SHANE WEMHOFF: Welcome, everyone. I'll get us started here. Shane Wemhoff, with Behlen Manufacturing. I am in the process improvement team leader of our buildings division. Trent works as our PLM administrator. I'll give you a brief background on Behlen and what we do.

We're located in Columbus, Nebraska. We have facilities in Baker City, Oregon, McGregor, Texas. Our home headquarters is Columbus. We have a company in Omaha, Nebraska, Hilton in Florida, Sarasota. Then we have offices in Boise, Idaho and Lublin, Colorado.

We're kind of diverse. We have a livestock equipment group. So if you've ever been to Tractor Supply or if you know what Tractor Supply Bomgaars is, the big tanks and the cattle handling equipment and all that kind of stuff, we build that. Other things that we build are green bins, and we're in the pre-manufactured metal building industry. That's our division.

So along with that, we purchased PLM about three and a half years ago to help integrate our business, get rid of paper. And after we done that, we had an ERP system where we were entering information. And we had an MBS system also, what does all of our engineering design. So PLM in the middle, we found we were doing a lot of multiple data entries.

So with the data entry, you can have mistakes. So if you're entering a date in Fusion, which also we need that date in our ERP system, which is JD Edwards, people weren't updating that, so our work order weren't up to date. And we needed Fusion, which was our new way of tracking or building through the system to all match the same thing. So that, therefore, came our integration with Jitterbit and some of the scripting.

Some of the other things was we had a lot of workspaces that we needed over and over again. So instead of trying to have the users remember to go make them workspaces, we were spawning them as we were pushing workflows.

So it's a backdrop of what Trent is going to be presenting. So with that, I will let Trent get started.

TRENTON R. EARLEY: Thank you, Shane. Hi, everybody. So today what I want you to take away from this class when we're done or I guess what I'm going to go over is the scripting inside the workspace, pass data back and forth from workspace to workspace.

Show you how an auto transition from one workflow. You can push a workflow transition into another workspace. Conditional scripting, that basically just allows people to push a certain workspace. If it's assigned to them, it shows up in Outstanding Work, and it keeps people on track then so they know when they have something to do, and they won't forget to do it.

Integrating data with Jitterbit. Is anybody familiar with Jitterbit? Anybody using Jitterbit? Couple? OK. So that's what we use to integrate our data in between our ERP system and our Fusion tenant. And we also have a few web pages that we have built to create reports and stuff like that, just advanced reporting that we can't do within Fusion.

An HTTP requests, that's the API request that we use to call Jitterbit to pass the data. And then the scripting within Jitterbit, there's a few. You can kind of wash the data a little bit with scripting inside Jitterbit itself, so I'll go over that a little bit.

So with conditional scripts, that's what we use to make sure that people are on track. They don't forget they have something to do. It shows up in Outstanding Work when something is in there. Basically, when it's assigned to them, it will show up in Outstanding Work, and that's done with conditional scripting.

The validation scripts, I'll go over those too. Those will basically allow something to happen inside the workspace as long as all the conditions are met. So if you need a field filled out at a certain point in the workflow, then the validation script will make sure that that has been met before it will allow the transition to be pushed.

I'll go over action scripts or how you get and set data in between two workspaces. It's basically any actions you need taken or done with the action scripts, I'll go over those. And the milestones, I'll go over to how we set our milestones in our tenant, and it works pretty well for us. So I'm going to go over that with you.

So what I want you to be able to do after this class is over as pass data in between two workspaces on the same job, and I'll show you how to relate workspaces in order to do that, allow certain users to push transitions. Use library scripts. So if you have a script that is a basic script that you use constantly, I'd recommend saving it as a library script, and then you can just call it through of a method.

HTTP requests, we call Jitterbit. I want you to be able to get, post insert upsert. Upsert is actually just something that is specific to Jitterbit. They have an upsert. That's basically an insert and an update in one request. So I'll show you how to do that.

Build the operations inside Jitterbit. I mean, that's basically just how you would call the request and the response, then map all the data in between. If you're calling with Fusion, map all the data from Fusion into your database you're putting the information into and then insert/update information in Jitterbit.

