whitepaper

Creating iPhone Apps With ZERO Code

By Bob Cusick

Creating iPhone Applications With ZERO Code

Along with about 13 million of my "closest friends" - I, too bought an iPhone to replace my aging “Charlie's Angels” Mororola phone. Now whether you hate the iPhone or love the iPhone doesn't really matter - what matters is Apple (and now Google, and Palm and RIM) have created mobile devices that are actually compelling. They're more than just a phone, they're a mini-computer in your pocket.

Because people are seeing them more as mobile communication devices rather than "just" a phone - it also raises their expectations as to how they interact with data. Especially data they interact with on their non-mobile devices.

The bottom line is that more and more executives and users will begin to ask for access to their data applications (or at least subsets of them) while they're on-the-go. And that will mean that someone (most likely you or someone on your staff) will probably be asked to create some of these types of mobile applications sooner rather than later (if you haven't been asked already).

For the purposes of this project, I decided to focus on the iPhone for a few different reasons:

1. It has a decent, nearly full-featured browser (no Java Applet support or Flash yet)

2. The screen resolution is fixed - no need to code for different screen sizes (like the absolute nightmare that exists in the Windows Mobile Market)

3. I wanted to quickly create some iPhone applications that I could use for myself

Armed with this short list – and having no “real” stakeholders to please (other than myself) – I set about trying to “flesh” out my design and requirements in a semi-formal manner. I didn't go “nuts” or anything – I just wanted to get the basics down so I would have a road map to follow – to see if the thing I was envisioning was even possible.

The Vision and the Requirements

I wanted to create an application that would, itself, build iPhone applications. Basically, a meta data repository that either IT folks or even (gasp!) end users could use to build specialty iPhone applications without having to wait for a "coder" (who is always knee-deep in "urgent" projects) to get around to building it.

The application had to be fast, flexible and somewhat sexy, and once deployed - it had to run anywhere on anything (OS- and database-wise). The last thing I wanted to do is create this framework and have it limited to only a certain subset of users on a certain subset of hardware that was capable of running on only a subset of database engines.

Everyone has his or her own "favorite" tool for creating applications - but for this project I was wide open to anything and everything out on the market today.

I was reading Tom Thompson's excellent article in the Dr. Dobbs Journal (now only available online) entitled "Porting Javascript Applications to the iPhone" (http://www.ddj.com/mobile/211200910) - where he describes how he took some example HTML code and modified it for the prototype application he was writing. That's one way to go: use someone else's sample and hand code the application.

While that's a great approach if you only need a few applications, or you need some really custom stuff, I was hoping that I could create something that I could easily create a lot of different iPhone applications from – using different database servers running on different vendors' databases and platforms. And, more importantly, I didn't want to have to hand-code all the inevitable change and enhancement requests that would come down the line.

The other thing that I wanted (if possible) - was to create something that would already support AJAX with little or no code - that also had a very small client-side JavaScript library due to the iPhone's limited capabilities. Because of the iPhone's limited screen real estate, I wasn't going to spend a lot of time trying to build something that had a multiple document interface... so that was one less thing to worry about.

After talking with some other developers and having a thorough Google session of my own - I defaulted back to my own personal development tool of choice: Servoy.

Servoy meets all of the project requirements, and because I can script it in JavaScript or straight Java (or both) - I wouldn't have to do the "down and dirty" coding of database connections, data binding, connection pooling or any of the other stuff I would have to in other environments.

In addition, Servoy will automatically translate all the forms that I specify in the “builder” application (a mini IDE) into HTML with AJAX built in without any code on my part. That means I “all” I had to do was to develop the native client "builder" application and let Servoy handle the actual iPhone deployment.

Best of all – since Servoy was handling the whole thing for me – I would have a centralized, database repository with multiple solutions stored as data. That means it would already be in the regular backup schedule, and I could have multiple database repositories on multiple different database backends – all with dozens or even hundreds of individual iPhone applications.

OK – that's pretty cool. So, I decided that the idea had some merit – and then move into the next (and one of the most fun, in my opinion) - the proof of concept prototype.

Creating the Prototype

So now that I had decided on "what" of the application, I had to now determine the "how." I wanted to build a rough, “quick and dirty”, proof-of-concept to make sure what I wanted to do would actually work. I fired up the Servoy Developer and created a basic schema and then sat down to decide on what I wanted the final output to be.

After looking at a few different iPhone applications, I decided that the default "Contacts" application would be the inspiration for the generated iPhone applications. I took some screen shots on the iPhone (hold down the power key and home buttons), emailed them to myself and used Photoshop to snag some graphic elements.

Since Servoy ships with Sybase's iAnywhere database and has some sample tables - I created a form based on the "companies" table. I then worked out what font/styles to use for the various elements. The official Apple Human Interface Guidelines for iPhone are only available to registered iPhone developers - which I'm not - so I did the next best thing: I searched the web and looked at other iPhone web frameworks' CSS.

Since Servoy's style sheets are also based on CSS, I had the basic look and feel up and running in no time. I created a simple list view of all the company names, and a simple company detail form.

