

ArcGIS for iPhone - Developing Applications Using the iPhone API

Using the ArcGIS for iPhone API, you can extend the reach of your GIS to include Apple iPhone and iPod Touch devices. The ArcGIS for iPhone API can access data from ArcGIS Online or from your enterprise servers and allows developers to implement custom ArcGIS functionality in iPhone applications. During this session, you will learn how to develop compelling iPhone applications that incorporate GIS functionality.

<http://video.esri.com/watch/85/arcgis-for-iphone-developing-applications-using-the-iphone-api>

Video Transcription

00:01 My name is David Cardella. I'm joined here by my colleague, Divesh Goyal.

00:06 And I'm also joined in the room by other members of the development team.

00:11 We've got Mark Dostal over here, the lead architect developer of our ArcGIS application.

00:20 It's available on the App Store now and built on our API.

00:26 Nimesh, we've got Nimesh here, product engineer for our API.

00:31 I'm not sure if Eric Ito is here or Jeff Shaner. No? No other team members? Okay. Great.

00:40 So you may have heard us... Excuse me for a second. We'll make sure...Okay.

00:55 So you have heard us talk about this term iOS, so before we get too far...

01:01 ...the name that you have in the catalog for this session is an Introduction to ArcGIS for iPhone, Developing Applications.

01:13 Well, a month or so ago, Apple renamed their operating system that runs on all of their touch devices.

01:20 So they have the same operating system that runs on the iPhone, the iPad, and the iPod touch.

01:26 And it used to be called the iPhone OS.

01:30 Well, because this was an operating system that ran on other touch devices...

01:36 ...they decided to rename their operating system to iOS.

01:40 So let's try this again. ArcGIS for iOS Developing Applications.

01:48 Okay, so we have also followed suit and named our API the ArcGIS API for iOS...

01:56

02:01 ...are also supported on all three of those iOS touch devices, as well.

02:06 So who has experience developing with iPhone, iPod touch, iPad applications? Hands?

02:17 No. Who's fairly new to it or little to no experience and wants to start getting into it? Okay, that's good to know. Okay.

02:24 So that's most of us here, so Divesh and I had authored this presentation to come at a high level and...

02:33 ...so it's good to know that we've hit most of the folks here today, so that's good.

02:38 You're in the right spot. What are we going to talk about?

02:41 We'll talk about what the product is and specifically what the API is and what you get with the API.

02:47 The language you will use to program iOS applications is called Objective C.

02:53 It might be different than any language you've seen before, so we'll go through some basics first.

02:59 And then we'll talk about some key concepts that Divesh and I feel will be in most applications that you build.

03:07 And that's my favorite part of the session because we put Divesh to work at that point...

03:11 ...and by that point, you guys are really tired of hearing me talk, believe me.

03:15 So after that, we'll also talk about your deployment options, so the custom applications you build...

03:20 ...what options do you have to deploy?

03:23 We'll talk about the status of our project and where we're going, and then we'll talk a little bit about our early adopters...

03:30 ...and who's currently using the API and what kind of applications they're building.

03:35 So what is ArcGIS for iOS? Well, it's really two components.

03:39 It's an application and it's an API.

03:43 The application is actually built on the API, and it's available in the App Store.

03:49 It's been available for over a week now, and it allows you to access maps that you've authored...

03:55 ...as you'd access these maps either through ArcGIS.com, our online GIS system, or through maps that you've hosted...

04:04 ...in your own on-premise ArcGIS Server.

04:09 The API itself is a native Objective C API.

04:14 This is important for a couple of reasons.

04:16 The first reason is that if you've been following the, I guess you could call it...

04:20 ...controversy between Apple and Adobe and other companies, Apple recently changed their licensing terms and conditions...

04:31 ...and they're not accepting apps to the App Store that are written in a different language and converted over to Objective C.

04:41 So for example, you can't write Flash-based applications...

04:45 ...use Adobe's utility to convert that to Objective C, and submit your app to the App Store.

04:49 So Apple has reserved the right to reject those applications.

04:54 We are in compliance with Apple's new regulations in that this is a native Objective C API.

05:01 It's got very tight integration into the existing Apple iOS development tools, Xcode and Interface Builder, okay?

05:10 The other important part about it being a native Objective C API is that the few of you who do have some experience...

05:16 ...developing apps in the iOS platform will find it a very easy transition.

05:23 In other words, we haven't created a new IDE. You still use Xcode.

05:28 We've also created some templates that integrate within Xcode.

05:30 Documentation integrates within Xcode. We have samples, as well.

05:40 Okay, so where can you get the API?

05:43 The API is available right now actually in the iOS resource center, and it is a package.

05:49 And the package contains a few things.

05:52 It contains some reference help that gets integrated within Xcode, the IDE, and as I already mentioned...

06:01 ...it contains templates and samples.

06:03 So you can download this from our website.

06:05 The resource center also contains some other tools or resources that you might be interested in...

06:10 ... some community tools like blog and forum, but also we've added some conceptual help that's posted online, as well.

06:17 So we've got reference as well as conceptual help.

06:21 And well, I guess, well, let's do a demo. Oh, this might be difficult with this.

06:29 Geez, I feel like Wayne Newton in Vegas right now. I'm not quite sure what to do with this, but... all right. Okay.

06:38 I think we're okay. I think we're okay.

06:43 Can you guys hear... okay, oh yeah, perfect.

06:48 So what I have here is a web application and this web application is a service request application...

06:57 ...so it allows someone in the community of Naperville, in this case, to report a problem within their community...

07:05 ...whether that's a pothole, graffiti, Divesh making a lot of noise next door...

07:10 ...that kind of stuff, so they can report those things.

07:14 But it also includes an iPhone application.

07:25 And let me...let me start up the right application first. So.

07:46 Okay, third time's the charm. Nice. Let's just zoom in a little bit here for you.... a little out of focus.

07:57 So we've got a service request application.

07:59 This was built by our team that also builds our government templates for mapping.

08:08 So I start up the application. It immediately wants to use my current location.

08:12 So this application, for example, would be made available through the App Store by a local government...

08:18 ...in this case, Naperville.

08:20 I'm not going to use the current location.

08:23 It brings up a map of Naperville.

08:24 There's other ways in which I can specify a current location.

08:29 I can specify it by address or I can zoom in to the application, and I thought this goes without saying...

08:39 ...but this is an application that is built on our API, so ... that's great.

08:45 I want to put this service report, though, in an area that I can then go back and find it.

08:52 So I don't know. Divesh, you're going to have to help me out. Try to remember where I put this, will you?

09:07 Let's try doing this with one microphone in your hand and then... Okay.

09:12 So let's get to an area where we've got some data. Great. Good. Where do I want to go? Let's

go right by the hospital here.

09:25 So the third way in which I can specify a potential problem is by clicking on the map.

09:30 It immediately prompts me or any user of the application to enter in more details.