So I'll start with setting data inside the workspace with JavaScript. So what we have here is basically the user end on the admin side. And Field ID is going to be what we will use to call the field to get the data that's stored inside that field. So with this one specifically, we have the job number here and then inside on the admin side, then the job number, this job underscore number, would be how we would call that field to either get the data or set the data.

So what I'm going to go over first is dot notation, and it's how we have the scripting set up inside our tenant. There's other ways to do it, but we're using dot notation. And the notation is basically referring to the period that will separate the items that you're trying to work with. So I'll show you that in just a second.

The Field ID was what I just showed you, and that's how you're going to reference the field that you're trying to either get or set the data in. And so here, it would be on this side. So the dot notation in that last line I showed you was the job number. So when you're calling this ActionScript inside the workspace you're calling it in, it will be referred to as the item.

And the dot notation would be the period here is what's referring to, but the dot notation would then call the job number in that Field ID so that with this line here, I'll be able to reference that data field, and then I'll be able to manipulate the data or get the data or do whatever I want. But that's how we're going to use the scripting inside Fusion to get access to data.

And here's just another example of dot notation. You can see that there's two here, so you're working your way back from this addBom function in the boms workspace, I guess, in the item. This would be the workspace that's calling this action. And inside the curly braces will just be where your code would go for the addBom.

And to declare a variable, it's just the var. And so with this line here, I have a variable named today, and I'm calling the date function, which would be-- it's a built-in function inside JavaScript. And I'm setting it to today's date by declaring it a new date. So with this line here, all I'm doing is getting today's date. And the next line below it, through dot notation, I'm setting the ship date that's the item that's calling this ActionScript to today's date.

So I hope I'm not going too fast for anybody or anything. So this is how we'll create a new workspace and relate it to the workspace that you're in. So how we use it on our side is we have related workspaces, and this is how you're going to pass data back and forth in between related workspaces or related items on the same job.

So with our tenant specifically, we have-- I'm not going to throw out all our workspaces, but say we have a scheduled process, a design process. And project central is basically the center of the whole job. Keeps a lot of the information so that we can have an easy reference to all the information on that job. So with those three workspaces, we relate them to each other so we can pass the data back and forth, and it it's just how we have ours setup to easily pass data back and forth.

I'm not going to go through all this code. I'm sure some of you have questions, and I'll leave some time at the end to answer some questions. And if not, we'll be at answer bar after this. We can get together and we can help you guys out.

But I'm just summing up here what we're doing. We're creating a new properties variable. We're passing all this data from the workspace that's calling this, and then we're calling the Create New Item, which is the function that we have over there. We're passing all the information from this item. We're calling the Create New Item function, and we're passing that data to it to create a new workspace, and then we're setting it as a field inside the workspace that's calling it.

So again, I don't want to get too detailed into the code itself, but I just want to paint a picture of what you can do with Java scripting. And so on that last slide, we had-- let me go back. We had this ws_projects_central, and basically that's our project central workspace. And so we get the field workspace ID to create that new workspace on the admin side when you're creating work spaces, so this is where you get that line to call that workspace when you're creating it.

So we have our related workspaces at the bottom of each workspace, and that's where we have ours set up. It works well for us. But this is basically just how they'll show up when they are created and then related to the workspace that you're in. So here you can see that we have 1, 2, 3, 4, 5, 6 related workspaces in this one workspace itself. So this is just how we have ours set up, but this works really well for us to pass data back and forth and keep all the data in sync and easily find data that's in one workspace, in another workspace. Its convenient to access all the data.

So to create a related workspace, this is a picture of the admin side of when you're creating an item inside of a workspace. And this specifically is going to create a field that will hold the related workspace.

And so as you can see, the Field ID is the related drafting, so this is our drafting process. And to do this, you create a pick list, so you can see here the data type with your pick list. And the Pick List options will show up below it, and that you do in the Pick List manager. You create a pick list and you'd set it to the workspace that you're trying to set as the value in this pick list.

So I won't do a demonstration of how to create a pick list right now, but if anybody wants to get together again after this, we can go ahead and I'll show you how to do that. So created it as a pick list. When you have the pick list created, you'd find it here in this dropdown list and you just said it has a single selection. And we set our display as Link, that checkbox there so that you can easily just click that and it will take you to the next workspace because otherwise it would just be the item descriptor and it wouldn't be very useful.

