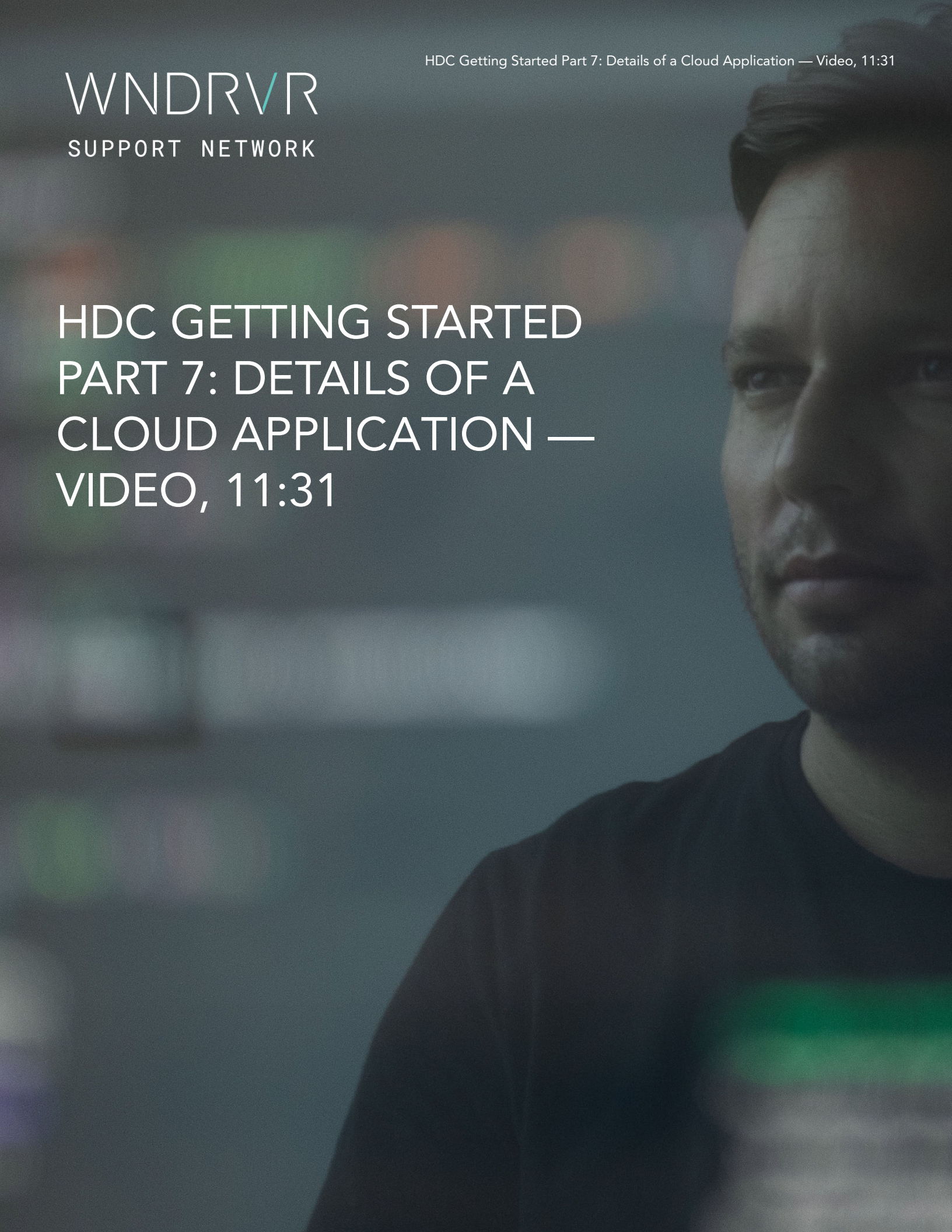
WNDRVR
SUPPORT NETWORK

HDC GETTING STARTED PART 7: DETAILS OF A CLOUD APPLICATION — VIDEO, 11:31



Corporate Headquarters

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): +1-800-545-WIND
Telephone: +1-510-748-4100
Facsimile: +1-510-749-2010

For additional contact information, see the Wind River website:

www.windriver.com

For information on how to contact Customer Support, see:

