Optimizing Data Handling in an Existing SAPUI5 Application: Transitioning from Static to Dynamic Data Implementation via Data Binding

Duale Hochschule Baden-Württemberg Mosbach

Brian Maina Nyawira (4525531)

Lecturer: Alan Ritchie

Course: Scientific writing 2025

29 June 2025

1 Introduction

In modern web development, processing data to serve it to the end user is a key ingredient to software success. There are different approaches to handling data in an application; however, this paper focuses on data binding. This research was conducted on an SAPUI5 application created as an SAP Web Components viewer. It was built using UI5 Web Components, an open-source, framework-agnostic UI Components library. According to Obert and Buzek, SAP's UI development toolkit for HTML5, or SAPUI5, is a primarily JavaScript-based library used by developers to build web-based applications with a modern and responsive UI (Obert and Buzek). Moreover, recent developments in UI5 have introduced TypeScript as an alternative to cater to developers who prefer it. The Web Components viewer used static data hardcoded into different views, which, in the long term, would limit the application's scalability. This research aims to investigate the optimization of data handling in the viewer by employing data binding technology.

Web applications running on SAPUI5 utilize a Model-View-Controller (MVC) architecture, in which the view represents the user interface, and the controller houses the code for the business logic used in processing the data. Deacon describes the application model as the object that knows views exist and that those views need some way of obtaining information and notification (Deacon). This research strives to address the use of dynamic data handling in this architecture, yielding an investigation into how it connects the view, the model, and the controller in a UI5 application, which parallels Özdikililer's statement that data binding is a connector between user interface and business logic, between destination and source (Özdikililer). Rustagi similarly emphasizes the role of MVC in structuring web applications (Rustagi).

How the data flows between the view and model can be categorized into three modes: unidirectional, one-time, and bidirectional data binding. The latter indicates that changes made in the view reflect on the model and the database, and changes made in the controller are passed to the model and view. In contrast, unidirectional or one-way data binding means that changes can only be made from the model to the view and not vice versa (**SAP2024c**). One-time data binding means the values for the view are read only once from the model.

Özdikililer notes that data binding primarily allows for data matching and manipulation in documents from Java, JavaScript, XML, and other languages (Özdikililer). Abiteboul et al. observe that XML transmits structured data as it describes textual formats (Abiteboul et al.). XML is a popular semi-structured data model characterized by a dynamic scheme not constrained by a rigid structure (El-Dahshan et al.). However, Özdikililer adds that using XML will not solve the problem of efficiently extracting the required portion of the underlying data (Özdikililer).

As UI5 uses XML and JavaScript, it creates an AJAX (Asynchronous JavaScript and XML) application

that sends data as raw XML. Benson and Grieve state that by using XML, instead of the server constructing HTML pages to send to the client, the server sends the data needed by the client as raw XML (Benson and Grieve). XML has features unnecessary for data transfer between client and server, which led developers to devise a simpler solution: JavaScript Object Notation (JSON). When creating the model for data binding, most developers prefer JSON due to its robustness and close link with JavaScript. JSON's method of declaring values is inherent to JavaScript, so developers do not need to code a parser to interpret the data, which aligns with Benson and Grieve's observation that programmers can address the data as native JavaScript objects (Benson and Grieve). As established, UI5 applications' views are created using XML, necessitating a data handling technology.

The preferred technology is data binding. Özdikililer states that data binding's undisputed contribution is the ease of data manipulation from a single point and the absence of deficiencies in the update process (Özdikililer). As a result, data is easily edited by the admin or user with editing rights, as the information is associated with a specific object class accessible from any part of the web page.

It is worth noting that the ease of editing comes with a flaw, requiring greater vigilance and better security measures. Özdikililer notes that SQL injection errors are often ignored (Özdikililer). Akram and Meredith add that the chosen binding and validation style also dictates the reliability of data parsing, which is especially important when parsing complex security constructs associated with web service security specifications (Akram and Meredith). Fortunately, UI5, as a frontend technology, is well-equipped to mitigate security risks by leveraging client-side logic and delegating backend operations, such as query execution and data processing, to server-side technologies. UI5 also provides a robust framework for binding and validation, allowing developers to implement strict data validation rules and ensure client data adheres to expected formats before being sent to the server. Additionally, the UI5 framework supports secure communication practices, such as HTTPS, enabling encrypted data transmission between client and server.

Balasubramanian explains that UI5 supports different types of data binding: element binding, property binding, aggregation binding, and expression binding (Balasubramanian). Element binding, also referred to as context binding, allows binding elements to a specific object in the model, creating a binding context and enabling relative binding within the control and all its children, which is especially helpful in list-detail scenarios (**SAP2024b**). Property binding involves binding individual properties of a control to a property of the model, allowing data from the model to be displayed in UI controls. However, one can only bind control properties to model properties of a matching type or use a formatter or data type to parse and convert the data as needed.

Aggregation binding, or list binding, can be used to automatically create child controls according to the model, either by cloning a template control or using a factory function (**SAP2024b**). The latter involves

defining a function that returns a control when called for items, whereas cloning a template refers to defining a template control that will be cloned multiple times to display items in a collection.

Expression binding is an enhancement of the SAPUI5 binding syntax, allowing expressions to be provided instead of custom formatter functions (**SAP2024b**). This type of binding saves the overhead of defining a function and is useful for manipulating data before passing it to UI controls.

In conclusion, this paper delves into the use of data binding to replace static data, ensuring software scalability and proper execution of best practices in handling data for UI5 applications.

Works Cited

blx@hook@bibinit

Abiteboul, S., et al. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

Akram, A., and D. Meredith. *Securing Web Services: Practical Usage of Standards and Specifications*. IGI Global, 2007.

Balasubramanian. SAPUI5 Data Binding: A Comprehensive Guide. 2023, Accessed 22 June 2025.

Benson, T., and G. Grieve. *Principles of Health Interoperability FHIR, HL7 and SNOMED CT*. Springer, 2021.

El-Dahshan, K., et al. "Handling Big Data in Relational Database Management Systems". *Computers, Materials and Continua*, vol. 72, no. 3, 2022, pp. 5149–64. https://doi.org/10.32604/cmc.2022.028326.

Deacon, John. Model-View-Controller Architecture. 2013, Accessed 22 June 2025.

Obert, M., and V. Buzek. *SAP UI Frameworks for Enterprise Developers*. Apress, 2023.

Özdikililer, E. "Data Binding in Front End for Web Applications". *The Journal of CIEES*, vol. 1, no. 2, 2021, pp. 31–34. https://doi.org/10.48149/jciees.2021.1.2.6.