So at the bottom of the workspace where we have our related workspaces set, it says Link, so you go down, you click that if you want to get to say you're in the drafting process you want to get to the schedule process, you go down and find the related schedule process and you just click the link, and it will take you there. So it makes easy access to all of the workspaces on one job.

So when you're in one workspace and you need to get one quickly, that's related to that same job. You'd have them in your later and workspaces. There you'd go down to the bottom, click the link, and go to that next job to put in information. So again it's just how we have ours set up. It works well for us.

So now I want to get into a little more dot notation with passing data back and forth or getting data from another related workspace. So again, the periods there are separating each item. In as you can see here, we have the job number in the related drafting process. That's related to the item that you're in when you run this script. And so with this line of code, I'm setting the job number in that related drafting process to X0123. It's just a regular job number we use for our tenant.

And when you have related workspaces, so say specifically here I have a related drafting process and a related schedule process. So what I'm showing here is we have a scheduled process that has a related drafting process that has another related workspace that we're in when we're running this script. And so you can see you're just going down the chain of related workspaces. You can work with a workspace that's related to another related workspace.

We use this quite a bit, but a lot of times we have that related workspace already in that, but sometimes we don't have them all related to each other. So this is just working its way down the line, but as you can see here, I'm setting the ship date in the scheduled process that's related to the drafting process that's related to the item that I'm in when I'm running the script. So I just wanted to show that just to show that you can work your way down through related workspaces with dot notation.

So what I've learned just with using JavaScript inside the Fusion tenant is when you're setting data in a related workspace, make sure that it has a matching data type. They have single-line text, which if any of you are familiar with programming, that would be a string. So if you have a single-line text field, you cannot set an integer or a date in another field. So if you're setting data or getting data and taking it back and forth, you want to make sure that the value types are of the same type.

Derived fields. I don't know if any of you are familiar with derived fields at all, but derived fields, basically, when it's in one workspace, you set that value. It will be automatically set in the other workspace. You cannot set derived fields when you're accessing the data through JavaScript. So make sure if you're trying to set data or even get data, make sure it's not a derived field.

So permissions. When you have your permissions set up inside your tenant, the scripting will ignore all permissions. So a person doesn't have permission to set data in another workspace or get the data in other workspace. The scripting will ignore all that. So you want to be careful with that too if you want to not allow certain users to access the data, then you want to make sure that you're not scripting in a workspace that you don't want somebody to get that data.

And keep it simple. I mean, if you're getting too elaborate with your scripts, you might want to rethink your process and maybe find a different way to accomplish what you're trying to do. You want to keep it pretty simple, I mean, 10 lines, 15 lines max, maybe. So don't make it too hard for yourself because a lot of times, if you have your scripts that are too long, they'll run too long and sometimes it will cause an error, time out, or something like that. So just keep it simple.

So now I want to get into scripting inside the workflow. What we use quite a bit is conditional scripts to allow certain users to push workflow transitions. This is good to make sure someone's not doing something they shouldn't do, but it also is good to show up in people's outstanding work too. So that if somebody has something to do, they don't forget to do it. It will be right there in their dashboard in their Outstanding Work. So you do that with conditional scripting.

Validation scripts. We use these quite a bit as well, these specifically to our tenant. If we have a date that we want set inside of a workspace at a certain point in the workflow or name put into a certain field or something like that, a certain point in the workflow, we've used validation scripts to either allow or disallow someone to push that workflow transition until that condition is met inside that workspace.

Action scripts, you can run these from workflow transitions as well. And that was basically what I was showing you with the dot notation. And when these scripts run, they run in order of the condition runs first to check to see to make sure everything has met all the conditions. The validations will run after the conditioned scripts, and the ActionScript will run last. And time outs for the scripts, it's nine seconds for action validation scripts and four seconds for condition scripts.

So what this is is a condition script, and what we have set up on this example, specifically, is since my name is in the Customer Service Representative field, I am allowed to push this transition to closed. If my name was not in that field, I wouldn't have that option. So I'll show you the code of how we accomplish this.

