Journal of Technological Innovations
Est. 2020

# Intelligent Page Layout Delivery: Empowering Values with Advanced Validation

**Naveen Koka**
*Email:* na.koka@outlook.com

## Abstract:

In modern software development, efficient handling of user interactions within form-based applications is essential. The proposed solution introduces a dynamic approach to page layout delivery, aiming to enhance user experience and streamline data management processes. Central to this solution is the concept of virtual page layout definition, which encapsulates crucial information regarding form objects, fields, and their configurations. By storing this information either in a database table or JSON format, developers gain flexibility in managing and customizing page layouts to meet diverse application requirements.

At the core of the runtime engine lies a structured lifecycle designed to facilitate the insertion or updating of page layouts. During the "On Load" phase, essential parameters such as record ID, object name, and user persona details are processed to initialize runtime operations effectively. Subsequently, the engine meticulously evaluates page layout configurations, ensuring alignment with the specified object and user persona. Leveraging this information, it retrieves field values from the record, enabling seamless integration of data into the designated page layout.

Upon execution of page layout actions, the runtime engine undergoes stringent validation checks to uphold data integrity. It meticulously examines predefined validation criteria, ensuring that only actions meeting these criteria proceed for evaluation. Following the execution of page layout actions, messages indicating success or failure are promptly displayed, offering users real-time feedback on the outcome of their interactions. This abstract encapsulates the essence of the proposed solution, highlighting its focus on dynamic page layout delivery, validation, and usercentric feedback mechanisms.

**Keywords:** Dynamic page layout, Web Forms, Registration Forms, Web application

## 1.INTRODUCTION

Current digital landscape, data storage plays a pivotal role in shaping the efficiency and effectiveness of various systems and applications. The optimization of data storage mechanisms holds significant implications for enhancing overall performance, scalability, and user experience. As organizations grapple with vast volumes of data, the need for codification strategies to streamline storage and retrieval processes becomes increasingly imperative. This quest for optimization extends beyond mere storage considerations, encompassing factors such as data integrity, accessibility, and security. Against the backdrop of rapidly evolving technologies and evolving user expectations, the

pursuit of enhanced data storage solutions emerges as a cornerstone of modern digital infrastructure. By delving into the intricacies of data storage codification, organizations can unlock a wealth of opportunities for improving operational efficiency, enabling seamless scalability, and fostering innovation across diverse domains. This article delves into the nuances of data storage codification, exploring its role in driving enhanced optimization and empowering organizations to navigate the complexities of modern data management effectively.

## 2. PROBLEM STATEMENT

Regular Forms are integral components in various applications, spanning from user registrations to enterprise-level solutions, often serving as the initial point of evaluation for app usability. Historically, these forms have exhibited a rigid structure, necessitating code modifications and releases for even minor adjustments such as adding or removing fields. This inflexibility not only consumes time but also introduces the risk of errors. Recognizing this pattern, application developers have sought to address these challenges by implementing dynamic layout engines, enabling the seamless addition and removal of fields without the need for code alterations or subsequent releases. However, as requirements evolve, particularly in scenarios where diverse layouts are contingent upon the values derived from selected fields, new challenges emerge.

The focal point of discussion pertains to the delivery of distinct layouts based on the values extracted from a designated field. This requirement underscores the need for an adaptive approach that accommodates varying layouts in response to dynamic data inputs, ensuring the efficacy and relevance of the displayed information. Thus, the problem statement revolves around devising mechanisms that facilitate intelligent page layout delivery, empowering applications to dynamically adjust their visual presentation in accordance with the contextual information gleaned from user inputs or system variables.

## 3. PROBLEM STATEMENT SCENARIO

Consider a scenario where a form includes a field labeled "priority," and the objective is to tailor the layout of subsequent page fields based on the priority level selected. Specifically, when the priority is designated as "High," the layout should dynamically adjust to display different sets of fields compared to when the priority is indicated as "Medium" or "Low."

This requirement highlights the necessity for a responsive layout mechanism capable of recognizing and adapting to varying priority levels. The challenge lies in implementing a system that intelligently reconfigures the page layout in real-time based on the priority value provided, ensuring that the displayed fields align with the corresponding priority level's specific needs and expectations. Thus, the problem statement underscores the importance of devising a

solution that enables seamless transitions between different layouts contingent

upon the selected priority, thereby enhancing the efficiency and user experience of form interactions.

## 4. SOLUTION

The proposed solution involves the virtual definition of page layouts, wherein the layout configurations are established conceptually rather than as static structures. At runtime, these virtual layouts are utilized to dynamically paint the page according to the specified configuration, ensuring adaptability based on input values. The runtime engine plays a pivotal role in this process, as it evaluates the provided values and subsequently renders the screen layout accordingly. This entails a continuous assessment of the data to determine the appropriate fields and their corresponding positions within the layout.

Furthermore, the runtime engine is tasked with executing designated actions associated with each field, if applicable, upon data entry. These actions may encompass validation checks, data processing tasks, or other functionalities tailored to individual fields. Importantly, the runtime engine must facilitate seamless data saving procedures, ensuring that entered information is accurately captured and stored. By implementing this solution, applications can achieve a dynamic and responsive form interaction experience, wherein page layouts are adjusted in real-time based on contextual data, enhancing usability and efficiency for end-users.

