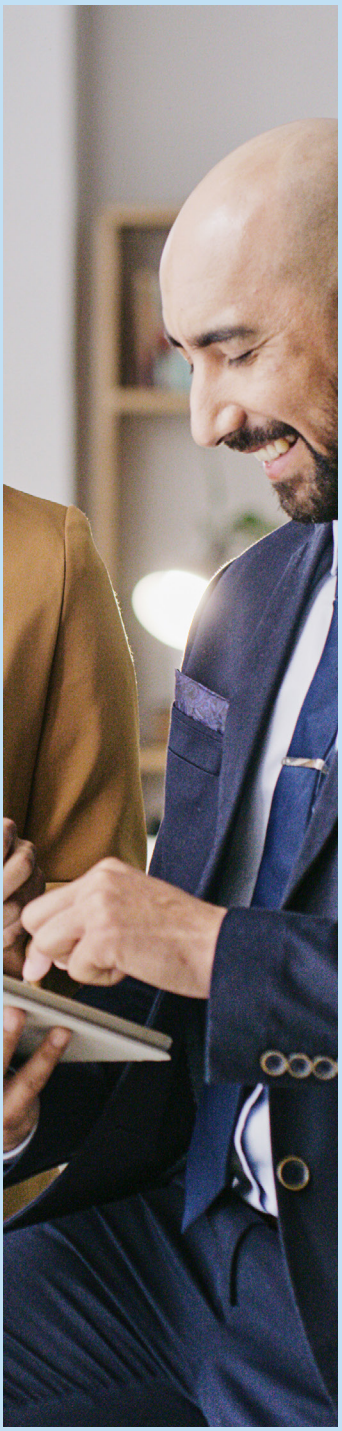
Infosys cobalt

VIEW POINT



PCF – FRAMEWORK TO A GUARANTEED TOP-NOTCH CUSTOMER EXPERIENCE

Infosys®
Navigate your next

Table of Contents

Introduction

In the rapidly changing world of app development, Microsoft offers the right foundation for simplifying the development custom components for enhanced user experience. The Microsoft Power Apps Component Framework (PCF) does not just that, but much more - whether you are a professional developer or an app creator.

The Microsoft Power Apps component framework allows professional developers and app builders to provide end users with a rich user interface experience and extend the functionality of Microsoft Dynamics 365 in addition to out-of-the-box features with visually appealing apps. One can design custom code components using PCF and it is considered a great solution to make the user's interface quicker and UI-rich. It is known to reduce the number of clicks and time spent on the form by a user.

Any developer or app builder who is familiar with the web development lifecycle and HTML web resources such as Type Script, NPM, and React can use the framework introduced in the White Paper. However, even if your work primarily involves using low-code technologies, you can still benefit from this White Paper by building canvas apps using low-code technologies.

Some examples of PCF include country flags, mobile number formats, and autofill suggestions for address searches and more. By the end of this White Paper, you will have a clear understanding of Microsoft PowerApps Control Framework (PCF) and its related technologies, how to leverage it for a customer experience, and why you should choose PCF over Custom Pages.

Why PCF?

PCF offers developers a powerful way to extend the capabilities of Microsoft PowerApps and Microsoft Dynamics 365. Here are a few reasons why it's beneficial.

Low Code

PCF allows one to create custom components with the least amount of code thereby bringing it within the reach of even the non-developers.

Consistency

PCF components can be embedded with ease inside the Microsoft PowerApps Portals interface to create rich & responsive UI controls that are better than standard Dynamics 365 OOB controls.

Performance

PCF has leverage on Framework's lifecycle and well-integrated with platform optimization features like caching and memory management.

Reusability

PCF components can be reused across the solution, which promotes both consistency and efficiency during development.

Scalability

Microsoft PowerApps Portal, backed by PCF, provides a scalable solution suitable for various business needs.

Rapid Development

PCF simplifies and shortens the development process as it allows one to quickly create and deploy custom components on-the-go.

How did we implement a PCF framework?