We start with declaring a Boolean, basically, which a Boolean is setting it to true or false. So once this condition is met or if it's not, regardless, the return value will be what this field is or what this variable is here. So we're defaulting it to false because we don't want to automatically assume that they can't push it.

Then we're getting the user's last name and first name from their user ID, and these are built-in functions inside Fusion. And so we access the user ID and declare it to this user variable here. We then concatenate a string of their last name, a comma, space, and their first name instead of to username.

And with these you have to be very careful because you have to make sure that they match the value that's in the pick list itself. So some of them I ran into problems where the space wasn't in the script, and so it wasn't working. So just little things like that you have to pay very close attention to make sure that you have it all written correctly so that it will work.

So we have the username now. It's the last name, first name and so this will be an if-then function to check to see if that item, which I'll go back here just to show you exactly what I'm referring to here. This CSR field is the field ID for this Customer Service Representative field. So if the CSR is the user that's logged in, which at this point it's me, this will enter.

This billing variable will now be changed to a true, and then it will return the true value. But if this wasn't my name in this field or if the field was blank or something, this would not be entered. This would not be changed true, so it would return false.

And in that case, this little green line would not be green. I would not be allowed to push that transition. So this is a great way to make sure that someone that has the permissions and should be pushing that workflow transition is actually pushing that workflow transition.

This is an example of validation script. Here, I try to push this, but I didn't have a Result by Date. And you can see here that the Result by Date is blank. So when I try to push this transition, I'm allowed to push it because the conditions are allowing me to push it, but the validation is not letting me push it because I don't have a Result by Date in the workspace.

So I'll show you the code. Real simple code. First we're starting with an array. An array is basically a variable that you can add values to. So I declare a message variable. I check to see if the Approved By, which is referring to the Date Resolved sorry. This should say-- just forget what this says, I guess. But this is referring to data that's-- or the field ID of that Date Resolved field there.

So since it's equal to nothing, which is null, then this would enter because if there's nothing entered in that field, basically, what it's saying, then whatever you put in here would be what would be passed to this message variable. And then we can see that date must be entered as Resolved by Date. So this is what I'm passing it. And basically, if this array has a value in it, the validation script won't allow the workflow transition to be pushed. And so then the return value on this one specifically would be the date must be entered in the Resolved by Date.

And it gets displayed here in the workflow right above the comments, right underneath where you push the workflow transition. So since that's displayed, it won't allow you to push that transition until this field, which is supposed to be Resolve by Date, sorry, which is if that's has a value in it, then this would allow that transition to be pushed and the message wouldn't be passed to it.

And so with this example now in our tenant we have workflow transitions that if we're doing something in one workflow and we want a related workspaces workflow to have a transition pushed, we would push this, and when we push that, then this one would push itself then. So I want to walk through this with you. So the first variable is related drafting, which this is just storing the related drafting workspace of that other workspace, and we want to push that transition in.

The next one we'll get the transition ID of the related drafting process. So we want to make sure that this is in the correct state before we try to push it to the next workflow state. Because if this wasn't here, we wouldn't have the access to the ID to push the next workflow transition.

So we want to make sure that's in the correct state. Then we take this related transition work variable. We check to see what the ID is. If it's 221, then this long line here would perform that workflow transition, and you would pass it the variable or the ID of the workflow transition that you want to push. And you can write in basically whatever you want that state to have recorded of what this was.

So in this case, Reviewed would be passed to it. So in the line, it would store the review value there. So basically, get the related work item, find which transition ID it's in, so basically, which transition it's at, push the transition, and store the value.

In our tenant, we have the Milestones tab set up this way. And the way we have it set up is as we go through the workflows, we can get certain dates and set them in the start and end dates here. And we have this basically-- is anybody using their milestones? Michael is. A few of you are.

So this is a great tool to keep jobs on track. People can go in these and check these to make sure things are happening. As you can see, two of these are red. That means that they are overdue. Something should have been done by now. The green, this one is at the final state in the workflow transition, so that one is done.

And the oranges are coming due, so this is just a good way for people to keep track of where the jobs are and where they need to be with certain points in the project.