09:35 The other thing it does is it does some reverse geocoding.

09:37 So you see there that we've determined the address in which I clicked.

09:44 And then I'm prompted to enter more information, like report type, and I've got a list of several to choose from.

09:51 So we can specify streetlight repair, we'll enter in a comment, and I can't spell...

10:04 Ideally, we could enter in a photo, and that photo can be either from the camera roll of the device...

10:09 ...or you can use the camera itself on the device.

10:12 I'm not going to do that, but I will add some contact information.

10:21 So one of the big benefits of having... let's just enter in some fictitious data here. Nobody call me, now.

10:30 Okay. One of the good things about having an application like this is that...

10:39 ...it decreases transparency between local government and the citizens that live there.

10:44 So that's why I added in some contact information whereby my local government can get back to me...

10:51 ...in that they've dispatched someone to repair this light or that the light has already been repaired.

10:56 Okay, so that's great.

10:58 So now with any luck...sorry about all of the feedback... with any luck I can just go ahead and refresh.

11:14 I'm just instantiating a pan. I'm not sure, is this... does that look around... this thing.

11:23 Well, we've got an ID here, right, 12404.

11:28 So we can query... well, whenever you don't have network connection, this yellow cable here...

11:36 ...will probably help you out, so I'm...

11:43 Don't look for that on the iPhone, though. Okay. Let's wait a couple minutes. Okay.

11:52 Fire hydrant leak, that wasn't it, right? No, 108... is it here... 124 is what I'm looking for. So let's just refresh this.

12:07 There we go. So we just see that a point has been added here, and indeed it is 12404. Okay, great.

12:16 So we see here that we've got our data. If I had chosen to take a picture and post it...

12:22 ...it also would be available via the attachments.

12:27 So that's an example of a custom application.

12:30 You may be wondering, a couple of people have actually already asked me whether that application is available.

12:37 And the answer is that it's not available yet, but it will be made available.

12:43 Okay, so this is going to be, that application you've just seen is going to be part of our government templates.

12:48 It's going to be a free download from the government template resource center, and what we're going to allow folks to do is...

12:57 ...customize it in one of two ways.

13:00 So we recognize that some folks will have experience developing applications using Objective C and some won't.

13:06 So you can download the project, you get all the source code, and you can either customize it through a configuration file...

13:12 ...so you can put your own logo, you can brand it, you can point it to your own data, all using a configuration file...

13:19 ...or if you want to do a little bit more heavy customization, you can go in, you've got all the source code available to you...

13:25 ...and you can edit that project.

13:27 At that point, you can either upload it to the website, in this case, if you're local government...

13:34 ...you can edit it for perhaps an internal application that does asset collection.

13:40 So there's many uses for that, but that is going to be made available shortly.

13:46 What can you do with the API?

13:47 We've got three main concepts. We can access REST endpoints or service layers, and we support multiple projections.

13:55 We support tiled as well as dynamic services. We also support Bing and OpenStreetMap.

14:02 The API also allows you to build applications that allow users to interact with the map and draw graphics...

14:08 ...and sketch graphics on the map.

14:10 Part of that graphics layer is the callout window that I just showed you in the application, where...

14:15 ...you can click on the map, get a callout window, and either give more information about that location...

14:21 ...or allow, in this case, a user to enter in more information about that location.

14:25 And then, of course, we've got a whole bunch of tasks or capabilities that you could do with the API...

14:31 ...which includes the standard things like query and identify but also geometry operations like measuring...

14:37 ...geoprocessing tasks. If you currently have some geoprocessing tasks that are served out...

14:43 ...you can access these tasks through your custom applications.

14:46 And very important is collect GIS features, or data collection.

14:51 This is analogous to editing, so the API gives you access to the new feature service at 10.

14:57 It allows you to edit your features.

15:04 What do you need to get started?

15:05 Well, let's start on the hardware side. On the hardware side, you are going to need an Intel-based Mac.

15:13 There's just no way around that. That is the development platform.

15:18 When we first gave this session back at the DevSummit, I specified that an iOS device was optional.

15:26 That's actually wrong. You absolutely need a device, whether it's an iPad, an iPhone, an iPod touch...

15:35 ...because the simulator is not enough.

15:38 You can develop something that looks great on the simulator and that you can interact with very easily using the mouse...

15:44 ...but as soon as you get it to the device, chances are you're going to have a much different experience.

15:52 So definitely, a device for testing.

15:56 Software. You need Apple's iPhone SDK. Yes, it's still called iPhone SDK.

16:03 You need 4.0. That is the latest version, that is the version that our API supports.

16:09 So how do you get this SDK?

16:11 Well, you can go to the iPhone developer resource center, Apple's iPhone developer resource

center...

16:16 ...and if you don't have a developer account, you can create one at no charge.

16:21 And you can download their iPhone SDK at no charge.

16:24 And the iPhone SDK includes a couple of IDEs; Xcode, which allows you to write your code; and also...

16:33 ...Interface Builder, which allows you to, in a very interactive way, design your user interfaces.

16:44 And then you're going to need our API, which is available from the iOS resource center.

16:51 Some of the technologies you're going to have to be familiar with is, first and foremost, Objective C.

16:56 Now those of you who do have C experience will find this transition quite easy...

17:00 ...but C also takes some of the same paradigms from other programming languages, as well...

17:07 ...Flex being one of them.

17:11 Also the Cocoa Touch platform. That is a specific Apple API that is specific to iOS touch devices.

17:20 So what happens when you tap on the device, what happens when you slide...

17:23 ...what happens when you double-tap, for example?

17:26 So there is another API that Apple offers on that stack.

17:31 And then of course, there's the iPhone SDK itself. Okay.

17:39 So let's talk about some basics of Objective C...

17:44 ...and I think I missed a demo in there where I was supposed to build my very first application, didn't I?

17:52 All right. Maybe I'll do that now.

17:55 So just to show you how easy it is to build your first application, who was in the session that we did immediately before this?

18:05 A few people. Okay, so you can tweet or check your e-mail for this, because this is the same...

18:10 ...this is the same demo that you've just seen. Or if you missed something...actually, I had to go through it fast...

18:15 ...so if you missed something, please, check it out.

18:16 So here is Xcode. We've got Xcode. It wants me to immediately use one of my previous projects.

18:25 I'm going to cancel out of that and I'm going to create a new project.

18:30 And when I create a new project, I can choose from a number of templates that Apple gives me...

18:35 ...or I can choose from a number of templates that we give me, we give ourselves, I guess.

18:42 And these templates follow some of Apple's most popular templates...

18:46 ... in that there are types of navigation-based applications...

18:50 ...tab bar applications, which is very similar to the ArcGIS application that we developed...

18:57 ...and then the simplest is this view-based application.

19:00 So we'll choose the view-based application and we'll call it Dev Session.

19:06 Why am I using a template?