In Servoy you only have to specify a single line of code to jump to another form - so it was easy to make the click on the list show the related detail and vice versa. I didn't have to worry about the database connection or data binding - and everything I built and previewed in the Servoy Developer would automatically get rendered in plain HTML/CSS automatically for me by Servoy.

All that was left to do was view the progress – to make sure that everything would look “pixel perfect” in the browser. I fired up Mobile Safari to have a look. Servoy Developer has a built-in application server so it took less than 5 seconds to view the application "live" on the iPhone. Everything worked exactly the same as it did in the native client (another one of the benefits of Servoy) - and once I made some slight adjustments to the style sheet - I finally had a good-looking and (better yet) working prototype!

Refining the Design

Now that I knew I had a viable end product design, I turned my attention to the configuration application or "builder" application in detail. I knew that the schema for this part would be pretty straightforward - there were going to be iPhone "solutions" that contained forms and those forms would contain objects. I was able to build a very straightforward 5 table model to encapsulate the basics.

I decided early on that I wanted to add a layer of automation that would create all the required forms for a particular solution on-the-fly so they could be easily changed. The last thing I wanted was to be locked into the old “generate, upload, preview” model that is typical when creating browser-based applications.

That meant that I needed to create a flexible enough data repository to make all the form objects nearly as configurable as they are in the "regular" Servoy Developer. In essence, I was setting out to nearly rebuild Servoy Developer – INSIDE Servoy Developer.

Beginning in Servoy 4.1 – creating objects (forms, fields, relations, valuelists, relations, etc.) is very simple, very fast, and very powerful – in only a few lines of code.

I don't know about you - but when I'm working on a blue sky personal project like this - I want to create something that I, myself, would actually use. And because I had the luxury of not having a particular deadline or a set of hard requirements or a bunch of stakeholders and end users to please - I had a lot of flexibility to do things the way I wanted to do things.

I started with the basics. Create a solution. Create a form. Put form objects on the form. Generate the Servoy forms programmatically, check on iPhone. I first actually created a fully functioning form – so I could mess around with the object locations, get the sizing and the look and feel correctly – so when I auto-generated the objects – they would come out looking correctly and be in the approximately the “right” position on the screen.

I was really amazed at how quickly it all came together. I spent maybe 20 hours and about 800 lines of code getting the basics of both the builder application and the deployed iPhone application to work.

I was so excited about the fact that I had build a mini-IDE in Servoy and that it would automatically create iPhone applications on the fly... that I made a “rookie” mistake.

Rather than just rolling out what I had on my development server and let other try it – instead I made the "mistake" of showing it to some iPhone fanatics. They immediately fell in love with my little hacked-together application - and it wasn't long before the change requests and enhancement requests came flooding in.

At that point, I decided that since the basic concept was sound and the application was basically functioning - it was worth the time and effort to turn it into a full-blown, detailed application.

Now I really had to sit down and do at least a functional specification – to make sure that I had all my thoughts down on paper – before I just starting coding stuff randomly. Because, as we all know – once something goes into production and people like and use it – you'll be supporting and enhancing it forever.

I went back to the drawing board to figure out all the things I wanted (and needed) for the IDE to be able to handle. The best way for me to do that was to try to build a “real” application – one that I would actually be using.

It was through this exercise that I came to realize that this was going to probably be a little bigger project than it started out to be – but I really believed it was the right thing to do. So, without hesitation – I grabbed some legal pads and sketched out as many requirements as possible.

I added features such as re-ordering the forms and objects, added support for most of the object types that Servoy supports (text fields, text area, HTML areas, labels, buttons, etc.) and even added three custom objects - a "Navigation Item" , a "Spacer" (that was a result of not wanting to create a bunch of separate labels and graphics every time I wanted a navigation item) and a “Record Navigator” object for moving between records.

Then as I continued to build iPhone applications for myself, I found that I needed to show related data, filtered data, related value lists, and on and on an on. That's really the benefit of thinking, writing and revising – thank goodness I used Servoy and not just plain HTML and CSS.

Servoy really allows me to be as agile as the ever-changing requirements.

It would up that I spent (no surprise here) most of the time and 60% of the code) "sand boxing" all the GUI elements to make sure that users don't accidentally go astray. I wanted to make it as "user-proof" as possible – and that takes a lot exception handling, the disabling of objects, checking dependencies, etc. That means more code – but still it's much, much, much less code than if I had to code it in .NET or PHP or Flex/Flash, etc.

Well, as I'm sure you've experienced in your own projects, it takes much more coding to make an application "easy" and "smart" than it does to rely on users entering the right thing in the right spot. The good news is that I didn't have to write all the "guts" of the code as well - only the application logic and error-checking. All the heavy lifting of database connections, data broadcasting, client state, HTML generation, AJAX calls, etc. is handled by Servoy with absolutely zero code on my part!

As of this writing, the application is still technically in late alpha, but if you would like to check it out for yourself you can download the free Community Edition of Servoy and my iPhone Application Builder from the Servoy site at: http://www.servoy.com/iphone – or you can download it from the “Resources” section in the Clickware site as well.