Step 1: Open command prompt and enter the below command statement.

pac pcf init --namespace TestPCF --name HelloWorld --template field

- <name> -------> Folder name.
- <namespace> ---> manifest file.

Step 2: By giving the below prompt, you will be able to Initiate and install all the requirement components for PCF.

npm install

- dependency added
- npm install @fluentui/react
- npm install react | npm install react@16.7.0
- npm install react react-dom

Step 3: Above steps will create a PCF template with index.ts file. This file will allow us to create a PCF component.

Step 4: .tsx file will allow developers to create reusable components.

When you create a resusable component, index.ts file will act as startup program to render any number of components

write index.ts & other ts or tsx files as per requirement. design & logics

- ReactDOM.render(React.createElement(PcfComponent1, props1), this._container);
- ReactDOM.render(React.createElement(PcfComponent2, props2), this._container);

Step 5: Code Compilation and Build. Post which it will be published on the browser.

npm run build
npm start watch

Complex PCF Controls

By creating a re-usable grid component with configurable properties for defining required columns out of whole result set.

Create reusable component that can be used as plug-n-play by providing data.

1.

Filter By

Enter value

Date ↑↓ Tran Code ↑↓ Amount ↑↓ Payee Name ↑↓ Check No ↑↓ Return Code ↑↓

2.

Filter By

Enter value

Posting Date ↑↓ Check Type ↑↓ Check No ↑↓ Amount ↑↓ SEQ No ↑↓ OutsortReasonCode ↑↓

3.

Filter By

Enter value

Token Status ↑↓ Token Id ↑↓ Exp Date ↑↓ Status TKN/PROV ↑↓ WID Name ↑↓

Address finder: We have created this generic Address finder, which allows us to search for any address by key in first few characters.

1 Meads Place, Bulls, State001, 4818

1 Mead Lane, Horopito, State002, 4696

1 Meadow Lane, Pakaraka, State003, 0472

1 Meadow Lane, Richmond, State004, 7020

1 Meadow Street, Kaipoi, State005, 7630

1 Meadow Street, Lumsden, State006, 9730

1 Mead

Multiselect: Leverate on FluentUI/ReactJS to create this user friendly multiselect.

Multi select lookup

India, Singapore

Asia

- [x] India
- [x] Singapore
- [ ] Malaysia
- [ ] Hongkong
- [ ] Indonesia
- [ ] Thailand

Europe

- [ ] Malta
- [ ] Germany
- [ ] Italy
- [ ] France

Benefits over Custom Pages

By creating a re-usable grid component with configurable properties for defining required columns out of whole result set.