19:08 I'm using a template because it makes things... Sorry.

19:10 Why am I using Esri's template?

19:14 Well, I'm using Esri's template because it makes things easy for me as a developer using our API.

19:20 It already sets up some build settings for me that prevent me from having to do it manually.

19:27 Like pointing to the path to where our ArcGIS API is installed, for example.

19:33 Now there's nothing stopping you from creating an application from scratch using one of Apple's templates...

19:39 ...but be aware, and this is all documented, that you're going to have to set some build settings manually.

19:44 So all we're doing here is taking the same Apple template and we're adding the build settings that are specific to our API.

19:53 The IDE creates a couple of classes for us.

19:56 There's Delegate class, which handles the startup and shutdown of our application.

20:00 And then there is a view controller that has both a header and an implementation file.

20:08 So rather than...you've seen me type, right?

20:11 So let me type with that 311 application, so you certainly don't want to see me type when I create this application.

20:16 So I've got a cheat sheet here, and all I'm going to do is just cut and paste some code and I'll explain what I'm doing.

20:24 So this is a constant that points to the REST endpoint of our topo map...

20:30 ...and that's going to be the data that we display for this application.

20:35 I need to instantiate or to at least declare an AGS map view object.

20:44 This object, as you might guess, is part of our API and it's the object that's going to display our map.

20:52 I also need to specify that this object is a property and I can do that with the property declaration.

20:59 And this is a special property; it's called IB Outlet.

21:02 Because this is an IB Outlet, I'm going to be able to access this object from within my Interface Builder IDE.

21:11 So I can tell the system, tell the application, the data you get from this object, put on this view.

21:18 Okay? Great.

21:21 Am I taking too much time, Divesh?

21:23 As always.

21:24 As always, yeah.

21:27 Okay, I'm going to use the synthesize declaration to set up getters and setters.

21:31 Because I have explicitly created this Map View object, I need to release it or in this case...

21:37 ...set it to Nil, and then here is all the business logic for this application.

21:45 Now I also get a couple of commented out methods when I create the template...

21:52 ...and the method that I want to uncomment is View Did Load.

21:55 So we're going to display this data once the view loads.

21:59 And so we just instantiate an incidence of an AGS tiled map service layer.

22:03 We add it to our map view that we created.

22:06 And then because we've explicitly allocated this tiled layer object, we need to release it.

22:14 We're not done yet.

22:16 So we've loaded this data into our object, but we haven't taken the data for that object and displayed it to the user.

22:22 In order to do that, we're going to open a Nib file.

22:24 A Nib file or an XIB nib file represents the visual screen, the visual UI that our users will see.

22:33 So right now this is blank.

22:36 I'll go into my library here and I'll search for a UI view component...

22:40 ...and all I'll do is just click and drag it over to my view and have it take up the full page.

22:46 So if I go to my document inspector now, I see that I have a view within a view.

22:51 Except this view is going to be a special view.

22:53 It's going to be a view of type AGS map view.

22:58 So I'll specify with the correct punctuation...

23:03 ...that that is of type AGS map view.

23:06 The last step that I need to take is to take that object that we've created.

23:11 Remember it's called Map View.

23:12 Remember I made it an IB outlet, a special kind of property.

23:16 That gives me access to it in the visual IDE of Interface Builder.

23:21 I can then say all the data from that map view object should be displayed in my map view visual object.

23:31 So we'll save that, go back to Xcode, and hopefully I wired everything up right.

23:38 Right, so we get our application, we get all of the native navigation components for us built within the API...

23:50 ...and of course, we get all of the great data from our topo map, as well.

23:56 So that's your first application. Congratulations.

23:59 All right. Basics. Objective C basics.

24:06 So you've seen some pretty weird syntax, right? A lot of square brackets.

24:13 And I know when I first started getting into it, I can't speak for Mark, Nimesh, or Divesh...

24:18 ...but it was an adjustment for me to take a look at all those square brackets.

24:23 But once you get used to it, it becomes second nature.

24:26 So how do we use these square brackets to make method calls?

24:30 Well, first thing is, we don't really call them method calls.

24:32 We call them sending messages.

24:34 So in this example, we are sending the Cancel message to the query operation object, okay?

24:42 I'm probably going to use those terms interchangeably, but just so you know, message is analogous to a method call.

24:48 So where we have a typical or traditional dot method notation, we replace with these square brackets. Okay?

25:00 How do we send messages in setting properties?

[25:03](#) Well, here we have a header file. Don't try and compile this code. It won't work, obviously.

[25:09](#) We have both a property, in this case, a string, as well as a method, okay?

[25:18](#) So in order for us to set or to set the value of a property, we do use dot notation.

[25:25](#) But in order for us to call this method on this class, we use the square brackets and send a message, okay?

[25:33](#) So that's typically the way we set properties and send messages in Objective C.

[25:39](#) So you're going to see a mixture of both dot notation as well as the square brackets.

[25:48](#) Multiparameter methods.

[25:50](#) So here we have a location manager object, and we've got a couple methods on this.

[25:55](#) Did update to location and from location.

[25:57](#) So we've got a method location manager... that's not an object. Pardon me.

[26:02](#) And we've got a couple of parameter methods that we can pass in, and they're both of type CLLocation, in this case. Okay?

[26:14](#) The naming of methods. You're going to see in our API that we have very verbosely named methods.

[26:22](#) And if you don't like it, you can blame Mark and Ryan. No, I'm kidding.

[26:26](#) We did this on purpose.

[26:27](#) So we have this method here called Did click callout accessory button for graphic.

[26:34](#) Wow, that's a mouthful.

[26:36](#) But basically the reason for the verbose method names is because if you are reading your code...

[26:41](#) ...or if someone is reading code that you have already written...

[26:45](#) ...you'll be able to better understand a more verbose method name than one that is very short.

[26:50](#) And this isn't something that we've pioneered.

[26:52](#) This is what Apple does.

[26:53](#) So if you look at their iOS API, they do the same thing.

[26:57](#) I think this method name is probably short compared to some of their longer ones.

[27:01](#) So just be aware of that, and you probably want to follow the same pattern when you're developing your own applications.

[27:10](#) Okay. Memory allocation.

27:13 So we allocate memory and instantiate instances of objects using the Alec alloc keyword and typically using the INIT init keyword, as well.

27:23 So whether it's just INIT init or it's INIT initWithURLwith URL, initWithStringINIT with string, initWithNSRAINIT with NSRA, INIT initWithwith something, okay?

27:33 So typically we Alec alloc and we initINIT.

27:39 I did mention that we need to release objects.

27:43 So in many other programming languages...

27:46 ...of C#, VB, I can't speak for Flex but...

27:50 ...these objects, most of the time, get released for us.

27:53 So how do we know when we need to release an object?

27:56 You need to release an object when you own it.

27:59 That's simple enough, but how do you own an object?

