented in Section III, before examples of the implemented components are given in Section IV. This includes the integration into the HTML pages as well as partly the visual representation. The concluding Section V provides an outlook on further planned work on our components.

II. Web Components

At the heart of the web components are the custom elements, which allow the definition of user-specific HTML tags that bundle their own user interface and the associated logic.

The components inherit from the HTML-element class and implement a series of methods that are then called later when the components are added to the Document Object Model (DOM) tree.

III. Architecture of the Database web components

While the web components run in the browser, they have to communicate with a database server. The developed web components don't communicate directly with the database, but through a thin Representational State Transfer (REST)-based access layer (see Fig. 1). This service maps a logical database identifier to a specific database on the server side.

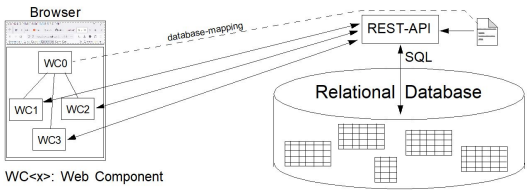

Fig. 1. Architecture of our database web components.

Beside the data, the web components also request metadata about the table from the database. The metadata is used, among other things, to construct the forms for a data set.

IV. Components

This section presents the web components we have developed so far. The Mondial [3] database was used for the examples (screenshots and SQL queries).

A. Table-component

The *db-table* component (Fig. 2) is responsible for displaying the content of a database table. It also allows navigation (scrolling) within the datasets (1), sorting by column values (2) and the formulation of additional conditions on the datasets (3).

 ISBN: 978-1-68558-138-1

First Prev 1 / 2 (15) Next Last ①

② 1 Area

Name	Code	Capital	Province	Area	Population
					< 100000 ③
Holy See	V	Vatican City	Holy See		840
Monaco	MC	Monaco	Monaco	2	31719
Nauru	NAU	Yaren	Nauru	21	10273
Tuvalu	TUV	Funafuti	Tuvalu	26	10146
San Marino	RSM	San Marino	San Marino	60	24521
Liechtenstein	FL	Vaduz	Liechtenstein	160	31122
Marshall Islands	MH	Majuro	Marshall Islands	181	58363
Saint Kitts and Nevis	KN	Basseterre	Saint Kitts and Nevis	269	41369
Grenada	WG	Saint Georges	Grenada	340	94961
Antigua and Barbudas	AG	Saint Johns	Antigua and Barbuda	440	65647

First Prev 1 / 2 (15) Next Last

Fig. 2. web component *Table*

The corresponding HTML-code is shown in Listing 1.

Listing 1. Embedding of db-table component in a web-page

```
<db-table  table="country"
           connection="mondial"
           pagesize="10">
</db-table>
```

B. Dataset-component

The *db-dataset* component is responsible for displaying a single dataset and optionally editing it. Fig. 3 shows the dataset of "France".

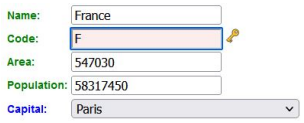

Fig. 3. web component *db-dataset*

Listing 2 shows how the dataset "France" can be embedded in a page. The parameter `key` expects the value of the primary key of the dataset. The name of the primary key attribute does not have to be specified as it is determined from the metadata of the table.

Listing 2. db-dataset component in a web-page

```
<db-dataset  table-name="country"
             key="F">
</db-dataset>
```

C. Selection-component

The *db-select* web component (Fig. 4) shows a selection box, from which values can be selected and searched via a prefix search. The values are specified by an SQL-select statement. The SQL-statement can either be specified directly by the `sql`-attribute, or it is specified using the attributes `code`, `value`, `table` and (optional) `filter`. Listing 3 gives examples of the embedding into a page.

Listing 3. Examples of *db-select* components in a web-page

```
<db-select  table="country"
            key="Code"
            value="Name">
</db-select>
```

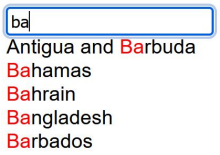

Fig. 4. web component *db-select*

D. Query-Component

The visual representation of the *db-query* component is similar to that of the *db-table* component in Fig. 2. The main difference is that no `table` parameter is specified, but an arbitrary SQL select-statement. Listing 4 shows an example in which the SQL statement determines the number of cities with more than one million inhabitants in the individual countries.

Listing 4. db-query component in a web-page

```
<db-query  sql="
      select  co.name,
              count(*) num_big_cities
        from  city ci
        join  country co
          on  co.code=ci.country
       where  ci.population > 1000000
       group  by ci.country, co.name
       order  by 2 desc"
           pagesize="10">
</db-query>
```

E. Server-component

The server component is a non-visible component in a page. It is responsible for the mapping to a concrete database on server-side. Listing 5 gives an example, how the web component is integrated inside a HTML page. The *db-server* component communicates with a RESTful service which is specified by the parameter `url`. The service, which is written in PHP, is then also responsible for mapping and accessing the specific database (specified by the parameter `database`).

Listing 5. Specification of a db-server component inside a web-page

```
<db-server
      url="https://dbkda.smiffy.de/mondial/dbwc"
      database="mondial-2010">
</db-server>
```

 ISBN: 978-1-68558-138-1

V. Conclusion and Outlook

We have implemented the first prototype for our web components and are currently in the process of expanding the components so that all meta information available in the database is also evaluated in the components. This can already be seen in Fig. 3, where a selection box with the dereferenced value (Paris) is displayed instead of the foreign key for the capital. We are also planning to extend the *db-dataset* component so that it not only provides its own form, but can also handle external forms. This will make it possible to completely customize the layout to your preferences or requirements. Further work concerns aspects of security, such as authentication and authorization.

References

[1] G. Juste. Exploring the Advantages and Disadvantages of Incorporating Frameworks into Web Development. https://www.linkedin.com/pulse/exploring-advantages-disadvantages-incorporating-frameworks-juste/. Last accessed 06.02.2024.

[2] https://www.webcomponents.org/specs. Last accessed 06.02.2024.

[3] W. May. Information Extraction and Integration with Florid: The Mondial Case Study, Universität Freiburg, Institut für Informatik, 1999, http://dbis.informatik.uni-goettingen.de/Mondial. Last accessed 06.02.2024.

 ISBN: 978-1-68558-138-1