MVC

Java script C# Html Helper or Html Tag

مزایای استفاده:

- کاهش هزینه و زمان پیاده سازی حدود یک چهارم
- افزایش کیفیت حداقل دو برابر
- افزایش مدیریت تغییرات Change Management

معایب استفاده:

- ایجاد کنترلها در سمت سرور و کدنویسی در سمت کلاینت (معماری MVC) که به نسبت معماری ضعیف تری نسبت به معماری MVVM هست.
- استفاده از Type script بجای C#
- ایجاد زیرساخت، کاری نسبتا سخت و زمان گیر هست
- زیرساخت به نسبت سناریوهای کمتری را ساپورت می کند، و برای سناریوهای پیچیده زیرساخت کارایی خوبی ندارد.
- نیاز به تکرار کدها در سمت کلاینت و سرور (C#, Type script)
- Transfer کردن دیتا بین کلاینت و سرور

Blazor Client Side

مزایای استفاده:

- بهبود موارد قبلی(Cost & Time, Quality, Maintenance)
- سهولت در ایجاد زیرساخت
  - امکانات زبان #C در سمت کلاینت
  - پشتیبانی خیلی خوب از Component Oriented
  - معماری MVVM
- زیرساخت به نسبت سناریوهای بیشتری را ساپورت می کند، و برای سناریوهای پیچیده کارایی آن بهتر و پیچیدگی آن کمتر است.
- بکارگیری کدهای سرور در سمت کلاینت
- بدون UI Latency

معایب استفاده:

- بعلت ارسال dll ها به سمت کلاینت ممکن است زمان Load اولیه طولانی باشد
- Transfer کردن دیتا بین کلاینت و سرور دارای مشکلات زیادی است.
- نیاز به Serialize & Deserialize کردن Expression ها در سمت سرور که کاری بسیار سخت و پردردسر می باشد.
- همچنان زیرساخت نمی تواند سناریوهای زیادی را پشتیبانی نماید.

Blazor Server Side

C# Razor Component

مزایای استفاده:

- بهبود موارد قبلی(Cost & Time, Quality, Maintenance)
- سهولت در ایجاد زیرساخت
  - امکانات زبان #C در سمت کلاینت
  - پشتیبانی خیلی خوب از Component Oriented
  - معماری MVVM
  - **عدم نیاز به Serialize & Deserialize کردن Expression ها در سمت سرور**
- سهولت در یادگیری و بکارگیری
- **با استفاده از زیرساخت بسیاری از سناریوها را می توان پیاده سازی کرد، زیرساخت کارایی خود را حتی در سناریوهای پیچیده دارد، و دارای پیچیدگی نسبتا کمی می باشد.**
- بکارگیری کدهای سرور در سمت کلاینت
- بدون Load اولیه
- Full state بجای Stateless
- استفاده از Interrupt (event-action) بجای Post back
- برای پروژه هایی که مبتنی بر داده هستند (Data Centric) این روش دارای مزیت های زیادی است.
- **مشکلات Transfer کردن دیتا بین کلاینت و سرور وجود ندارد. و عموما با اعمال تغییرات در سمت سرور کلاینت بصورت اتومات بروزرسانی می شود.**

معایب استفاده:

- UI Latency: در صورت پیاده سازی صحیح (عدم درخواست های مکرر به دیتابیس) UI Latency قابل اقماض هست ولی در غیراینصورت سیستم ممکن است دارای UI Latency باشد.
- نیاز به اتصال دائم به اینترنت: برای پروژهایی که حتی در زمان قطع اینترنت باید کار کنند استفاده از این روش توصیه نمی شود.
- بعلت انجام عملیات در سمت سرور سناریوهایی که مبتنی بر Mouse Move می باشند، مانند Drag & Drop این روش از Performance خوبی برخوردار نیست. و برای این سناریوها بهتر است از کدهای کلاینت ساید استفاده کرد.