28:03 You own an object when you explicitly instantiate it using the Alec "alloc" method or using the Alec "alloc" keyword or the "new" keyword.

28:13 You own it when you copy one object to another.

28:18 You own it when you use the "rRetain" on an object.

28:22 So those are the three rules that you as developers really need to remember because if you don't release your own objects...

28:28 ...you're going to have memory leaks.

28:31 And you're going to get rejected if you submit an application with memory leaks to the App Store.

28:35 And it's really, really easy to do, very easy to do, very easy to create a memory leak in your application.

28:42 Fortunately, there are good tools with the iPhone SDK called Instruments.

28:46 I think one's called Leaks, another is called Instruments.

28:50 They're very good to tools to not only tell you if you've got memory leaks in your application...

28:55 ...but help you diagnose and figure out why you have a memory leak, and in a lot of cases...

29:00 ...at least, with my simpler applications, it's because I haven't released something that I own.

29:07 I am almost finished talking. So let's build something. Enough of the theory; let's build something.

29:13 Here are some of the five key concepts that we feel will be in most applications that you write.

29:19 There's no hard-and-fast rule, but we want to go over some of the most common ones.

29:23 Obviously, creating a map and adding layers to it, as you saw that I just did.

29:27 Querying data, not only querying data but getting at the attributes of this data.

29:32 You want to display selected features, as well, and also use the GPS.

29:37 So there's going to be many times when you build an application, you want to use the native capabilities of the device...

29:42 ...whether that be the GPS, the camera, the address book, e-mail, et cetera.

29:48 So we're going to look at the GPS.

29:53 Should we do the demo?

29:58 Yeah, maybe we should do the demo. Okay.

30:00 So Divesh is going to show us the final version of the application that we're going to build...

30:06 ...such that when you see some of the code that he's showing you...

30:09 ...you can relate it to something visual that's going to be in our final application. Okay.

30:15 And of course, you didn't understand a word of what David said there.

30:24 All right. We're up.

30:27 So I have here the iPhone simulator open and this simulator's simulating the iPhone 4 device.

30:34 You can actually change the hardware that you want to simulate.

30:37 You can choose an iPad or a specific version.

30:41 I have the application already in here, so let me go in and launch that.

30:45 What you will see is when the application came up, it's using the StreetMap data from ArcGIS Online...

30:52 ...but it's automatically zoomed in to the state of California.

30:57 What I can do is I can click on this button and bring up a panel...

31:01 ...which allows me to query for counties in California which have a population greater than some value.

31:07 In this case, it was 250,000 people.

31:13 I can change the slider's value and re-query for those counties, and now you see that I have less counties displayed on the map.

31:23 As I zoom around, I can actually click on one of these counties and bring up information about them in a callout window.

31:36 For example, this is Fresno County, and I'm showing the population in 1990 and 1999.

31:43 If I want to see more details about this feature, I can click on the Accessory button...

31:48 ...and it brings up this view about all of the attributes on that feature.

31:55 I can also go back and hit this button.

31:59 This button tries to find my location on the map using the GPS.

32:04 Now the new simulator that Apple's put out with iPhone SDK uses Wi-Fi IP addresses to find locations...

32:11 ...so let's see how good it's going to work here.

32:18 Yeah. I think it worked pretty well.

32:23 So that's something new in that simulator, right?

32:25 Right.

32:28 So briefly, this is the application that we're going to build...

32:30 ...and we're going to go over each aspect of what we needed to do to build this application.

32:36 Great. Okay. So as we build this application, we thought what we would do is...

32:49 ...we would talk about theoretically what you need to do for the key concept and then put Divesh to work.

32:56 So what do we need when we create a map and adding layers.

33:00 You've seen a lot of this already in the second demo that I gave.

33:04 It's the AGS map view object.

33:05 So we declare that, instantiate an instance of that, and then we instantiate an instance of some type of service layer.

33:12 In this case, it's a tiled service layer, and it points to a service URL or REST end point.

33:18 The example that we gave, it was the topo basemap.

33:23 At that point, when you have a layer, you add it to your map view.

33:30 So that's about it, right? Yeah, don't forget to clean up. Of course.

33:33 Remember, when you own an object, you must explicitly release it.

33:37 You know, I think I want to explain in delegate. You don't know your own...

33:43 Apparently, Divesh knows my slides better than me.

33:46 So other events that you can trap for on the map are listed here. These are just some of them.

33:53 And it's done through delegate objects, delegate events.

33:57 So we can get notified when the map loaded, when a layer becomes visible, when a user has interacted with the map in some way...

34:07 ...like clicking on it, when the user has interacted with a map and we should show a callout.

34:14 So that can all be controlled through delegate methods or delegate events on the map.

34:29 One. Okay, so I have my Xcode project open, and this is the very first part of the application.

34:37 And now I'm going to go through the same steps that David went through to create a map view-based application and wire it up.

34:44 But I am going to do a little bit more.

34:47 So first I want to show you is my view controller, and for the programmers out there...

34:52 ...you will recognize that Apple uses a lot of model view controller design patches.

34:57 And the controller's the heart of your application.

34:59 It controls the views that are displayed on the phone and it captures user interaction and response to it.

35:07 So I've already gone ahead and created a map view member variable.

35:11 It's got an IBOutlet IB out click keyword so I can manipulate that Interface Builder, and let me explain the Xcode layout a little bit.

35:21 So on the left-hand side here, I have the groups in file span, and I can see the classes that are part of my project.

35:25 And these include both the header and the implementation files.

35:33 The header is basically just a way to tell people what are the methods and properties that are public on your class that they can use.

35:42 Apart from that, there's the resources folder, and this will contain the images that you want to use in your application...

35:49 ...and also those interface files, which you use to build your user interface.

35:54 So let me just briefly show you that I've already gone ahead and added a map view to my main view.

36:01 And if you look at my view controller, you'll see that I have that map view member variable, and it's correctly hooked up.

36:07 So I don't need to do anything much here.

36:09 I'm going to go ahead and quit this.

36:13 Let's come back to the header file.

36:16 Now if I want to switch to the implementation of this class, I can use this nice little shortcut that says Go to counterpart.

36:25 And it's going to show me the implementation file, which ends with a dot-m.

36:30 Now the first thing that I want to is, when my view loads, Apple calls this View Did Load method.

36:37 This is where I'm going to add layers to my map.

36:42 So I've already got the code for that.

36:45 Let me go ahead and run through that code for you.

36:46 Basically, what I do is I access the map view and I set my own class as the delegate.

36:49 This is because I want a map to call into my class when it changes its extent or somebody clicks on it...

37:03 ...or there are other events happening on the map, so I can respond to it.

37:07 The next thing I do is I create a URL that's pointing to an ArcGIS Online service, in this case, the street map.

37:16 And I add it to the map using Add map layer. And that's it.

37:21 Now when the application starts, if you notice, it zooms automatically to the state of California.

