ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/alpesh-kanubhai-patel/289

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 1 & Iss: 2

Research Article

Navigating the Transition: Best Practices for Migrating from Salesforce Classic to Lightning Experience

Alpesh Kanubhai Patel*

Alpesh Kanubhai Patel, Information Technology, Abingdon, Harford

Citation: Patel AK. Navigating the Transition: Best Practices for Migrating from Salesforce Classic to Lightning Experience. *J Artif Intell Mach Learn & Data Sci 2023*, 1(2), 1265-1267. DOI: doi.org/10.51219/JAIMLD/alpesh-kanubhai-patel/289

Received: 02 April, 2023; **Accepted:** 18 April, 2023; **Published:** 20 April, 2023

***Corresponding author:** Alpesh Kanubhai Patel, Information Technology, Abingdon, Harford, E-mail: Alpeshkpatel24@gmail.com

ABSTRACT

The transition from Salesforce Classic to Lightning Experience marks a pivotal shift in the Salesforce ecosystem, offering more than just a refreshed user interface. This transformation introduces a modernized architectural framework, enhanced customization capabilities, and advanced integration options that fundamentally change how developers, administrators, and end-users interact with the platform. This article delves into the technical aspects of this migration, comparing the key differences between Salesforce Classic and Lightning Experience, and highlighting the benefits of adopting the component-based framework central to Lightning. It also outlines the critical steps necessary for a successful migration, including re-engineering legacy systems, optimizing performance, and implementing effective change management strategies. By addressing the challenges inherent in this transition and providing best practices, this article aims to guide organizations in fully harnessing the power of Salesforce Lightning Experience, ensuring a smooth migration and unlocking new opportunities for innovation and growth.

Keywords: Salesforce Classic, Salesforce Lightning Experience, Lightning Web Components (LWC), Aura Framework, Component-Based Framework, Data Security, Integration, Migration Process, Customization, Performance Optimization, MuleSoft, Lightning Data Service (LDS), Change Management

1. Introduction

While this is more than just moving to a new version of the Salesforce user interface, this move from Salesforce Classic is a really enormous technological change in many ways. In addition to the refreshed UI and enhanced aesthetics, Lightning Experience introduces a very different framework and set of tools that greatly revolutionize how developers, admins, and end-users will leverage this platform. This paper gives an in-depth, technical analysis of the migration process, discussing architectural differences, customization, and best practices around handling challenges that are inherent in this transition.

2. Architectural and Technological Differences

Component-Based Framework

Component-based and framework-centric-this is the crux of the experience. At the very bottom, it has a component-based framework. In contrast to the page-centric model in Salesforce Classic, it's powered by a more contemporary architecture that was centric to reusable components. Those were either developed with the older Aura framework or the newer LWC, both of which allow modular, efficient, and scalable development. LWC itself is based on modern web standards: ES6, Custom Elements, Shadow DOM, and modules, providing an improved performance and security experience.

Key Advantages

- **Reusability:** Components can be reused across different applications, reducing redundancy.
- **Isolation:** The use of Shadow DOM ensures that component styles and scripts do not interfere with the rest of the page,

enhancing security and maintainability.

- **Performance:** Native browser capabilities leveraged by LWC offer improved load times and responsiveness.

3. Data Access and Security

Data access and security are both managed through a combination of normal Salesforce security features, such as profiles, roles, and sharing rules, and new features embedded as part of the Lightning framework. For example, LDS provides an accessibility standard to access, cache, and handle records within any given Lightning component. It eliminates the need for Apex controllers, thereby making data retrieval easier and with fewer chances of security vulnerabilities.

Security Consideration

- **Locker Service:** This security layer isolates Lightning components into their own namespace, preventing cross-site scripting (XSS) and other common vulnerabilities. It enforces strict access controls on DOM manipulation and API usage.
- **Platform Events and Change Data Capture:** These features provide real-time data integration and synchronization, crucial for maintaining data consistency across different systems.

4. Advanced Customization and Integration

Salesforce Lightning Experience extends the capability for customization and integration far beyond what was possible in Salesforce Classic. The Lightning App Builder introduces a user-friendly, drag-and-drop interface for customizing pages with either standard or custom-built components. This empowers developers and administrators to tailor the user experience precisely to business needs. Additionally, a new suite of Base Lightning Components provides out-of-the-box functionality that can be extended and customized to fit specific requirements, allowing for a more personalized and efficient Salesforce environment.

Integration Improvements

Salesforce API Enhancements: In Lightning Experience, Salesforce has enhanced its REST, SOAP, and Bulk APIs, making them more robust and capable of integrating with any external systems. One significant enhancement is the addition of composite resources to the REST API. Composite resources allow developers to perform atomic transactions, meaning multiple related operations can be executed as a single unit. This reduces the number of API calls needed and ensures data consistency across transactions, improving both performance and reliability.

MuleSoft and Other Middleware: The integration landscape in Salesforce has been further bolstered by MuleSoft, a powerful middleware tool that facilitates seamless connections between Salesforce and other enterprise systems. MuleSoft enables organizations to integrate disparate systems, ensuring smooth data flows and process integrations. This is especially valuable in complex enterprise environments where data consistency and real-time data access are critical.

