**Welcome** to module 12, Custom NetSuite Pages. In this module, we're going to look at how you can create custom NetSuite pages so that you can display targeted information, or if you want to use that to enter information that's not really specific to one record. So that's what we're going to cover in this module.

So here, we're going to look at the script type that would allow us to do that, which are suitelets, and we're going to look at how we can use suitelets to build custom NetSuite UI pages. Now, you're not limited to building UI pages, but that's what we're going to concentrate on because that's the common use for suitelets. We're also going to learn how we can process URL parameters. So this is another way of passing information to another script, and specifically, this would be the way that you're going to use if you want to pass information from a script to a suitelet or a RESTlet, but for RESTlets, we're going to talk about that in the next module.

We're also going to look at how you can redirect a user from one page to another, similar to our go-to-record action, so that you can also compare, this is how you do it in the workflow, and this is how you do a similar thing in Suite script so you can see which one's easier and which one would be more efficient, I suppose. But that's something that we're going to look at. So to begin with here, the important thing that you have to remember is that you have to understand the request and response cycle because for a suitelet, that's something that you would have to think about because that's how suitelets work.

So the request and response is basically, that's just how the internet works, I suppose, where you have a client, and then your client would be sending a request to the server. And that's really how clients and servers communicate. So for an HTTP request-- let's say let's use that as an example-- if you are requesting for a web page, let's say netsuite.com, then your client, say your browser, would be sending the URL to the server, say www.netsuite.com. The server would recognize that request and then would send the appropriate response, which is the web page.

Now, it doesn't always mean that if you send a request, the server would recognize that request. Sometimes, the server does not recognize the request. In the case of a browser or a URL request, what happens is that you mistyped the URL. Instead of netsuite.com, you might have typed in netsuite.cor maybe, or something like that. So if you send that information, the server will not recognize the request, and maybe you will get an error message, such as page 404, page not found, or something like that.

So that is something that you have to remember if you are dealing with a suitelet because a suitelet would have full control to a request and also to a response. So unlike with a client side script or user event script-- where a client side script would be limited to the things that you can do on your browser, and then a user event script would be limited to actions that you actually perform on the record-- with a suitelet, you have full control of everything that's going to happen, meaning if you are going to send information to the server, you have full control in what you're going to do with that request. Also, that means that you have full control with the response that you are going to send back.

So for the response, what you can do is that you can send a NetSuite page as a response. So that allows us to create our own custom NetSuite page just in case we want to display some detailed information or information that you can't normally get in a normal record form. That also means that, since I have full control of the response, I'm not limited to sending a NetSuite page. I can send any kind of information to the browser.

I can send something like a custom HTML page, a full blown HTML page that's not designed by NetSuite. I can also send an XML form. So I can send any kind of information because I have control on the response that I'm going to send. And again, comparing that to a user event script-- if you use a user event script, you are limited to whatever the server would return to you, and again, you only have the before load, before submit, and after submit.

And this is why, with suitelets, suitelets are very popular because of that flexibility. You can use a suitelet to design your own page if you want to use a custom NetSuite page. Or you can use that as a web service, and that's something that we're going to look at in the next module.

Now, if you are going to use a suitelet, these are some examples of what you can do with a suitelet. So you can use a suitelet to create custom NetSuite forms or pages. Again, you can design whatever page you want. You can make it as small as you want. Maybe just have a few fields on that NetSuite page, or you can have a full blown page where you have several fields, and then you have several automation in those fields.

You can also create a custom list, and this is something that is very important if you want to deal with a mixture of records on the same list, because if you want to, say, look for employee records, you're limited to looking at all employee records. If we go to list employees.employees, or if you go to list relationship.customers, the only record that you're going to see is for that particular type. So if you're looking at customers, it's all going to be customers. But what if you want to have a mixture of information?

Maybe you want to have a mixture of customer records and then transaction records that are related to that customer. You can never do that with a normal list, or you can't also do that with a search. Remember, if you are going to perform a search, you're limited to searching a particular type. So when you create a search, it's going to ask you, which record do you want to search for? So that's something that you need to consider.

Like what I mentioned, you can also create a custom HTML page, a full blown HTML design, CSS, and everything. We can also send XML. If you want to use a suitelet for XML, that would be a good use case for creating something like an RSS feed. So maybe you have a company, and you want to publish some of the products for your company so that maybe your partners and some of your fans might be able to add your product list on their website. Then you can just create a public suitelet, and then have that public suitelet available to your partners and your subscribers.

You can also send or use a suitelet to transfer text based information such as JSON. So in case you want to allow your partners or subscribers to request for information from you, you can send them a JSON file instead of an XML file, or just generally use suitelets for web services. Now for this module, we're going to concentrate on the UI side of a suitelet, so we're going to deal with custom forms.

Custom lists are pretty much the same as custom forms. The objects are a little bit different, but the concept is pretty much the same. We're also going to look a little bit a HTML, which is also the same as sending any kind of text based messages. For web services, we do have a separate module in that, and we're going to talk about that after this one.