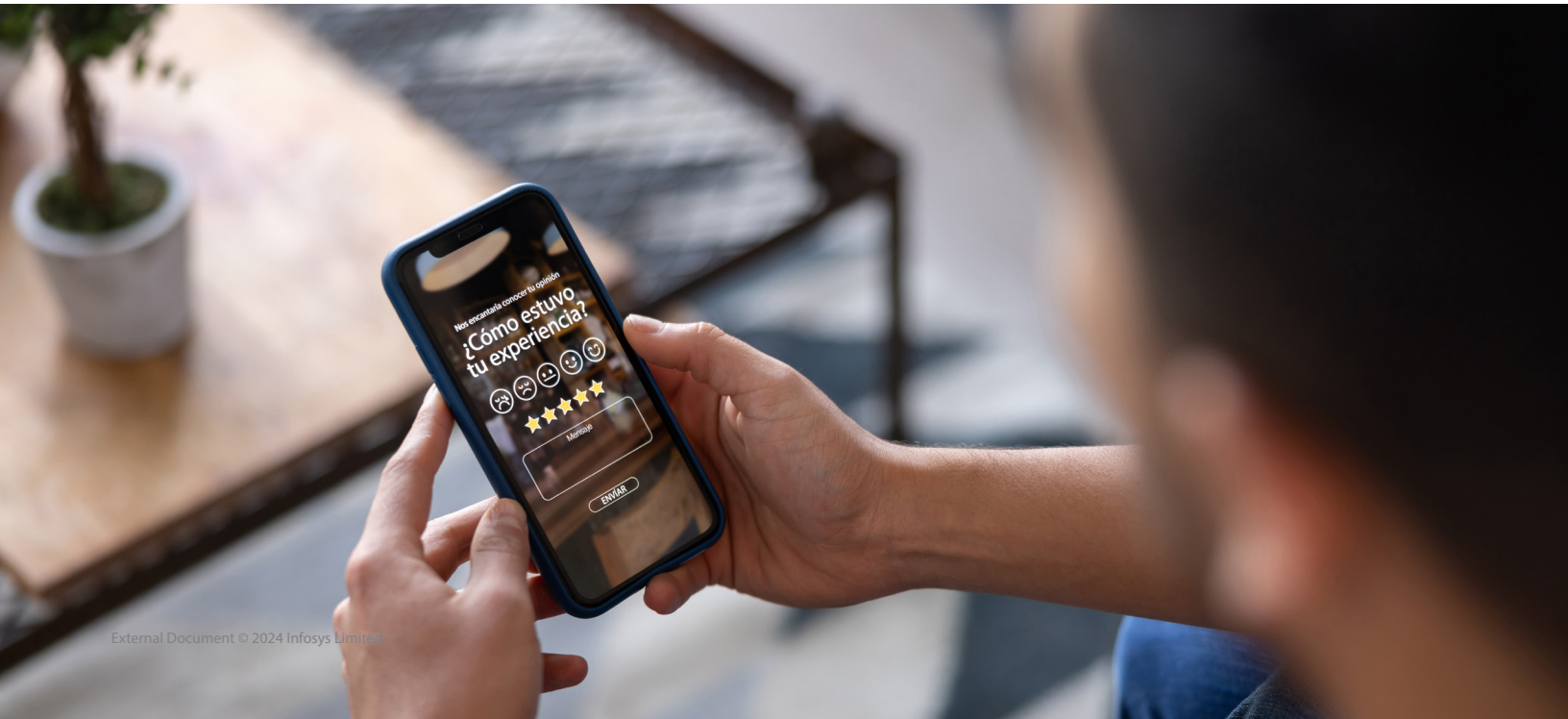
Create reusable component that can be used as plug-n-play by providing data.

Custom Pages

- Low-code platform to achieve custom screens. Lesser scope of flexibility due to limited controls to play around with.
- Take a significant amount of time to load thereby impacting performance and resulting in longer query handling times.
- The page reloads every time when one navigates to and fro. The page does not restore the datapoints and every visit seems like a new one.
- Delegation limit (data fetch) of only 2000 records.
- Custom Pages are not reusable and hence not a viable choice when deadlines are tight.
- Developers are required to share the custom page and its app components to allow another developer to make changes. This leads to slower development cycles.
- With custom pages, one cannot bundle all the files in a single solution.
- Only Microsoft Power Fx query can be used for validations.

PCFs'

- Offer more flexibility to develop the screen and a provision to control the data display.
- Faster load times thereby translating to quicker turnaround times and an improved efficiency.
- With PCFs, data is stored in cache thereby leading to faster information retrieval and minimal refresh rates.
- There is no limit in datafetch and data will get presented when we scroll down (without any additional code). This will make page memory footprint very low.
- PCF components can be reused across modules, and one does not have to reinvent the wheel from the scratch.
- Parallel development is possible while working with PCF.
- With PCF, one can bundle all files in a single solution.
- No limitation / dependency on specific javascript framework, however, it is recommended to use ReactJS with Fluent UI for best results.



Best Practices

1. Modularity

PCFs provide a plethora of modular controls that can be applied across fields, forms, and entities. The controls are simple to use and promote reusability due to their inherent capability.

2. Responsive Design

PCFs have a responsive aspect to them. One must ensure that the PCF controls are working as expected across devices of varying screen lengths.