Technical Steps to Migrate to Lightning Experience

- ***Assessment and Analysis:*** The migration to Lightning Experience begins with a comprehensive analysis of the existing Salesforce Classic implementation. This involves several key steps:
- **Customizations and Code Review:** Identify all Visualforce pages, Apex classes, triggers, and JavaScript buttons that require re-engineering or replacement. This step ensures that all customizations are accounted for and evaluated for compatibility with Lightning.
- **Integration Points:** Map out all integration points with external systems, including APIs, middleware, and data flows. Understanding these integrations is crucial for ensuring continuity and functionality in the new environment.
- **User Profiles and Permissions:** Review existing security settings to align them with Lightning's enhanced security model. This includes reassessing permissions and profile configurations to ensure they are appropriately restrictive and meet organizational security policies.

Re-engineering and Development: Many customizations in Salesforce Classic, particularly those involving Visualforce pages and JavaScript buttons, require substantial re-engineering:

- **Visualforce to Lightning Conversion:** Visualforce pages can be wrapped within Lightning components using lightning:container, or they can be entirely rebuilt using Lightning Web Components (LWC) or Aura components. Rebuilding with LWC is recommended for better performance and future scalability.
- **JavaScript Button Replacement:** Since JavaScript buttons are not supported in Lightning, they must be replaced with alternatives like Quick Actions, Lightning Components, or Flows. These replacements provide similar functionality while leveraging the modern architecture of Lightning.

LWC Development

- **Component Life Cycle Management:** Understanding and effectively utilizing the lifecycle hooks of LWC, such as connectedCallback and renderedCallback, is crucial for initializing data, handling events, and cleaning up resources efficiently.
- **Integration with Apex:** LWC components often need to communicate with Apex controllers for server-side processing. It is essential to follow best practices, including respecting governor limits, bulkifying queries, and using asynchronous processing where appropriate, to ensure efficient and scalable solutions.

Training Users and Change Management

Effective training and change management are vital to the success of the migration:

- **Training Programs:** Develop targeted training sessions for different user groups, focusing on the new features, navigation, and workflows in Lightning Experience. This helps users become familiar with the interface and increases their productivity.
- **Change Management Strategy:** Implement a comprehensive change management plan that includes regular updates, feedback loops, and support mechanisms. This strategy should aim to minimize disruption and ease the transition for users.

Testing and Quality Assurance

Thorough testing ensures that all functionalities work as expected in the new environment:

- **Unit and Integration Testing:** Use Salesforce's testing framework to write and execute unit and integration tests for Apex code. This ensures that new components integrate seamlessly with existing logic.
- **User Acceptance Testing (UAT):** Involve a subset of end-users in testing to validate that the system meets business requirements and that users can perform their tasks effectively. UAT is crucial for identifying potential issues from an end-user perspective.

Deployment and Post-Migration Support

The deployment phase must be carefully planned to minimize business disruption:

- **Staggered Rollout:** Consider deploying the migration in phases, starting with a pilot group. This approach allows for the incremental rollout of the new experience across the organization, providing the opportunity to address issues as they arise.
- **Monitoring and Support:** After deployment, closely monitor system performance and provide ongoing support. Address any issues promptly and gather user feedback to refine and improve the user experience.

Challenges and Best Practices

Performance Optimization: While Lightning Experience offers significant performance enhancements, developers must optimize components and data access patterns to prevent latency issues:

- **Efficient Data Loading:** Utilize Lightning Data Service (LDS) and caching mechanisms to minimize server calls. Load data lazily where appropriate to improve performance.
- **Component Optimizations:** Avoid deeply nested components and ensure that each component is optimized for performance. This includes minimizing the use of heavy client-side processing and ensuring efficient rendering.

Legacy System Compatibility: Maintaining compatibility with legacy systems and processes can be challenging. A careful review of existing integrations and data flows is necessary to ensure they continue to function correctly after migration. In some cases, legacy systems may require updates or re-engineering to align with the new Lightning environment.

Continuous Learning and Adaptation: Salesforce's platform is constantly evolving, with new features and capabilities being introduced regularly. Continuous learning and adaptation are essential for keeping up with best practices and leveraging new capabilities. Staying informed about platform updates and participating in the Salesforce community can help organizations maximize their investment in Lightning Experience.

5. Conclusion

Migrating from Salesforce Classic to Lightning Experience is a complex but rewarding process. This shift involves a deep architectural transformation, enhancing scalability, security, and performance. By adopting a well-structured approach-including assessment, re-engineering, testing, and user training-organizations can navigate the migration process successfully. The transition to Lightning Experience not only future-proofs Salesforce implementations but also unlocks new opportunities for innovation and growth.

6. References

1. Salesforce. (n.d.). Lightning Experience Development Guide. Salesforce Developer Documentation.
2. Salesforce. (n.d.). Salesforce Lightning Design System (SLDS). Salesforce.
3. Salesforce. (n.d.). Locker Service Overview. Salesforce Developer Documentation.
4. MuleSoft. (n.d.). Integrating Salesforce with MuleSoft. MuleSoft Documentation.
5. Salesforce Lightning Web Components Developer Guide. Salesforce Press, 2021.
6. Gitau, M. (2020). Mastering Salesforce DevOps: A Practical Guide to Building Trust While Delivering Innovation. Apress.
7. Santoro, J. (2019). Salesforce Lightning Platform Enterprise Architecture. Packt Publishing.
8. Bonasera, R. (2021). Salesforce Lightning Platform: Advanced Features. O'Reilly Media.
9. Salesforce. (n.d.). Einstein Analytics and Discovery Developer Guide. Salesforce Developer Documentation.
10. Salesforce. (n.d.). Best Practices for Building Lightning Components. Salesforce Developer Documentation.