So a little bit of a comparison between a client side script and a user event script versus a suitelet-- so whenever you create a script for a client side script or a user event script, you always deploy that to a particular record or to a particular form, meaning you're really limited on what you can display on that form. So if you deploy that to a custom record, you are limited to the fields that are available to that custom record. That also means that, in a suitelet, you actually create your own form, meaning you start from scratch. So you can put any information that you want there, and you can leave out any information that you don't want to display on the form, unlike with a client side script or a user event script. They're limited to what you have.

Now, you can control the form, maybe add additional-- or maybe remove some fields if you want to, add fields if you want to. But again any changes that you make on a record, that would also change the underlying database structure, unlike a suitelet, where the form that you're going to create is just mainly for the UI. It's not going to change anything on the background.

Now, this would also mean that, since you're building a form from the ground up on a suitelet, all of the automations that would happen on a client side script or a user event script will not happen on a suitelet form. So if you create a client side script or a user event script, some of the fields would automatically be populated. Now we've seen that in some of the records that we have scripted-- for example, our phone call form, where the status or the organizers automatically set. With a suitelet, those behavior will not happen.

If you want to set a default value, you have to explicitly mention on your script or on your configuration that this field needs to have this particular value. So again you have that flexibility to do whatever you want, but also that means that you are responsible of setting the values and doing everything because the system would not do it automatically. Also, with a client side script and a user event script, this is attached to a particular record. That means you're limited to what record you attach to.

Well, you can attach the multiple records or deploy that to several records at a time, but that doesn't mean that you can combine information from those records. With a suitelet, since this is a standalone page, you are free to get any record from anywhere, so you can combine transaction records and entity records and CRM records. If you want to have like one page that has everything for your CEO-- instead of giving all of the details, maybe you want to create a page or a high level overview of everything so that your CEO can look at that-- you can create that page easily with a suitelet, and that's something that we're going to see once we start going through the walkthrough and on the exercise.

Also, since a suitelet is built from the ground up, that means that you can use custom HTML, which you can never do with a client side script or a user event script. So with a suitelt, you can just create anything you want. You can design like black and red web page if you want to. That's something you can do. But with a client side script and the user event script, you're limited to the UI design.

Now, with a client side script and user event script, we do have a field called the Inline HTML field, which gives you the option of adding HTML tags to your particular page, but that's limited to that small field. It's not the whole page. It's just one section of your form that has custom HTML. So maybe if you want to add custom code, like header code for your SEO or anything like that, so that's something that you can put in on a client side script or a user event script, but again, it's not the whole page. It's just one small section.

Now, here's an example of how you can create a suitelet. Now, for a suitelet, you can see here, building a suitelet is very object oriented programming-wise. It's not like HTML, where you would have to design everything.

For example, if you want to create a NetSuite page, what you're going to do is that you're going to use the UI module. So on the UI module, you're going to use the Create Form-- or actually, this was renamed to Server Widget. Anyway, so you are going to use the UI or the Server Widget module, and then from that Server Widget module, you have a function called Create Form that would create the page for you.

Now, just by using create form, all of the elements or all of design elements that you have on your page would automatically be created by the system. So you don't need to worry about CSS, like what's this going to look like? Do I have to align this to this?

What's going to happen if I do a mobile device? Would it automatically change that? You don't have to worry about that. So all of the design elements would be handled by the system, so this is very easy for us developers, especially if you're not familiar, to front end design.

Anyway, so once you've created your form here, if you want to add fields-- and in this example, we have two fields on the system. So we can add a field using the Add Field method, and as you can see here, with the Add Field method, you also have some options to add your field type, say, if you want to add free form text-- so this text is actually free form text. Or if you want to add a date and time field, you have DATETIME. So you can see here that this is a UI enum value, so you don't have to think about the value that you're going to use. Just use the enum value.

Now we can also change the display preferences similar to what you can do with, say, an entity field. So if you want to set a field as inline, so there is a UI value that you can use for that, and also there is some properties that you can use if you want to disable that, make that mandatory, in all that. So that's something that you can do with a suitelet.

And as you can see here, this is really nice, especially if you're not really familiar with design. I mean, again, some people are really good with design, but some people are not. But at least this would help you get through that.

In this walkthrough, what we're going to do is that we're going to create a suitelet that would allow us to update the employee notes. So instead of updating the employee notes on the employee record, we're going to transfer the control to our suitelet. Then have that update the employee record for us. What we want to learn here is that we want to learn how we can use the Server Widget or the UI module, and also how we can use UI suitelets that would be displaying custom pages.

Now, let's go ahead and build our script, and let's close this one because we don't need that. And let's create our suitelet script. Now, for the suitelet script, I'm going to name this sdr\_sl\_update\_m\_notes. So we are also going to update notes here similar to what we did in our workflow, but this time, we are going to update the notes based on a suitelet.

Now, for my suitelet, I need to make sure that I add the Server Widget module. The Server Widget module allows me to create custom user interface designs just by creating objects, and that's really the beauty of using a suitelet, because I don't need to any design skills. All I need to do is just throw all the objects that I need and let the system design the form for me.

So here is my suitelet, and the first thing that I'm going to do here is that I'm going to create my form objects, so var form, and I can do that by using the Server Widget module. We have Server Widget. Then on my Server Widget, I can just use the Create Form method. Then on my Create Form method, I have two parameters here.