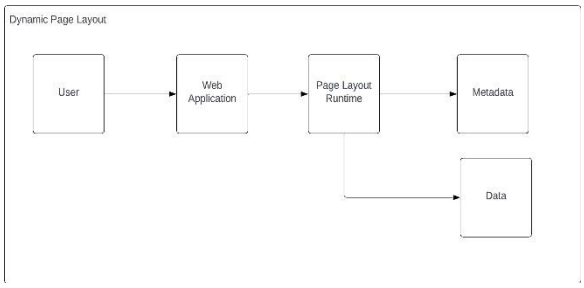


Fig 1: Dynamic Page Layout

## 5. Page Layout Keywords

Frequently encountered keywords in discussions regarding the solution are as follows: An "Object" serves as a conceptual representation of stored data, existing in a virtual capacity within the system.

"Fields" denote the conceptual definition of attributes associated with the data, forming essential components within the virtual representation.

"Criteria" refers to the expressions employed for assessing and evaluating the attributes encapsulated by the fields, aiding in decision-making processes within the system.

## 6. Virtual Page Layout Definition

The definition of a virtual page layout encompasses crucial details about the form object and its associated fields, essential for the delivery of the page layout. Presented below is a model for virtual page layout configuration, which can be stored directly in a database table or as JSON format.

On Page Layout Header: This serves as the primary configuration entity storing vital information about the page layout. By referencing the ID of this object, all necessary configuration data for the page layout can be retrieved. The header includes attributes such as Title, ID (Primary Key), Type (Read-only, Create, or Editable), Object Name (referring to the associated object), Success Message, and Failed Message. operations.

Page Layout Section: Designed to delineate various sections within the page layout, facilitating the organization of data. Each section is identified by a unique ID and encompasses attributes like Title, Page Layout Header (foreign key), and Disable Criteria, allowing for section disabling based on specific conditions.

Page Layout Field: This component defines the individual fields within the page layout, crucial for data input. Each field is assigned a unique ID and includes attributes such as Name, Page Layout Section (foreign key), Disable Criteria (criteria for field disablement),

and Hide Criteria (criteria for field hiding based on conditions). Validation Rules: The term "Validation rules" pertains to the sophisticated validation criteria mentioned earlier. It involves establishing a comprehensive set of rules that must be met prior to data storage. These rules serve as stringent criteria ensuring the accuracy, integrity, and compliance of the data being processed. By defining these rules, developers can enforce strict validation protocols, guaranteeing the reliability and consistency of the data saved within the system.

Controlling Field: A "Controlling Field" is a pivotal element dictating the rendering of the page layout, characterized by its unique identification through the object name and persona. Its attributes include an ID, serving as the primary key for individual records, and a "Value" parameter indicating the data value to evaluate and determine the appropriate page layout for rendering. Additionally, the "Persona" attribute specifies for which user persona this definition is intended. Crucially, the "Object Name" denotes the referenced object's name, while the "Field Name" signifies the specific field to observe for the designated value. Lastly, the "Page Layout Header" serves as a foreign key linking the controlling field to the corresponding page layout header, delineating which layout to display based on the evaluated value.

These components collectively form the virtual framework for page layout configuration, enabling efficient management and customization of form layouts. The structured approach allows for seamless retrieval and manipulation of configuration data, empowering developers to tailor page layouts to specific requirements. Whether stored in a database table or JSON format, this model provides a versatile solution for configuring and delivering dynamic page layouts tailored to diverse application needs.

## 7. Runtime Delivery

The runtime delivery encompasses a defined lifecycle aimed at executing insert or update actions pertinent to the page layout. For the runtime to function effectively, it necessitates essential information such as the record ID, object name, and user persona details to be provided. These details serve as crucial parameters enabling the runtime to appropriately configure its operations in accordance with the specific requirements of the page layout.

On Load During the "On Load" phase, the runtime initiates its operations by initializing the necessary configurations and settings. This phase marks the beginning of the runtime's execution cycle, wherein it prepares to handle the loading of page layouts. By leveraging the provided record ID, object name, and user persona details, the runtime strategically adapts its behavior to cater to the unique characteristics of the associated page layout, ensuring seamless rendering and interaction with the user interface.

Firstly, the runtime verifies the record ID and object name for which the data is intended to be displayed. It proceeds to retrieve the corresponding data for the identified record. Simultaneously, it retrieves the configuration settings that align with the specified object and user persona.

Next, the runtime iterates through all configurations that match the object name and the persona of the logged-in user. It then proceeds to fetch the field value from the record that corresponds to the retrieved configuration settings.

Subsequently, the runtime retrieves the page layout header information from the controlling field. It then proceeds to fetch the details of the sections and fields associated with the identified header, thus completing the process of fetching the relevant page layout configuration for rendering the data effectively. The runtime engine conducts a check to determine if the page layout sections include disable criteria. Upon identification, it proceeds with the evaluation of these criteria. Should the evaluation result in failure, indicating that the criteria are not met, the respective section is disabled as per the defined criteria.