37:29 And I do that by listening to a special delegate method that the map fires when it loads...

37:35 ...and that delegate method is called Map View Did Load.

37:40 So when that fires, I know that the map is ready and what I do is I just create an envelope with the extent that I want to zoom to.

37:48 And I tell the map to zoom to that extent.

37:51 Now you may be wondering, how do I know what methods are on the map's delegate?

37:56 Which ones should I implement? How do I find out?

38:00 Let me switch back to the header file, and you'll see that my demo view controller class is implementing that AGS map through Delegate.

38:08 So it's broadcasting that I implement this protocol and please call in to me.

38:15 If I hold down the Option key and double-click, you can see, I can see the reference for this class, write with an Xcode.

38:23 So I can see what methods I can listen for and then react to.

38:32 Let me go and run this project, and if I did everything okay, my map should load and it should automatically zoom to the state of California.

38:50 I was kind of getting used to you talking there. That was good. Okay.

38:55 What do we want to do next? Well, apparently my screen is frozen again. Okay.

39:08 Well, we're also going to want to, in most cases, query some of this data and display some of the results.

39:14 And we do this through two objects called AGS Query Task and the AGS Query.

39:20 You may notice as we introduce more and more objects, they're all prefaced with AGS.

39:27 There's no concept of name spacing in Objective C, so we, I guess, create a virtual name space by prefacing our objects with AGS.

39:38 So when you're developing, there's no mix-up between what's an iPhone object and what's one of ours.

39:46 So we create an instance of a query task, and in this case, our query task is INITed with a string.

39:55 And that string is a URL that points to a REST endpoint, points to a service, but that service has been query enabled.

40:05 Okay, so that's very important, so when you publish your services, you have the option to allow...

40:10 ... or to specify that that service does support a query task and that's what you're going to need to do.

40:17 After you've created a query task, you need to create the actual query itself...

40:22 ...and a couple of properties that you're going to have to set on that query...

40:26 ...are the Where clause of the Where portion of the SQL clause, in this case, a population.

40:33 We just hard-coded it in, or I've hard-coded it in for this example. Divesh has not done that.

40:39 So we need the Where clause but also any output fields, so we may want to return all the fields.

40:45 In this case, we specify an asterisk or we may want to select specific fields.

40:52 Then, of course, we need to execute the query by sending the message Execute with query...

40:58 ...passing in our query object to our query operation.

41:05 Don't forget to handle errors. How are you going to handle errors?

41:08 Well, we do provide delegate methods for you that notify you not only when you get results back...

41:15 ...but also if there are any problems with your query, as well, so be aware of that.

41:22 So we've queried the data and presumably we've gotten features back, but the application

right now...

41:28 ...we haven't shown you what to do with those features.

41:31 So in this most simplistic way, we want to at least display those features on the map for your user.

41:37 In order to do that, we'll use that delegate system or that delegate object that I've mentioned for the query task.

41:44 It's called the AGS query task delegate. And we use it.

41:49 That will tell us or notify us when we have results back...

41:54 ...and also any errors, but it will notify us when we have some results back.

41:58 When we have results back, we can then create a symbol and all the other objects and things we need to do in order to render it...

42:07 ...so in this case, a symbol, a fill color, an outline color.

42:10 And then we apply it to a graphic.

42:13 So you see the last code window there, we say graphic-dot-symbol equals the fill symbol that we just created.

42:19 Well, where do we get that graphic?

42:21 The delegate method that tells us we have features returned gives us that graphic.

42:26 It gives us actually a set of graphics that we can loop through.

42:32 And at that point, it's a matter of adding the graphic to a graphics layer.

42:35 So I had mentioned earlier in the session...

42:38 ...that we have the ability to display and either allow our user to sketch or display things on a graphic and...

42:44 ...selected features or results from the query are one of those elements in which we can display.

43:00 Okay. So we're back in our application. I've done a little bit of work while David was talking...

43:05 ...which I always do, and which he always does.

43:08 And I've gone ahead and added this button to this application.

43:13 And when I click it, a view pops up with a slider.

43:16 And I want to use the value of the slider, I want to let users change the value...

43:20 ...and then I want to use the value to query which counties have a population greater than that.

43:26 I've already got two labels in the view, and as you can see, as I move the slider, the Labels value gets updated.

43:32 So I've already done that work.

43:35 What I want to do is when the user lets go of the slider, I want to execute a query.

43:40 Now some of you might be tempted to execute a query as the user's changing the value...

43:45 ...but don't do that because you'll end up firing a lot of query operations that are unnecessary.

43:51 So let me go ahead and show you the changes I've made to that view controller.

43:57 Basically now, I have some member variables for the slider and the labels that I've added.

44:04 I've also got these methods that get fired...

44:08 ... when a user clicks on that button to bring up the view or changes the value of the slider.

44:14 There's this one method, Slider Touch-up. In this method, I want to actually execute my query.

44:21 But before I do that, let me go ahead and open up the Interface Builder file and wire up that method.

44:31 So as you can see, this is the layout of my interface file now.

44:35 I have a toolbar at the bottom with that button, and I have this view that I display, which has those two labels and the slider.

44:43 Now if I look at my view controller, you notice that the Slider Touch-up method is mentioned there...

44:50 ...and it's not connected to anything.

44:52 I'm going to go ahead, drag that, and drop it on the slider.

44:57 Now the slider has a lot of events that it fires, so I want to hook up my method to the right event...

45:03 ...and in this case, it's a touch-up inside.

45:06 That event is fired when the user lifts his finger off the slider.

45:14 So I'm going to go ahead and save that and quit Interface Builder.

45:19 Let's look at the implementation for this method.

45:24 So jump to definition. It's currently blank. Let me go ahead and copy the code that I want to execute.

45:34 So basically what I'm doing is when the user changes the value of the slider...

45:37 ...I cancel any outstanding query operations that might be running.

45:42 I create a new query task, and I point it to the layer that I'm going to query in my service...

[45:49](#) ...in this case, the Counties layer has an index tool...

[45:52](#) ...and you can find this URL very easily from Services Directory off your ArcGIS Server.

[45:59](#) Again, I set the delegate of the query task to this class...

[46:03](#) ... so I can respond to things like when the query succeeds or fails.

[46:08](#) I create the query. I set its Where clause, populations greater than the value that I get from the slider...

[46:15](#) ...and I tell it to execute the query.

[46:17](#) Now this is going to happen in the background, and when the query is done...

[46:22](#) ...there are going to be a couple of delegate methods that get fired on my class, so I need to implement them.

[46:27](#) As a good programmer, I'm going to implement the failure method first.

[46:32](#) I'm sure all of you already do this.

[46:35](#) And I show the Alert view and tell, hey, you know what, the query failed. It's pretty simple.

[46:40](#) And if my query succeeds, I'm actually going to be passed in a feature set by the query task.