Now, the first parameter would be the title of my form. Basically, here, I'll just name this my Update Employee Notes. Now, the title of your widget would be here, so whatever's listed here, just below your navigation bar. So for example, here in this case, it's Expense Report, or if I move to another page, so here's Script, so this is an example of a title. So whatever you're putting on the title would be displayed here.

Now, there is also another option. So if you want to know more information about the Server Widget module-- let me just expand this a little bit and go to my Server Widget. Now, for the Create Form method, you're going to get this object-- the serverwidget.form object. So if I expand this, you can see these are the different elements that I can add to my form.

For example, I can add a button. I can add a credentials field, or I can add a regular field. A field group would be a grouping of your fields. A Reset button and so on, so I have a sublist. So anything that you want to add, and again, the formatting doesn't really matter because that will be handled automatically by the system.

If you are going to use the Create Form, if I open this Create Form definition, you can see here that it has two parameters. One parameter would be the title, which we've already said, and an optional parameter, which is the Hide Navigation Bar. Going back here, I'm going to add that Hide Navigation Bar just to show you what that looks like. Here, I'll just say HideNavBar so HideNavBar, I'll just give it value of true. And we're going to see what will happen if I use that HideNavBar option.

Aside from that HideNavBar option, going back to our form options here, so I'm going to use-- well, not the Add button. I'm going to use the Add Submit button, to add a Submit button on the form. I'm going to mention what's the difference between a Submit button and Add button, and then we're also going to add a few fields. I'm going to use the Add Field method.

Going back here, we already have our form object, and after creating our form object, time to add our fields. We need three fields-- one field to display the name of our employee, second would be the actual notes that we are going to display, and then a third field, which is something that we can hide, but we're going to use that information to load the employee record, which is the internal ID of the employee. So we need to get that.

Here, I'm going to create a Name field option. I named my variable NameFLD to distinguish from the Name value, whatever the name of the employee is, with the Name UI element, so Name field. It's up to you in how you want to name that.

For some people, what they do is that they use the Hungarian notation. Hungarian notation is used for loosely typed languages, where variable might be anything, Hungarian notation would be adding the data type at the beginning of your variable. For example, here, if I'm creating-- I'll just comment that out to remove the error.

If I'm creating a variable, and then this is a string, I can just say STRName. Or if this is a record, I can say RECEmployee, or if this is a Boolean value, I can say BSettings or something like that. Or some people use Is, like isMet or something like that. It's really up to you in how you want to name that. You can use Hungarian notation, or you can just use whatever notation you are accustomed to.

Now, then, I have my variable, all I need to do is just use my form object, the one that we've created here. And then on that form object, I just use the Add Field method. The Add Field method has three parameters. The first parameter would be the ID of your field, similar to when you are creating a field on your account. So for example, here, if you're creating a field, I'll say CustPage\_sdr\_m\_name.

Now, notice what I did here. When I defined my ID, I added the prefix CustPage. Now, this is something that you have to do if you're creating a suitelet form because on a suitelet form-- unlike, say, a custom entity field or a script parameter or a workflow field or anything that you can customize in the NetSuite UI, the prefix is automatically given. But on a suitelet, since we're building from the ground up, those kinds of automations would not be there. So you need to make sure that you remember to add CustPage, and then after that CustPage, it's really up to you and what ID you want to use. We're going to use just the same ID naming convention.

Then after that we're going to define our type, so type would be what kind of field do you want to add? Do you want a free form text, checkbox? Do you want to add a currency field or whatnot? There is an enum value that you can use as part of your Server Widget, so here, I have Server Widget dot, and then I have my field type enum Then on my field type enum, if I want to add the free form text, that is just simply Text.

Then lastly here, we have our label, so label would be whatever is displayed on our page. For my label, I'll just call this Name. And then since we need three fields, I'll just duplicate this, and then for my field, I will say Notes. I'll just change this to Notes, and of course, change this to Notes.

And then for this one, I'm going to change this to Employee ID, Emp\_ID, and here, the name would be Emp\_ID. And by the way, you need to make sure that everything would be lowercase. No spaces, or else you will get an error message.

By the way, for the Notes field, I don't want the free form text. I want something larger, and that would be my text area. And then for the ID field, I'll just call this my employee ID, or just Emp\_ID.

It doesn't really matter what I use here because I don't think this is something that I want to display on the form. I can if I want to, but that would not really be useful for your user. If there's anything that you have on your form that's not really relevant for your user, I don't recommend that you display that because that will just add additional confusion or complexity to the forum that otherwise could be avoided.

I have my name field. I've added a text field here, a free form text with name, and then I added a text area, so I called it notes. Then I added an employee ID field, which is also a free form text, and then that's going to be Employee ID.

Now, another thing that I want to add is I want to add a button. To add a button, I'm going to use form.addSubmitButton specifically. Note, under the Submit button, all you need to do is just that, just call the Add Submit Button. No other configuration needed.

You can get a reference of this if this is something that you would be changing, like if there's any property that you need to change on the Submit button. You can also store this on a variable, maybe just BTN. But we're not really going to change that Submit button. So maybe just change the name, but we don't really need to store that in a variable. But that does return a field object.