3. Error Handling

There might be scenarios wherein a PCF does not load or there might be no records to show, or some sort of user action to show the desired results. To handle such cases, it is imperative to implement a rigorous error handling mechanism that will inform the end-users of the issue at hand and direct them to the next set of actions.

4. Testing

Conduct extensive testing in a Microsoft Dynamics 365 environment as well as locally.

5. Optimization

To achieve the best results, it is crucial to resort to code optimization. The following guarantees the same – deploy effective code, reduce needless API calls, and consider asynchronous loading when appropriate.

6. Documentation

For your PCF controls, include detailed documentation that explains how to install them, what configuration options are available, and what dependencies are needed.

7. Security

Follow the best security practices to ensure that your PCF controls don't introduce vulnerabilities. Avoid use of risky functions or unsecure handling of sensitive data.

8. Versioning

As a best practice, it is vital to keep track of changes while working with PCF, and this can be achieved via versioning.

9. Incorporate TypeScript Features

Leverage the benefits of TypeScript and its features namely - bold typing, classes, interfaces, and more, to enhance both the scalability and maintainability of the code.

10. Community and Updates

Stay abreast with the latest in the world of PCF through blogs, forums, and communities. One can also refer to Microsoft's space for a glimpse into what's new.

More on PCF

We will now discuss the related technologies that make the PCF framework one of the most workable options in the Microsoft Dynamics 365 world. Current Microsoft Dynamics 365 environments favor the Microsoft PowerApps Component Framework (PCF) for its seamless integration with multiple technologies. Using web standards, specifically HTML, CSS, JavaScript and ReactJS, it ensures compatibility with modern web development methods. Additionally, TypeScript is supported by PCF, which provides a statically typed environment for creating reliable components. Compatibility with Common Data Service (CDS) and Microsoft Power Platform makes it a smart choice for enhancing and customizing Microsoft Dynamics 365 apps.

Our experience/journey with PCF so far

While our exploration started with React standard class components, the state management using react hooks was better and simpler from a rapid development perspective compared to class components. The impact on performance is more evident when you try to scale the components to the next level and have more and more state variables. Based on these observations, we recommend the latest React functional components that support React hooks and comply with the latest React developments.

Conclusion

What we have shared is just tip of the mountain of opportunities that are made possible by PCF through its ease of customization, countless varieties of integrations feasible, while also catering to the localization, and accessibility requirements that enhance the user adoption to much higher levels.

Remember that time to market and an intuitive CX is the key to win in today's ever changing business landscape and PCF is the right fuel to throttle your way to success.

About the Authors

Pavan Kumar Channa

Principal Consultant

Pavan is a seasoned Technical Architect with over 20 years of experience in the IT industry. He excels in enhancing efficiency, productivity, and organizational effectiveness. Pavan's leadership skills are complemented by his expertise in process improvement, Software as a Service (SaaS), software project management, and requirement analysis. He is proficient in managing complex assignments and guiding cross-functional teams. His deep understanding of technology has consistently helped clients achieve their goals ahead of schedule.

Mehul Jain

Sr. Associate Consultant

Mehul is a skilled Microsoft Dynamics 365 Consultant with 4 years of experience advising stakeholders on optimal technology solutions. He excels in problem-solving and leveraging Microsoft Dynamics 365 to implement effective solutions. Known for his positive mindset and teamwork, Mehul collaborates seamlessly across the board. Proactive and versatile, he often steps into dual roles, including business analyst, ensuring smooth project execution. His technical proficiency and leadership skills enable him to suggest innovative, value-adding solutions, maintaining strong client relationships and ensuring successful day-to-day operations.

Pratik Poddar

Senior Systems Engineer

Pratik Poddar has demonstrated excellent learning capabilities and has quicky grasped a good understanding of Microsoft PowerApps Component Framework (PCF) and Custom Pages. He is highly proactive and quite accommodative in picking up work outside his scheduled plan. Pratik has grown in confidence during the course of the project and is a dependable resource. He is someone who does not shy away from challenges and has proved himself vital to the success of the project.

For more information, contact askus@infosys.com

Infosys®
Navigate your next