[46:49](#) This feature set contains all the counties that have a population greater than the value that I've specified.

[46:55](#) So what I do is I just create a simple fill symbol.

[46:57](#) In this case, a simple fill symbol with the color red that I want to use to display those counties.

[47:05](#) And I remove all graphics from my Graphics layer that might already be there.

[47:11](#) I look through all the graphics in my feature set, and I assign them that symbol...

[47:18](#) ...and I add the graphic to the Graphics layer.

[47:21](#) Once I've added all the graphics, I send one message to the Graphics layer saying...

[47:25](#) ...the data has changed; please redraw yourself.

[47:31](#) Oh, and I didn't talk about the network activity indicator.

[47:35](#) Basically, when I fire the query, I set this property to U.S....

[47:40](#) ...so basically I want the network property indicator to be turning away on the user interface...

[47:45](#) ...so the user knows something's happening in the background.

[47:48](#) And when I get some results back or when my query fails, I turn it off.

[47:55](#) So that's all the code I need to implement my querying. Let's go ahead and run this.

48:06 And you can see the network activity indicator is working in the background.

48:11 [Inaudible]

48:16 But now you can see, I have these counties in the map...

48:19 ...but if I click on them to see more information, I don't get any more.

48:23 We'll get into that a little later.

48:29 So Divesh, I thought you said you were working while I was talking.

48:33 I guess there was a little bug there, wasn't it?

48:35 Maybe you'll have it fixed by the time we get back to you. No, I'm just teasing.

48:40 Okay. What's wrong with my slides here? Okay, there we go.

48:43 So as Divesh mentioned, we've gone ahead and we selected features, we've displayed them on the map...

48:50 ...but we'd also like to display some of these attributes to the user, and we do this through the AGS Info template.

48:58 And so what you guys are going to do as developers, you're going to create your own template class.

49:05 And you're going to use the AGS Info template protocol to do that.

49:09 There's two important methods that you're going to need to override.

49:13 One is called Text for Graphic and one is called Detail for Graphic.

49:18 And the Text for Graphic represents the bigger, bolder text in the callout window that we're displaying here...

49:27 ...while the Detail for Graphic represents the text below.

49:32 So you're going to have to implement this and override these methods to tell the application and...

49:37 ... to show your users what fields, and what date, and/or what data you want displayed there when they click on a selected feature.

49:46 So once you've done that and you've implemented your Info template class, you can create an instance of it...

49:53 ...and then you can apply that Info template to the graphic or the resulting features that come back from the query.

50:01 So you've set up how the data or at least how the title and then the detail text is going to be displayed in your callout window...

50:13 ...but what you haven't done is...

50:15 ...you haven't specified how all that attribute information is going to be presented to the user of your application.

50:22 And you can do this through a delegate method. Oh, we've talked about this once.

50:26 It's one of those long ones. They'd click Callout Accessory button for graphic.

50:31 When does this delegate method get fired?

50:34 It gets fired when you click the More Details button in the callout, and the More Details button is that chevron...

50:40 ...that if we go back one slide, it's that chevron in the callout.

50:46 So when that is clicked, you are notified when that is clicked.

50:50 And then you can respond appropriately, and the snapshot we have up here, we're prompting for more information.

50:58 So this is similar to the 311 app we showed you.

51:01 What Divesh is going to show you is he's going to display information.

51:05 So he's not going to prompt for more information, he's going to display the information behind the selected features.

51:18 All right, okay. So I've gone ahead and done what David just talked about and created a new class called County Info Template.

51:27 This template will show information about a county that the user taps on in the callout window...

51:33 ...and basically just implementing the AGS Info template protocol.

51:38 Now for those of you who are Java developers or .NET developers, this is similar to an interface that you implement...

51:44 ...but unlike an interface, some of them at first can be optional in Objective C.

51:51 So let's see the implementation for my county template. It's really simple.

51:57 In the text field graphic, what I'm basically doing is I'm going to the graphic's attribute and pulling out the value of the name...

52:05 ...and that's what I'm going to display in the callout.

52:08 I also get a chance to display one line of detail for the feature by implementing the Detail for graphic and in that case...

52:17 ...I'm going to go into the attributes of the graphic and pull out the values for population in 1990 and 1999.

52:24 And then I'm going to create a string with it that displays those two populations.

52:31 So once I've created my info template, all I need to do is assign it to my graphics, which I've put on the map.

52:39 Now I'm going to look for the query task delegate methods that I implemented in the last section...

52:47 ...and as you can see, Xcode provides me a nice way to find the right contents in my class.

52:52 And it does that by allowing you to put Pragma directives in the code so that it can populate it for you...

52:59 ...and you can find the right method.

53:03 So this is that method I implemented in the last section that was displaying the features on the map using a fill symbol.

53:10 I have to do an additional step here.

53:13 What I do is I create an object of my County Info template and I assign it to the graphics...

53:21 ...just when I assign the symbol to a graphic. And that's it.

53:24 And at the end of that, I release my county template because I don't want to have a memory leak.

53:30 So now my graphic has a symbol and an info template.

53:33 It has everything to be displayed on the map and allow a user to click on it and bring up more information.

53:40 Let me go ahead and run this.

53:47 Execute a query. Fixed, David!

53:55 And if I click on the county, you can see, I get more information.

54:01 Now if I try to get even more feature information for this county, I can click on the Accessory button...

54:06 ...but nothing's happening because I haven't handled that delegate method.

54:11 So if we go back into my code and add the delegate method that gets fired when a user clicks on that button...

54:19 ...and that method is Did Click Callout Accessory button for graphic.

54:25 It just reads like English. It's pretty simple.

54:29 And what I do in that method is I create a feature details view controller, a new view controller that I've written...

54:36 ...and then display that view controller.

54:39 Now if you look at the definition for this view controller and the implementation...let me

switch to the header file...

[54:47](#) ...it's basically just displaying a table, and I won't go through the code for this because...

[54:52](#) ...Apple provides some great information on how to create tables...

[54:55](#) ...because fields see most of the interfaces of applications display tables.

[55:00](#) So I'm just going through all the attributes of the feature and displaying it in a table.

[55:06](#) Let me go ahead and run this again.

[55:15](#) And this time, my new view controller comes on and I can see all the attributes in this table.

[55:27](#) And the activity indicator bug is fixed. Touché, very nice.

[55:35](#) The last key concept we're going to talk about is the GPS...

[55:38](#) ...so we have wrapped some of the Apple iPhone GPS API, called CL Location...

[55:48](#) ...and we've wrapped some of what we believe to be...

[55:51](#) ...some of the more common GPS functions that you as developers will access...

[55:55](#) ...the ability to start the GPS, to stop the GPS, but also to get at the current point.

[56:02](#) Then once you get at that point, you can do various things with it...

[56:04](#) ...zoom to the point, display some more information, things like that.

