Architectural issues with AJAX requests and user journeys

Posted At : 13 October 2011 14:58 | Posted By : Shaun McCran
Related Categories: Best practices, AJAX

The more I use non standard ways of loading data and online content the more issues I uncover with how users can interact with a platform.

Adopting new technologies to build applications is all well and good but what implications does this bring to bear on your users? How do they apply their normal usage habits to your new and differently architected application?

There are a slew of existing technologies, and new ones emerging all the time that allow developers to break out of the traditional flow of an application or website.

Think about how things were ten years ago. Users loaded pages in sequential fashion. This was a clearly defined sequence from both a user journey point of view and from a technology point of view. If you looked in a directory on the server you could probably quite clearly see all the pages that made up a journey. They were probably entirely encapsulated in one system routine that included related files such as CSS and images, but essentially each pages was standalone. This sort of system helped users form browsing habits, such as using the back button and bookmarking individual pages.

Now skip forward ten years to now. Technologies like AJAX and JQuery allow a huge variety of display states and page loading methods. It is now a very simple task to inject dynamic content from a variety of sources into your pages, completely removing them from the normal flow of a traditional application.

Similarly there are major differences in back end data integration options. Systems like remote WebServices and Cloud based data layers that allow for remote data to be held elsewhere and not persist in any real way on your platform.

These technologies are viewed as new and dynamic ways of building applications by technical experts. But the biggest danger with them is that they fundamentally contradict how users have learnt to use the internet, and by association your applications.

As an example case it is now possible to build an entire application that triggers no new page impressions in a browser at all. This example application does not use its own data but pulls statistics from an Amazon based cloud service. But the Cloud service isn't a static database, it is constantly being updated and evolving with use. The problem's this example faces are many, the user cannot effective bookmark any specific point in the application. They cannot jump back into it at a desired point. Also if they perform any calculations or arrive at any recommendations how do you retrieve them? There is no local storage and you cannot count on the data being in the same state when you next visit.

Building applications in a traditional fashion means that you inherit the usability and standardisation that users are accustomed to from modern browsers. Removing the 'comfort blanket' of a browsers standard controls can seriously impact on the perceived usability of an application. I've been encountering some of these perceived issues and over the next few weeks I hope to explore and document how you can overcome them.