www.windriver.com/support

HDC Getting Started Part 7: Details of a Cloud Application — Video, 11:31

29 January 2019

1. HDC GETTING STARTED PART 7: DETAILS OF A CLOUD APPLICATION — VIDEO, 11:31

Published on 6 October 2015

Transcript

Time (mm:ss)	Narration
From: 00:11	Ok, let's take a refresh of the example application that we're going to look at. I wanted to bring this slide back up just to refresh your memory about the flow of how this is going to work. We're actually implementing an application to run in a Node.js server, and it will serve up pages and JavaScript to a browser when you point at it, and then the browser will be able to make requests back into the Node.js server, which will then be transformed into Mashery calls to RESTful API calls into the platform. The platform will make these calls and then return either JSON or XML data back, and the browser can then render that data in however it's defined in the JavaScript code for that rendering.
From: 01:01	Notice over on the far right, there's also what's called custom objects. These are the Groovy scripts that you can define as extended features into the cloud platform, and so you can define some of these and they could be called as well through the RESTful API. Ok, let's take a look now at what the application actually looks like when we run it. So, let's first of all take a look at an already running instance of it, and I've saved a link here for you. It goes to the Website of emsdemo.windriver.com, and you'll see that I have the application running here, and you can log in or sign up, and if I login at, for example, to "wrpoc8", and the same password, and go ahead and log in, and you'll see that it actually is bringing
From: 02:02	up a whole list of mini devices. This is actually a different sandbox than we registered our device in. This is one that has a quite a few devices registered already, but you can go and take a look at some of these, and you can click on them and you can bring up some different data elements about those devices. So for example, this is one running in my office here that I leave running all the time, and you can see I've done some experimenting with it as well. Anyway, that's what the application looks like. You can see that we've implemented a few additional features here, but essentially it's a simple application; presents a list of the devices on this side, provides kind of a dashboard view with some graphing. Some of the data attributes. If you collect history on them, it will plot a history graph for you here as well, and it would show any outstanding alarms, things like that. So, this is what the sample application looks like,
From: 03:02	so now let's go over and, let me close this for now, and then I'll show you the code in the directory, and you'll see that if you go into the "emsdemo" directory, you'll notice that this is a typical Node.js-type application using the Express framework, and we'll look at the code in a minute, but let's just run it first of all, and since it's a Node.js application, we'll simply run it like this, and you'll see that it starts up a server in the virtual machine here, and that server is listening on the local port of 4443. So if I open up my browser now, and I've saved the link to the local one as well, if I point there to the local one, you'll see that it's actually bringing up the same application, but it's running locally now instead of on a server out somewhere else. So let's go back and now take a little look at the code that
From: 04:03	implements this application. I'll close the browser, and we'll come back over here and we'll simply stop the application, and start up an editor to kind of look at some code here. Ok, so let's take a look now what some of the application code actually looks like. I brought up the code here in an editor, and we can notice that when we started it from the command line, we ran a file in the bin directory just called "www", and this you'll notice is the starting point for this Node.js-type application. This is all JavaScript code. And you can see that it's loading in

	some additional modules that are required, and you can see that it's bringing in some configuration data from a yaml file, and you'll see that it eventually starts a server down here. And it is starting an HTTPS server, which means it needs to load in some security certificates and keys, and so that it
From: 05:02	brings those in from that directory and you'll notice those are loaded in over here. And so, once it has started up the server, you'll notice it starts the server running an "app" application, and that is defined here, and that application then loads in again some more Node.js modules that are needed, Express as a framework for Web applications, Mongoose is a utility for interfacing to the MongoDB database, and various other pieces are needed as part of this build. But once the application is ready to go, it defines a number of routes that the browser can make requests into, and so you'll notice up here then that some of these routes are defined, for like the main route, and but then there's "logout", and "profile", and then there's a route for "getDeviceList", and then when those are called, when these routes are
From: 06:03	requested from the browser, you'll notice that it's actually request running an application down in the Wind River API module, and it's running the function "getDeviceList". So the Wind River API module is loaded in from here, and you'll notice in here though its defining many functions. But if we scroll down and you can see that the "getChartData", there's a number of other functions that are defined down here, but the main one that I was going to show you was the getDeviceList, and you can see that in the getDeviceList function here, it's actually going to then make up a post using the Mashery address, and it's concatenating the strings for the actual RESTful API call that it wants to make, and in this case it's asset find, and then it appends the API key which is part of the security mechanism for Mashery, and then it will actually
From: 07:02	update any additional headers before it, it sends the request. But once it sends the request out, it's simply waiting for the responses to come back, and those responses that come back are then parsed, and then presented to a list on the screen as we saw down the left-hand column of the the application. This is where it's actually populating that list. And so there are many other functions actually defined in this file, but you can also see that some custom components are defined as well. And those, if I go back and show you, some of those are defined in the application as Scripto objects, and and you'll see those defined later down here. But you can see there's many other application components that are defined, and I'll let you sort through some of the source code on your own. If you're familiar with JavaScript and Node.js development, it'll be pretty straightforward what you're able to find, and how this
From: 08:03	application actually works. I wanted to point out just some of the core features that you need to look at here, but you'll also notice down in the "js" directory, there's a file called "cloudplay.js", and this is actually the code that runs in the browser, and so when the browser initially makes contact with the server, one of the things that is sent back is this cloudplay.js component, and then this code actually runs in the browser, and when the browser loads the page, it actually invokes a function called document ready, and this is a jQuery type function, so, but it will load up and you'll notice that now in the browser it's defining functions for, you know when certain buttons are clicked, or certain things are happening on the screen, and then it will send the API request back to the server. So, you'll see some of these things going back and forth, and if you recall the diagram where I showed
From: 09:04	that the whole flow of the initial request from the browser to the server, then the responses back to the browser, the browser then loads the cloudplay.js code into the browser, and then is able to respond and act on button clicks and things like that, which will then invoke some of these other calls out, and make requests from the server to then make those Mashery API calls. This code is too detailed for us to try to go through here. It is provided in your environment though for you to play with, and try out, and you can see how the API calls are pretty straightforward, making calls through Mashery into the cloud platform, and then on to, if there's any custom objects to find, you can make those calls as well. So now let's switch over and take a look at what's actually defined in the browser itself.

From: 10:02	And if we go back into our cloud platform, and log in again, and if we come to the Manage tab, and we'll take a look at the custom objects that are defined. And you'll see that we have a couple of custom objects to find, and you'll notice when you look at these, let me scroll this up a little bit, you can define the name of the function, you can define the code here which is essentially, it's called Groovy scripting, but it's essentially Java code, and you can scroll through the Java code and kind of see how that works, and you can actually cut and paste this right into the browser here and then simply come down and click the compile button. But typically you would use external Java development tools with Maven, things like that to, to develop your applications, but this is essentially where these actually go in the
From: 11:03	cloud platform. And so once these are loaded in, then you can actually make RESTful API calls. These are just new commands that you've defined and now you can make a RESTful API call into the cloud platform, and then get the responses back into your custom application that you're developing, and be able to use it from there.

Contact: nlyons

Content ID: 045834

template('WindRiver/function/JS/wrConditionalContent');