[56:09](#) [Inaudible audience question]

[56:11](#) Yes. I was asked to speak a little louder. I will, sorry. Is that okay? Should I repeat that?

[56:19](#) I think he said, "Can you speak a little less?"

[56:27](#) You and I will talk later.

[56:30](#) [Inaudible audience comment]

[56:32](#) What's that?

[56:34](#) [Inaudible audience response]

[56:36](#) All right. So this is the same application, getting back on track.

[56:42](#) I've added a new button here to allow the user to center my map on the GPS location.

[56:48](#) At this point, it doesn't work, we're going to wire it up and make it work.

[56:51](#) I've also added this view on top that's going to label it to its current location...

[56:56](#) ...and a placeholder label where I'll put the coordinates of the GPS location.

[57:05](#) Again, back to my view controller, you notice that I've added a new method, called Toggle

Location.

57:11 And I want this method to be fired when the user clicks on that GPS button.

57:16 So let me go ahead and wire this up. I'm going to again go into my Interface Builder file...

57:21 ...and you can see a pattern developing, right?

57:23 Create a method, wire it up, then implement it.

57:27 And now I can see the structure of my view, you'll see this button that I added.

57:33 And my demo view controller shows that Toggle Location method that I have added to my view controller.

57:41 I'm going to go ahead, wire it up to that button, and my job here is done.

57:51 Let's go ahead and implement that method. I have some code here, and basically what I'm going to do is...

57:58 ...I'm going to see if GPS is already enabled, then I'm going to stop it.

58:03 And if it is not, I'm going to start the GPS and I'm going to set the autopan property to True...

58:09 ...so the map view provides us a convenient way to keep recentering the map as the user moves around...

58:15 ...so you don't have to write any code.

58:18 Now, David and I have talked a lot about delegate methods, right?

58:23 And if you didn't understand the concepts, it's really simple.

58:26 We all are delegates for somebody else in real life, like our bosses and our wives.

58:31 They tell us to do something, we do it for them.

58:34 But the limitation of delegates is you can only have one delegate for an object.

58:39 You can have only one delegate for the map.

58:42 Now Notifications is another way in Objective C where you can give information to a whole lot of classes...

58:50 ...and we have some notifications that the map fires, like the map did in panning and the map did in zooming.

58:57 So what I've done here is I've gone ahead and added myself as an observer for these notifications...

59:05 ...so that I can know if the user is moving around and the GPS location is changing, I get a chance to do something.

59:13 When this notification fires, I'm telling iPhone to call my method called Show Current

Location.

59:21 Let me go ahead and implement that method.

59:25 And basically what I do with this is I go into the GPS of the map view and bring out the current point of the GPS.

59:32 I create a string with it and assign it onto the label. Simple.

59:38 Somebody's calling me.

59:40 It's your delegate.

59:45 And now if I click on this button, you'll see that it zooms in to this location.

59:58 That's great. Thanks, Divesh, especially the explanation of delegate methods.

1:00:04 That's unique. I've never heard that before. That's really good.

1:00:08 Okay, so what is the developer experience?

1:00:13 Well, again, in summary, download both the Apple iPhone API, install it.

1:00:21 Download our API and install it, and it will automatically integrate within the Apple development environment.

1:00:27 And then, of course, develop your application using a lot of the techniques that Divesh showed us today...

1:00:33 ...and then you'll need to determine a deployment model, and that model can either go through the App Store...

1:00:41 ...or you can install directly through iTunes, and we'll talk a little bit more about that, actually, right now.

1:00:47 And then once you've just selected that deployment model, of course, it's a matter of downloading and using on your iOS device.

1:00:55 So how do you as developers choose what deployment model you should be using?

1:01:00 Well, to answer that question, you need to ask yourself...

1:01:04 ...who's going to be using the application that you develop.

1:01:07 Is it going to be for consumers, or is it going to be for, for example, an internal enterprise solution?

1:01:15 If it's going to be used by general consumers, then you'll want to use the App Store, which means...

1:01:21 ...that you'll need to submit your application to Apple, have it approved...

1:01:25 ...and then they take care of the hosting, distributing, and the providing of stats, as well, for your application.

1:01:34 The approval process now is averaging between four and seven days, so, you know, within, well...

1:01:43 ...probably about a year or so ago, we heard a lot of nightmare stories about apps being rejected...

1:01:47 ...and why they were being rejected and it taking two weeks to get your app submitted and approved.

1:01:53 Apple's done a really good job of not only communicating the top reasons why they reject apps...

1:02:02 ...but also improving the turnaround time on applications that have been submitted.

1:02:07 And just an FYI, the top reason why apps are rejected is that you've promised something in your application...

1:02:16 ...and your application doesn't do it.

1:02:19 This could be one of two things.

1:02:21 In your description, you may say your application does A, B, C, and it actually doesn't do A, B, C.

1:02:28 Your app will get rejected.

1:02:31 Also, don't promote any functionality that's coming soon.

1:02:35 So don't say in your description, Our next update, we're going to have data collection; look for it in the App Store.

1:02:40 No, you're going to get rejected. Also in your UI, if you have "coming soon functionality" within your application.

1:02:51 So you've got a tab that says Collect Data, or Edit, and then when you click on the tab...

1:02:57 ... you have a message saying, This functionality hasn't been implemented yet; check back soon.

1:03:02 You're going to be rejected. So that is the number one reason why apps get rejected, so be aware of that.

1:03:09 It sounds obvious, it's a very easy, easy hole to fall into. So just be aware of that.

1:03:17 Oh, right, so back to my ... how do I get off topic? Divesh, help me.

1:03:23 Okay, so if it's an application that consumers are going to be using, you go to the App Store.

1:03:27 If it's an application that's an internal application, a solution for your company...

1:03:32 ...then you can deploy it through iTunes.

1:03:36 Now deploying through iTunes in an enterprise environment involves you getting an enterprise license from Apple.

1:03:45 Okay, so an enterprise license from Apple is actually, last I checked, was 299 [dollars], but don't quote me!

1:03:53 It could have gone up a bit, but it's in that range, and what it allows you to do is unlimited deployments...

1:03:59 ...to as many devices as you want.

1:04:03 The devices need to be owned by the entity or the company in which...

1:04:07 ...you're registering your enterprise license with Apple, so be aware that in order to distribute internally...

1:04:14 ...your final application through iTunes, you will need to get an Apple license.

1:04:18 [Audience question] Is that per year?

1:04:19 That is per year, it's a subscription for a year, yes.

1:04:25 Okay, so our current status and what's next.

1:04:27 I mentioned that the product does come with an application, and after one week...

1:04:33 ...we noticed over 21,000 downloads. Up to date, which is probably a week in three or four days...

1:04:41 ...we have over 50,000 downloads.

1:04:44 We're really happy about that.

1:04:46 I'll be honest, I'm a little surprised at that number.

1:04:48 I didn't expect it to be so high, and we're getting very, very good daily downloads.