And that's it. So last thing I want to do is that I want to display this in my response. So if I want to write this to my response, I can just go response, which is coming from my context object-- response.writepage, and then I'll write the form object that I've created.

And as you can see, this is really the beauty of using a suitelet because, if you're not a designer, this might be very difficult to create. If you're accustomed to doing some back end programming, maybe server side JavaScript or maybe Java or C# that's on the server, that doesn't really have a UI, then doesn't mean that if you're a developer, you can be-- well, there are a lot of good developers that are also good designers, but that's not the really combination of things that always happens. There are some developers that are good, but not everybody are good designers.

That said, as you can see we didn't really design anything. We just created an object, added a bunch of elements to our object, and then wrote that in our response. No design required.

One other thing that I can do here is that I can store this in a response object if I want to, and that might be something that-- let's go ahead and do that because you can get the response and request object from your context object. So I'll do that because those are things that you would probably access a lot. So request, and then var response is equals to context.response. That looks good. And we change this so that it only would be showing response.

Let me upload this and then create our script record. To create our script record, we can go to Customization, Scripting, Scripts, New. Again, the process is pretty much the same. If I go to my suitelet, create my script record, and then for the name, I'll call this SuiteDreams SL Update Employee Notes.

Then for the ID, pretty much the same thing-- sl\_update\_emp\_notes, and it can copy that. Put that in my deployment here before the name, right over here. Then for the ID, I can also copy, put this here at, and then save.

And that is it. We've created our script record, but how does our form look like? So let me open-- what does it look like?

Here, if I open my script deployment, you can see that I have a field here called URL, and this is how I can open my suitelet page. So if I right click this, open this is a new subtab, and you can see this is what my form looks like. So I have my Update Employee Notes. That's my title. And since I decided to hide the navigation bar, I don't have that navigation bar, which is-- I have to say it's kind of not good, so maybe I can put that navigation bar back.

Also, there are a few things that I want to modify with my form. First thing that I want to modify is that I don't want the name to be editable because I don't want the user to accidentally change the name of the employee. So that's not something I want to happen.

Also, again, the employee ID-- we need that employee ID, but we don't really need to display that on the form, so maybe we can go ahead and hide that. Also, one last change that I want to do here is that, instead of using Save, maybe I can just use Continue. Because I'm going to move from this suitelet to an employee record, I think Continue would be a better name for the button instead of Save.

So those are the things that I want to modify, so going back to our form here, for the hide navigation bar, this is optional. If you remove this, that would be the same as setting it to false. So instead of setting that to false, just go ahead and remove that because you don't need that. That's one change.

Now, another quick change that I want to do is that, for my Submit button, I want to change this from Save to Continue. So all I need to do is just change that value to Continue, like this. I have my Add Submit Button, and then also, I want to change how my name field and also my employee field displays.

My name field, I don't want that to be editable. I want to change that to inline. And then my employee ID field, I want that hidden because I don't need that on the form. That said, I'm going to add those changes here.

So for my name field, I'll just get that name field record, and then I have a function or a method called the Update Display Type. For my Update Display Type, I have one parameter or property here called the Display Type property, and that would tell me how do you want to display this particular field. I have different options, and this is an enum value, so I can just get that on my Server Widget module. And then I have my Field Display Type, and then inside my Field Display Type, I have different options.

Now, for this one if, I don't want my field to be editable, I can just set that to inline, and I also want to do this to my Employee ID field, so Emp\_ID. But instead of setting that to inline, I'm going to set this to hidden, and that's why I want to do. That's it. A few quick changes, and let's see how that would affect our form.

So let me upload this and go back to my suitelet and just refresh this by pressing F5. I don't need to do Control-F5 because this is happening on the server. As you can see, that looks better.

I have my navigation bar back. So I have my button, so my button now says Continue instead of Save. And then I also have these two fields, so I have my Name field, and I also have my Notes field. I don't have my Employee ID field because that's something that I really don't want to display on the form, but in this case, I still need the value. I just don't want to display it.

Now, earlier, I mentioned that there are two kinds of buttons that you can use. You can either use a Submit button, or you can use a regular button. Now the difference between the two-- so let me try to open an employee record. Let me see if an employee record has multiple buttons. There you go.

So as you can see, I have a button here that's colored blue, so any blue button or any colored button-- so it could be blue, yellow, depending on the color scheme that you used for NetSuite-- that would be your Submit button. Every time you click on the Submit button, what the system would do is that they'll gather all of the data that you have on your form and then transfer that information to the server. Regular buttons are like these two. I have my Cancel and Reset, and that would depend on what kind of automation you attach these buttons. This would not do anything unless you add in a function to those buttons.

You can use these buttons to submit. That's something that you can still do, but again, you have to make sure that you add the implementation to submitting that form or doing whatever, unlike with a Submit button, where the only thing that I did here was this, and it already works for submitting. That is the difference between the two.

Also, something that you can do, going back to my script deployment here, is that you can add a link to your suitelet from anywhere on your menu or your navigation menu. For example, because here, we're trying to open our suitelet from the URL, but what if I want that maybe here, how do I do that? Or maybe add that in my documents or whatever?

