Using XML and XAML to build better user interfaces for applications

User Rating: / 0
Poor Best Rate

M. Ryan Corbin

Article Review

Source: Using XML Files to Document the User Interfaces of Applications

The face of any software application is referred to as the UI or user interface. It is said that a good UI is the difference between good software and bad software. I believe this to be more than a true comment and more of mission statement for any programmers. After all, how useful is an application when it is too hard or confusing to use?

Inside the article "Using XML Files to Document the User Interfaces of Applications", the authors Mohammad Tubishat, Izzat Alsmadi, and Mohammed Al-Kabi describe particularly helpful insight into software design techniques in regards to saving and preserving the elements of the graphical user interface. The overall idea is to create a way to recreate the GUI (graphical user interface) states from one application to another.

While reading I thought the idea to be somewhat redundant in my personal experiences until I looked back at my previous course work. In my company we use programming elements and design tools to help generate a lot of the GUI. The project tool, like Microsoft Visual Studio for example, has built in tools where you can easily drag and drop UI elements into you project and save the GUI state as a template. This becomes incredibly handy when creating the other parts of your application as you can refer to the saved template files to rebuild the interface. In further reading of the article It becomes clear that this might not be an option for everyone. While it is simple for us to use a project management application suite how do you solve the problem of preserving the GUI state from our application to a completely different one? What do you do if you are designing an application from scratch?

The author's answer is to simply translate the GUI state into XML. XML has long being the universal language understood by every program and programmer. It's simple to create and easy to understand. Beyond XML for simple translation it also becomes a way to easily record changes in your development. Changing one area of an application could cause a domino effect throughout. The use of XML to keep track of each design state at every change allows for an easy "redo" if needed.

XML as the universal language of edits is not a new idea. The authors are actually proposing a different way to look at XML to keep track of the states. They give the example of a simple program in Windows like Notepad. Notepad has around 200 controls inside with each control possibly having 40 properties. (font / font choices). By multiplying these numbers you could be looking at 8000 states. If you change any one property from the 8000 it would trigger a new state which would then have to be remembered. The authors suggest that there is a simpler way using XML in a parent-child formation that could significantly reduce the 8000 down to 2000 creating a 75% reduction in the size of the XML. Smaller XML could mean faster processes, less space, maybe happier user.

XAML (eXtensible Application Markup Language) has grown out of the evolution (or idea of) regular XML. It became the specific generic mark-up language for applications and developers. Most likely XAML was needed because XML is too often considered a web development language. In XAML you can define a simple button in the graphical user interface without listing out arbitrary elements like color and style which is usually done in the design theme files. The state of the button can be easily defined to a few words which tell where and what about the button. The example given is <Panel11> <Button Content="OK"/> </Panel11>.

To simply the relevance of the article I can relate it to the building of my home. In layman's terms I should not have to reframe the house if I change the color of the front door. Sounds silly but that is what happens in the way software states were kept. One change in a property could alter every element in the user interface. The suggestion is to separate the states and provide "children" or lower classes of the state that only pertain to that state. So if I want to change the color of my front door it will only affect the door – not the framing. When programmers try and revert or compare past states they would have to look at the entire list as a whole and find a change. The authors suggest a tool that would look more at the tree structure (parent/child/child) structure and report back any differences there.

The point is that a better structure creates consistency that can be reused by all programmers when building one app or other applications. The ease-of-use factor is mirrored by the ease-of-programming that occurs with the consistency.

Like many computer and internet related elements there is a lack of consistency among developers. Each believes that their newest code is better than the others therefore competition fuels advancement. Advancement is great but eventually the populas has to agree to some sort of standards that become the "norm". The norms are what allow work to progress even further simply because you don't have to reinvent the tires when making a better automobile. The authors of this article are simply suggesting alternatives to help create a better standard.

References:

Tubishat, Muhammad, Alsmadi, Izzat, Al-Kabi, Mohammed (2009). Using XML Files to Document the User Interfaces of Applications. GCC Conference & Exhibition, 2009 5th IEEE, Digital Object Identifier: 10.1109/IEEEGCC.2009.5734242, Page(s): 1 – 4, Accessed on 05/04/2011 from IEEE.org (http://ieeexplore.ieee.org.ezproxy.liberty.edu:2048/search/srchabstract.jsp?tp=&arnumber=5734242&queryText%3DUsing+XML+Files+to+Document+the+User+Interfaces+of+Applications.%26openedRefinements%3D*%26searchField%3DSearch+All)

< Prev Next >