1:04:54 We were for a short time the number-one free productivity app in the App Store...

1:04:58 ... and then we got knocked off by an alarm clock.

1:05:03 Which you should use when talking.

1:05:05 Wow! Wow, wow, wow.

1:05:08 No, actually, I think the alarm clock was used to remind people to download and use our apps...

1:05:12 ...so it was hard to stay ahead of that.

1:05:15 And we were also featured on the new and noteworthy apps and downloaded in 79 different countries.

1:05:22 What do we plan for the future? We plan very short, incremental updates of functionality and...

1:05:28 ...even shorter incremental updates of any problems that we find within our software.

1:05:33 But functionality, we're working on a collection, data collection or data editing.

1:05:38 So what we are going to provide through the application is the ability for you, or your company or your consumers...

1:05:48 ...to access your data and edit it, add features to it, edit existing features, et cetera.

1:05:54 Improved user experience, user interface.

1:05:57 We're always looking for ways to improve the user interface, to make it more intuitive...

1:06:02 ...to reduce the number of taps that the user has to perform to get to a certain functionality.

1:06:08 Had a lot of feedback on offline use.

1:06:11 Right now our product is dependent upon being connected, and that is going to change.

1:06:17 We are going to implement eventually...and that is not coming in two or three months.

1:06:24 Just want to be clear on that.

1:06:25 But in a future release, we will implement offline usage.

1:06:30 Download your data, go into the field either with no connectivity or sporadic connectivity...

1:06:35 ...edit data, use the data; when you are connected again, post back to the server.

1:06:41 We will have that.

1:06:42 [Audience question] On iPhone?

1:06:43 iPhone, iPad, iPod touch, all the iOS devices we support.

1:06:47 I'm sorry, who asked that? Yes.

1:06:50 [Audience response] I did.

1:06:52 Great. Current status of the API.

1:06:55 The API is available for you right now, it's in public beta, it's available from the resource center.

1:07:01 We plan to go final with it in August.

1:07:04 Go ahead, use it, develop with it, get used to it, and be ready for when we go final to take your apps to the App Store.

1:07:11 And I can't emphasize enough how surprised I am.

1:07:16 I mean, I guess I always knew the App Store had very good penetration into the consumer market...

1:07:22 ...but I'm really surprised to see that we've had so many downloads, but also so many downloads from...

1:07:28 ...folks who aren't in the GIS industry, and you can tell because of the feedback they're giving...

1:07:32 ...the questions that they're asking.

1:07:34 You guys can take advantage of this same market, and it's all been created for you.

1:07:40 So it's actually a very, very good opportunity.

1:07:43 Incremental updates again for the API.

1:07:46 When the API goes final, it's not going to be available in the App Store.

1:07:49 It's not an application, it's a developer tool.

1:07:51 It's going to be made available in the same mechanism you can get it now, on our resource center.

1:07:56 We are going to have some incremental updates there.

1:08:00 Not only are we supporting the functionality that I mentioned that we're putting in the application in the API...

1:08:07 ...because again, the application is built on the API.

1:08:09 So what's ever in the application will also be exposed to the API.

1:08:13 We're also looking at ways in which we can support time-aware layers.

1:08:17 Time-aware layers is a new type of layer in ArcGIS Server 10.

1:08:22 And of course, we're going to expose offline use through the API, as well.

1:08:26 Questions, comments? If you have them outside of this week, of course...

1:08:32 ...you can use this alias, ArcGIS for iPhone.

1:08:35 You can tell that we created that alias before the name change, but we're keeping it the same.

1:08:40 But this is why we're here this week, to answer your questions and help you out...

1:08:44 ...and we've got members of the development team here.

1:08:47 We've got the Mobile Island, as well, to answer questions.

1:08:53 [Inaudible]

1:08:55 Did you want to show another demo?

1:08:57 Yeah.

1:08:58 So right now, what we did was we created, or Divesh created a very simple demo just to kind of...

1:09:05 ...illustrate some basic concepts.

1:09:08 We know that you're probably going to want to create some more complicated applications, so...

1:09:12 ...Divesh has a couple of samples that are a little bit more advanced that he'd like to briefly show you.

1:09:39 Okay, so let me give you the background for why I created this demo.

1:09:43 Basically, me and my friends go hiking a lot. All right?

1:09:46 And a couple of weeks back, we went to the Yosemite National Park.

1:09:50 Now, as all good friends like me and David are, we don't agree with each other.

1:09:55 So we landed in the park and, of course, we don't plan in advance.

1:09:59 We were wondering what we should do there, and everybody has ideas.

1:10:01 Somebody says we should go see a waterfall, somebody wants to go hiking, and we can't agree on that.

1:10:07 And so somebody pulled out their iPhone and said, hey, let's decide what we want to do.

1:10:11 And we're all trying to huddle around this one iPhone, kind of touch it and show them where we should go...

1:10:18 ...and it was not working out.

1:10:21 So what I did was, I created this small application that allowed people to collaborate with each other very easily...

1:10:28 ...as long as they're within Bluetooth range and they can share a map.

1:10:35 So let me go ahead and try and connect these two applications together.

1:10:40 And I've turned on my Bluetooth and I'm searching for Ppeers, and it's detected these phones.

1:10:46 Let me go ahead and connect to this other phone, and you will see a dialog pop-up soon that'll allow me to accept or reject the connection.

1:10:58 I'm going to go ahead and accept. Now these two applications are connected, and...

1:11:10 ...as I zoom this map and pan it, you can see changes on the other application.

1:11:13 And then I can tell my friends, hey, you know what, this is great.

1:11:17 Oh, it crashed! Well...

1:11:22 I thought you were working while I was talking.

1:11:24 So this application is not going to get into the App Store anytime soon.

[1:11:30](#) It's still a work in progress.

[1:11:32](#) But you can see the idea and I'll be doing a demo here tomorrow...

[1:11:35](#) ...where I hope this doesn't crash.

[1:11:38](#) I'll probably have to work tonight but...

[1:11:39](#) I guarantee you, it won't.

[1:11:42](#) I think we're out of time, so we'll try and... you have some more stuff to show?

[1:11:47](#) No.

[1:11:48](#) As always.

[1:11:49](#) You asking me if I can talk?

[1:11:50](#) All right. So come to the Demo Theater, we'll do this demo again and a few more demos...

[1:11:54](#) ...and we'll go into the code of how you can create applications like this, and really take it to the next level without crashing.

[1:12:02](#) Thank you.

[1:12:08](#) I do have a few more slides just of early adopters.

[1:12:11](#) I'm not going to show them because we have a mobile SIG right now in the room next door at 12.

[1:12:15](#) Lunch is going to be provided. We'll talk about our whole stack of mobile devices, and we'll talk about the iOS stuff, who's using it...

[1:12:25](#) ...another cool 311 demo, so thanks, thanks for attending.