If you want to add a link to your suitelet on your navigation bar, what you can do is that you can go to your script deployment, edit your script deployment, and go to the Links subtab. Here, if I scroll down, I have my Links subtab, and this is where you can add links to your suitelet. You're not limited to adding just one link. You can put that anywhere in the navigation, but first, you need to know which center you're using.

Right now, I'm logged in as the administrator, and administrator role uses the classic center. You can use the counting center, the partner, customer center-- really doesn't matter. But for me, since I'm using the classic center from the administrator role, I need to make sure I select that.

Once I have the classic center, it's going to ask me which navigation path do you want to use? For example, the section, do I want to put that under transaction or list, reports, or whatnot? Now, maybe for this link, since it's related to an employee, maybe I can just put that below the employee. That works. So list employees, then employees. For my section, I'm going to choose list, and then for my category, I need to put that under Employees.

Choose that, and then for the label, I can just use whatever label that I want. If I don't add in the label, it'll use the title, and my title is not really that user friendly. So maybe I'll just put in a more user friendly title. I'll just say Update Employee Notes.

And then I can just choose where do you want to insert that. So insert before I believe-- let me just check-- before Work Calendar so that it's beside the employee's thing. So insert before Work Calendars, click on Add, and then click Save, and then that's it.

If I go to list employees, then I can see here that I have my Update Employee Notes page. If I open this in a new subtab, you can see that I have my page working. But right now, it's not really doing anything yet if I click on Continue.

As you can see, it's just not doing anything because this is just an empty shell. We haven't really done anything to our suitelet yet, but we are just displaying that on our page, and that's what we are trying to do. So in the next walkthrough we're going to update this to complete our automation so we can get information from our employee record and then use that to update the notes of our employee.

So how are we going to process requests? So basically, what we did in our walkthrough is that we just built a shell. We built a UI. That UI doesn't really do anything. So what we want to do is that we want to be able to control what's going into our suitelet and also control what's going out.

Now, we've already seen what's going out. It's a form page. But let's see how we can get information from our request.

Now, as you've seen in the deployment record, there are two kinds of requests that you can get. You can get either a Get request or you can get a Post request. Now the difference between a Get request and a Post request would be how the information is passed on to the server. So for example, here, we can see that the information on a Get request is actually passed on to the URL as a query string.

So normally, and this is something that you would see a lot on the web pages, so if you want to add information, or if you want to include certain request information to your URL, you would put in a question mark and then add in a key value pair separated by an equal sign. So for example here, you have City is equals to Denver, so the request information or the parameter information is City, and then the value of that parameter is Denver. If you want to add more parameters to your URL, then you would have an ampersand, and you can use that as a connector. So here, we have an ampersand and then State.

That's my parameter, then the value is CO Colorado. Then you have another ampersand, and then you have an address. So that's what you have here.

Now, on a Post request, this is used if you want to submit a form. Every time you submit a form, regardless of which web page you're using, it's always going to use a Post request, so that's how it works. What happens with the Post request is that, instead of embedding your request information or URL parameter information on the URL, it's going to package that up and then include that in the header information-- or in the actual request. So you have your header information, then you have your body, and it's packaged up in one HTTP request, and then that's sent to the server.

Now, if you think about it, if you are going to use a Post request, that would be used typically if you want this do things like logging into the server. I mean, if you want to log into the server, you want to make sure you don't use a Get request because people would be able to see your username and password just by looking at the URL or the address bar. But with a Post, since that is packaged up before sending to the server, that's something that nobody would be able to see, at least not easily because, again, it's packaged up before sending to the server.

So there you go. So here is an example of processing a Get request like what I mentioned. Every time you open a link, either clicking on the link or typing on the URL, that would be a Get request. But also, this is something that is important to remember as a SuiteScript developer, if you are forwarding your request from a script to your suitelet, it's always going to be a Get request. Any redirect from a script to a suitelet would be a Get request.

Now here, if we process a Get request-- and what we've done so far is that we tried to open a link to the suitelet that we've created. So that means that we are processing a Get request, which allowed us to write our field information. And then we have our Submit button, which we renamed to Continue.

Now, if the user clicks on the Submit button-- so remember, you have a form, and you're submitting a form because you're clicking on that Submit button-- what will happen is that will be treated as a Post request. Now, in our case, at least in our walkthrough, what we're going to do is that we're going to redirect that user to the employee record and then have the employee record do whatever you want to do with that employee record. So again, quick review here, any suitelet opened on a link, a URL, or if redirected from a user event script or a client side script or whatever script, that would be treated as a Get request. And if the user clicks on the Submit button of any form, including the form that's on your suitelet, that would be treated as a Post request, and then you can process that information as a Post request.

Now, why am I stressing this information? I am stressing this because we are going to use that information to distinguish what we're going to do on our suitelet. If the user submits the information-- at least for this scenario, it's not going to apply to all scenarios, but I'm just saying that, in our scenario, this is what we want to do. So here, if the user redirects from a user event script, or if the user opens our record, we want to display the form, so that's what we want to do.