Furthermore, the runtime engine verifies whether each page field encompasses defined criteria. It systematically assesses the presence of criteria for each field. This comprehensive validation ensures that all fields adhere to the specified criteria, maintaining consistency and integrity within the page layout configuration. Before Page Layout action Before executing any page layout action, the system initiates a validation process whereby it scrutinizes the predefined validation criteria. Upon clicking the layout action, if any validation rule fails to meet the specified criteria, the action is not evaluated further. This ensures that only actions conforming to the established validation rules proceed, maintaining the integrity and accuracy of the process.

Page Layout Action The page layout action involves retrieving all data from the form fields according to the defined page layout specifications. Subsequently, it displays messages indicating success or failure based on the outcome of the action. This process ensures that the relevant data is accurately captured and that users are promptly informed of the action's result, whether it succeeds or encounters an issue.

## 8. Enhancements

8.1 Child Object Support

We By expanding the configuration capabilities, we can introduce functionality to display both parent and child data within the page layout. This enhancement mirrors the relational structure found in databases, where primary tables are linked to related tables through foreign keys. With this feature, users gain a comprehensive view of interconnected

data sets, facilitating a deeper understanding of relationships within the system.

Furthermore, extending support for all types of actions and criteria elevates the flexibility and versatility of the configuration framework. Whether it's performing data manipulations, executing conditional logic, or enforcing validation rules, the enhanced configuration empowers developers to tailor page layouts to a wide range of use cases. This comprehensive support ensures that applications can effectively handle diverse scenarios, adapting dynamically to user interactions and system requirements.

By embracing these enhancements, we unlock new possibilities for designing intuitive and efficient user

interfaces. The ability to display parent-child data relationships and support various actions and criteria enriches the user experience while promoting scalability and maintainability in software development projects. to

8.2 Create One Object To Another

Introducing the capability to transition from one object to another presents a significant enhancement to our system's functionality. By incorporating source and target objects, users gain the ability to seamlessly navigate between related entities, expanding the scope of data interactions. Central to this enhancement is the implementation of mappings, which facilitate the transfer of values from fields within the source object to corresponding fields in the target object. This mapping mechanism serves as a powerful tool, enabling users to automate data propagation and streamline workflow processes effectively.

With this feature in place, users can harness the full potential of data connectivity within the system. Whether it involves copying customer information to associated orders, or transferring project details to related tasks, the ability to map and transfer data between objects enhances efficiency and accuracy in data management tasks. Furthermore, this feature empowers users to create dynamic workflows that adapt to changing business needs, improving productivity, and facilitating informed decisionmaking across various departments.

## 9. USES

By adhering to these principles, all form-based platforms can harness the advantages of dynamic adaptability, facilitating seamless integration of requested changes.

9.1 CRM Platforms

The These principles can serve as foundational pillars when constructing CRM platforms.

## 10. Conclusion

The discussion highlights the evolution of page layout delivery mechanisms within software applications, emphasizing the importance of dynamic and adaptable solutions. Through the introduction of virtual page layout definitions, developers gain a structured framework to configure layouts efficiently, ensuring flexibility and scalability. The integration of runtime engines further enhances user experience by orchestrating the seamless delivery of dynamic page layouts tailored to specific user inputs and system variables. Moreover, stringent validation protocols uphold data integrity, while real-time feedback mechanisms offer users instant insights into the outcome of their interactions. Expanding configuration capabilities to include parent-child data display and support for various actions and criteria enriches the functionality of the system, empowering users with comprehensive data management
capabilities. Furthermore, the introduction of objectto-object transitions with mapping capabilities signifies a significant advancement, enabling automated data propagation and streamlined workflow processes. Overall, these enhancements underscore the system's commitment to delivering intuitive and efficient user interfaces, driving operational excellence and fostering innovation across diverse domains.

## 11. References

[1] Girgensohn, Andreas & Zimmermann, Beatrix & Lee, Alison & Burns, Bart & Atwood, Michael.
(1995). Dynamic Forms: An Enhanced Interaction Abstraction Based on Forms. 362-367. 10.1007/9781-5041-2896-4_60.

[2] Seckler, Mirjam & Heinz, Silvia & Bargas-Avila, Javier & Opwis, Klaus & Tuch, Alexandre. (2014). Designing Usable Web Forms – Empirical Evaluation of Web Form Improvement Guidelines Web. Conference on Human Factors in Computing Systems - Proceedings. 10.1145/2556288.2557265.

[3] Dogdu, Erdogan & Hakimov, Sherzod & Yumusak, Semih. (2014). A Data-Model Driven Web Application Development Framework. 10.1145/2638404.2638522.

[4] Chen, Kuang & Chen, Harr & Conway, Neil & Dolan, Heather & Hellerstein, Joseph & Parikh, Tapan. (2009). Improving data quality with dynamic forms. 2009 International Conference on Information and Communication Technologies and Development,

ICTD 2009 - Proceedings. 487 - 487. 10.1109/ICTD.2009.5426738.