Now, if the user submits the form, we want to update the notes of the employee record. So making that distinction, we need to make sure we add in an IF statement in our code to say IF it's a Get request, display code. Otherwise, it's a Post request. Process the employee record.

So again, that's just for the scenario, and it's not always going to be the case in all the suitelets that you're going to build. For some suitelets, you're only going to use Get. For some suitelets, you're going to use both Get and Post. For some suitelets, you're going to use only Post. So you have that option.

In this walkthrough, what we're going to do is that we're going to continue with what we were building in the previous walkthrough. So here, we're going to get information from our employee record as soon as the user saves the employee record, we want to get redirected to our suitelet form, and then grabbing the information that we need, the name, the current notes, and also the employee ID, transferring that to the suitelet. Then after updating the suitelet, the user would click on Continue, and then the suitelet would process the information in the background and update the employee notes, and then redirect the user to the employee record to verify if the notes were truly updated. What we want to learn here is that we want to learn how we can redirect to and from a suitelet and, very importantly, how we can process URL parameters and then use those URL parameters for submitting your information or getting information from another script.

Since we want to grab the values from our employee record, we're going to reopen our user events group for our employee record. And here at the bottom of the our After Submit, I'll just put that outside of our IF statement so that this would get triggered regardless if they're creating a new employee record or editing an existing record. And what we want to do is that we want to transfer information from here to the suitelet.

Now, to do the redirect what we can do is that we can add in the redirect module. So there's a module that is specific for moving users from one record to another, similar to what we did in the go-to-record action. So let me go ahead and add the redirect module, so I have n/redirect. There you go, and then I'm also going to add that in my definition here, redirect. And of course, I want to add that to my annotation here, so Add Param, redirect, and the variable name that I used is also, redirect.

Now, going back here, the only thing that I need to do is just add that redirect to my suitelet. So I have my redirect method, and then I have a method call that's called the toSuitelet. So that toSuitelet would allow me to not only redirect from my user event script to my suitelet, but it would also allow me to pass information from this user event script to the suitelet.

So here, I have my script ID field, and that's one property that I need to set so I'm just going to grab that in a bit. But also, I have my deployment ID. Again, something that I'll grab in a bit, but what we're interested in here, with the R parameters property, so parameters, and that parameters would be the object that you're going to pass to your request. So all of the information, such as the name, the notes, and the employee ID would be passed from this section.

Now, for the parameters, you can use any parameter value that you want. Just make sure that you avoid the standard parameters that you would see in NetSuite. For example, do not use ID because you would see ID a lot.

Here, you can see I have ID A. Do not use whence. Not really quite sure where that's used, but don't use that. Don't use type. Don't use things like that.

Now, instead of just avoiding that, what you can do is that you can name this a little bit differently. For example, you can use custparam\_sdr\_name, and this obviously will not do any collisions-- or it will not cause any collisions because nobody would really name this in NetSuite. But this is a little bit too long, so what you can do is that you can actually just shorten it to sdr\_name. Again, the goal is that you do not want to cause any collisions, And you can prevent collisions by making sure that it is uniquely named.

Now for us, again, I'll just call this SDR. You might want to add a few more things here, especially if you're using a lot of parameters, such as maybe adding a module name or something like that. It's really up to you.

Anyway, I want to get the name of my employee, so I can get that from my employee record. Again, just to review, we already have our new employee record here from our new record property inside my context object. This is the same After Submit that we have been editing since we've started the course. So here, so employee, and I'm going to use the Get Value method.

So for this one, this would be my entity ID. Then after that, I also need to get my notes-- sdr\_notes. So employee.getValue, and if you remember, we already used this in our script parameter for our workflow action scripts, so that'll be comments. And then for the employee ID, sdr\_empid, and then here, that's employee.id.

That's it. If you want to get the internal ID of the record reference, just to use ID, and that is pretty much it. Of course, let me go back to my script record and then grab my ID. So this is my script record ID. Let me get that and then paste that here.

And then go back for my script deployment ID. Copy that. Paste that right over there.

That's it. Make sure I remove the spaces. Otherwise, I will get an error for that. There we go.

So let me upload this, and let us see if it works. So if everything is working, what should happen is that, once we save a record, that record would redirect to this particular suitelet, and then it should pass the value for my entity ID or the employee name, employee notes, and also the employee ID. So let's see if this works. Let me go to an employee record. Let me edit this.

Can use any record. Doesn't really matter. Let's use Cameron Murdock, and then for Cameron Murdock, I'll just say I have notes from my record, so that's something that I want to load and something that I want to display on my suitelet. So let's see if this works.

I'm going to click on Save, and so it looks like we were able to redirect to our suitelet. But if you look at our form, I'm not seeing my name, and I'm also not seeing my note. So were we able to properly pass the value from our user event script to our suitelet?

Now, you might be thinking, no, it didn't work. But it did actually work because, if you look at the URL, so here we have sdr\_name, which is the name of our employee, which is Cameron Murdoch. We also have our notes, notes from record, and then we also have our employee ID, which is 14.

Now why did it not put the value on the form? Remember, we are only passing the value to our suitelet, and unlike a client side script and a user event script, where everything would be automated, since we're building everything from the ground up, you need to make sure that, if you're getting data from another source, you need to grab that data and then put that on the form. So that's what we're going to do.

So going back here to our suitelet, and then on our suitelet up top here, I'm going to add in a few more variables. So I have name. Then on my name, I just need to get the value from my request. Then on my request object-- so remember, you can get that from your context object-- I have the parameters object.

The parameters object would be this, so we're trying to grab this information. Then after I get that parameters object, I know that, inside that, I have three properties-- sdr\_name, sdr\_notes, and sdr\_empid. Request.parameters.sdr\_name, and then we'll just duplicate that, so I have name, notes, and I also have my Emp ID. There you go.

So let's change the name here, so Name, Notes, and EmpID. And you have to make sure that you spell everything correctly because if you don't, with the same case-- so for example, here, I used empid with all lowercase. You don't have these all lowercase. You can use mixed case if you want to, but I've already used lowercase, so I'll just stick to that.

Anyway, there you go. So we were able to get the value now from our request, but again, we've just gotten the value. The next thing that we need to do is we need to make sure we set that value on our form, so add that to my Name field, Notes field, and then EmpID field object references. So I'll put that right over here.

Now for my Name field, say Name field, and I have a property called the default value property, and all I need to do is just set that default value, for example, for my name again, just duplicate that because it's pretty much the same. So I have my name, my notes, and here, I'm setting my notes field. Then I also have my empid, and then here, this would be my empid.

Let's see if I am using the right references. I'll just use the highlights. That looks good.

Notes-- that's highlighted. Then empid field-- that looks good. So name-- there you go. Notes-- that's highlighted, and then empid. Want to make sure everything is properly spelled so that we can check if it works.

That looks good. Let me upload this, and let's see let me go back to my employee record. I'll use the same record. I'll just use Cameron Murdock. Let's refresh this to make sure that everything is working properly. Let's see-- let me click on Save, and if everything is OK, we should see the information on the form. And that's exactly what we have.

So here, we have Cameron Murdock, and then we have notes from record. And then you can see the URL parameters are still here, and as soon as I click on Continue we can see that we have undefined. So why do we have undefined? Because we haven't really done anything to our suitelet.

Now, remember our discussion between the Get operation and also the Post operation. Whenever you are transferring information from another script to your suitelet, that would automatically be a Get operation. Or if you are clicking on the link or opening a bookmark, that will also be a Get operation. But this time, we are submitting the form, so that means that we are moving from a Get operation to a Post operation.

So with that in mind, what we can say is that, if we are moving from a user event script to our suitelet in a Get request, then I want to display my form. Otherwise, if I'm submitting my form, then I don't need to display my form, but I need to redirect back to my employee record. So what I'm going to do here is that I'm going to add in a condition. So add in an IF statement here to say IF-- if my request method, so request.method, is equal to Get, meaning if I'm performing a Get operation, then I want to display this.

Grab this and then put that inside our IF statement. That looks good. Otherwise-- so that'll be an ELSE-- then here, we just add in, this would be my Post operation. So if it's Get, meaning display my page. If it's Post, what I want to do is I want to grab my employee notes and then the employee ID.

So here-- by the way, I need to load a few more things here. Since I need to load the record and redirect back to my employee record, I need to add in my n/record module to load my employee record, then to redirect to my employee record, I need to add my n/redirect. So the n/redirect is the same redirect module that we used in our user event script.

So Add Param, and then we have record. There you go, record. And then Add Param Redirect, Redirect. There you go.

So going down here, so what we want to do is that we want to load the records of our employees as equals to record.load. So here, I have a type, and we want to load our record.employee-- or type.employee. And specifically, we want to load ID here, my empid.

So where am I getting this EmpID from? So we're going to talk about that in a bit, but the values that we need would be my EmpID, which we don't know where we're getting yet, and my notes-- again, something that we don't know where we are getting yet. But let me just go ahead and do the things that we do know. We do know that we need to update our comments.

So set the value, and this would be my comments field. And then we're going to copy the notes from that, and of course, I want to make sure that I save my changes to the database, so employee.save. And also, we want to redirect from our suitelet back to our employee record, so that's redirect.

And instead of using two suitelets-- remember we use here two suitelet to redirect to a suitelet. This time, we're going to redirect to a record, so toRecord. And instead of asking for my script ID and also my deployment ID, it's going to ask me for the type, so here, type, and that'll be record.type.employee.

So I want to redirect an employee record. Specifically I want to redirect to my EmpID. That's it.

So that's what we want to do, and again, these are things that we already know. For the redirect to record again, that's very similar to what we have used in the toSuitelet, but again, the two things that is very important for this to work would be the employee ID and also the notes. One thing that you have to remember here is that, in JavaScript, there's no such thing as a block scope, and this might come as a surprise to people and say-- I'll create another variable called employee ID.

But in fact, the employee ID here would be the same employee ID that we will be using here. So if I don't add invar, and I'll just say empid in the notes, this would still work. This would actually refer to this. As you can see, if I highlight that, it already picks that up, and that's because, again, JavaScript does not have any block scope.

So it doesn't really matter if you're declaring this inside your ELSE statement. It's still going to be treated as the same function here. You don't need to put this outside. What some people would think is that I need to put that outside my IF statement so that it's accessible anywhere inside my IF and ELSE statement, but that's not really the case. So again, that's just a behavior that you need to think about in JavaScript.

That said, where are we going to get the value of our employee ID and also our notes? We can get those values from our form. Now, remember, we're no longer getting the values from our URL parameters, or we're not getting the values from our user event script. Instead, we're getting the value from our form.

Remember here, we are submitting the form. So we're getting the value here from the name and also from our notes, or from our hidden employee ID field. So whatever the user types in, those values should be used by your suitelet.

Now, since those values are going to be used by the suitelet, we're no longer going to use this parameter name. So it's not sdr\_name, but we're going to get these names. So to get the value of the notes, we're going to use custpage\_sdr\_notes, and we're also going to use custpage\_sdr\_emp\_id.

So going down here to our request, I'm still going to get that from our request object because that is submitted to our request. So that's request dot parameters dot, then let me grab the ID. I'll just copy that to make sure I don't misspell that. And then I'll also grab my Notes field and then put that right over here. Of course, I need this.

So again, make sure that you remember that, if you are using a Post request, you will be getting that from your form. Post request from the form. Get request from the URL.

So that's it. Let me upload this. Want to make sure that I update both, so I'm going to select my suitelet script here, and also my user event script and upload both files. So far, what we've been doing is that we're just updating one at a time, but let's just try to update two files at a time.

So anyway, let me refresh this. Let's see if it works, and let me click on Save. Then after clicking on Save, we have our suitelet, and let's try to update this just to make sure-- notes from suitelet.

And if everything is working properly, as soon as I click on Continue, I should go to-- looks like undefined. So let me go ahead and look at my script here. It looks like I have misspelled the parameters, so that's parameters. There you go. Sorry for that mistake.

Let me upload this again, and let's see if this works now. Let me go back here and click on Back. So again, refresh this, and let's try it again.

So let me click on Save, click on OK, and so far so good. I'm going to add notes from suitelet, and let's click on continue, and let's see. Record not defined.

I keep misspelling things. I do apologize for that. Record-- oh, I forgot to add my record here and also my redirect. There you go.

Let's see if I forgot anything else. I do apologize for that. Click on Back. Let me try that again.

Click on Save, and again, notes from suitelet, and I hope this time it works. Let's see, and there you go. So it looks like we were able to redirect to our employee record. So as you can see, we have the notes from our suitelet.

Now, going back to our code, just do remember, if you are going to create a page, and in this case, if you're creating and submitting a page, make sure that you filter based on the execution. If it's a Get method, then maybe you want the display your form, and if it's a Post method, you want to process your data. Also something for you to remember here, aside from not forgetting to put your modules in also, making sure that everything is properly spelled-- make sure that you're getting the values off of your form inside your Post.

So unlike your Get request, where you're getting that from the URL, for Post, you need to get that from your form. And again, this is something I keep repeating because this is one of the common mistakes that people make. So they just copy whatever is on here and then reuse that on Post.

So that's it for this walkthrough. So there you go. So a few things to remember.

If you are going to create a UI, remember that links and script process requests goes to a Get request, and whenever you're submitting a form, it always goes to a Post request. Also the field ID, this is something that you would need to put in. Make sure that it starts with custpage.

Now, in SuiteScript 2.0, it's no longer required. In SuiteScript 1.0, I believe it still is. If you don't put in custpage, or if you use mixed cases, you will get an error message, so I would recommend you do the same thing with SuiteScript 2.0. Always prefix your fields with custpage, and always just use all lowercase. Be careful of spaces, so if you have a space before or after or anywhere in between, you will get an error message when you create your field.

Now, in terms of the exercise that you will be doing, you will be doing similar to what I did in the walkthrough, except you're going to use a sales order record. Now, since we already have a redirect on our accounts using our go-to-record action, what you want to do is you want to disable that workflow if you haven't done so already, so go to the workflow. Put a checkmark on the inactive. So just make sure that the workflow is not running. Otherwise, it would conflict with what we're going to do with the suitelet.

So what you're going to do here is that you're going to open a sales order record, and after you open the sales order record, it's going to trigger your After Submit on Save, where you're going to collect information such as the customer date, order number, and also the total. And then you're going to display that on the form, including a special field called the financing price. Now, we want the user to update the financing price information from the suitelet instead of doing that from the sales order.

So once you are on the suitelet, the user would click on the Submit button, which would trigger your Post request. And once you trigger that Post request, you're going to load the sales order record, update the financing price information, save that the database, and then load and go back to your sales order. Now, you can put in additional things, and by the way, that's something that you need to remember that would happen.

If there are any scripts that are deployed on your sales order, that would also be triggered if you are redirecting. So if you are redirecting say from your suitelet to a sales order, if you have any scripts that are on your sales order that is running on Before Load-- only on Before Load-- then it would also get triggered. So at least be mindful of that behavior.

Now, for our exercise, we're not really going to use the before load script, but just do remember that would happen if you are going to do this in production. Now it's your turn. Please complete the following activity. This includes module 12, Custom NetSuite Pages.