SHANE Trent?

WEMHOFF:

TRENTON R. EARLEY: Yes.

SHANE WEMHOFF: I just want to explain prior to this our milestones. And before we had Trent set up some of the scripting, we were having our employees go in and manually set these dates. Well, you know what happens when you ask an employee to manually do something. It doesn't get done. So they weren't very accurate. So no one wanted to believe what was going on there because they weren't updating their stuff.

So when Trent come along, we had him come in and write the scripts to automatically set these, so that way we could be more accurate with what we were looking at here. So it's kind of give you a picture of what we were struggling with before we had Trent automate this for our company.

TRENTON R. EARLEY: So yeah, I'll go through the script of how we automatically set the dates there so that we don't have to rely on people to enter those in. I won't go through all this with you, but I just want to give you an idea of what's possible and how you accomplish it. So we start by calling, well, I won't get into it right now. If anybody has any questions after this, I can go through with you and go through each line of code and describe what's going on.

But basically, when this is called, we call this other function here, and all these values that are being created here are passed to this function, and this is basically all the dates that you just saw. That's how we're setting all those right there.

AUDIENCE: [INAUDIBLE].

TRENTON R. EARLEY: That's progression through. So it's like 30%, 50% then complete 100%. So if you have any questions at all, we'll get the answer bar after this, so I can go through it with you. But I just wanted to show you this and not really get into the thick of it, but show you what's possible and how you accomplish it.

So Jitterbit, this is how we integrate our data with our ERP system and also how we just expose all the data to create a couple of web pages to create custom reports. So I will just go through a quick demonstration of how you create a request inside of Jitterbit.

So this is the hierarchy tree on the right-hand side. This is all the connectors that are built into

Jitterbit to connect with our Fusion tenant. So as you can see, you create gets, ups, updates, upserts. Like I said, the upserts are specific to the Jitterbit. But it's just a really easy way for us to connect or expose our data, I guess, to either input it or get it from our ERP system and vice versa from our ERP system to insert data into our Fusion tenant.

So this is the first screen that you get into. When you're creating a connection, you'd basically just put your hostname and your log in, enter your credentials. And always test your connection. You don't want to be building this and get halfway through it and wonder why you're not exposing the data. Just always test your connections while you're going through the process.

And this is basically the-- I won't call it the Home screen, but this is where you start with your requests and responses. This will be what you're using here. This will be your workspace ideally you're connecting to, workspace name and ID. And so the buttons on the left or the right-hand side would be the request response, and the operation would be basically the chain that connects everything together.

So this first screen is showing the request, and specifically in this example, I'm going to be taking data from my Fusion tenant and exposing it and inserting it into a database. So with this one, we're going to source to none because the source is going to be our Fusion tenant, and the target will be the Fusion tenant itself as well. So basically just set it to None every time whereas you connect, if you're getting data from Fusion.

And the response structure would be as you can see, the source is our Fusion tenant that we had set up, that last step. The target is going to be a database on this example. You can name it whatever you want. It defaults to what it said here. It will default to whatever the name is of the tenant you're connecting to with either request or response. You can set that to whatever you like.

And then on the lower picture is our connecting to our database. So basically as we were connecting to our Fusion tenant assemblers, just enter in what kind of database you're connecting to. In this case, we're connecting to a MySQL database, enter your credentials to connect to that database, and again, just test your connection. Always test your connections as you're going through here. That's a big headache that we're running into little problems like that as we're setting up this initially.

And here's where the fun starts. So on the left-hand side, you can see in this example this is our Fusion tenant. And on the right-hand side, is our database we're connecting to. If you're lucky and you have these IDs set as the same IDs as in your database, then you can hit the AutoMap, and it will just actually map everything for you.

In all of our cases, that never works for us because it's always different names. So we always have to manually map them. But you just grab the value after you expand whichever field you're getting into. So this one would be our queue alert checkbox. So you'd take the value and just drag and drop it to the Queue Alert checkbox on the right-hand side. And you just basically do that all the way down the chain until you have all the data you want mapped into the database you're inserting data into, or upserting, update/inserting.

And then there's options to script inside your Jitterbit tenant, and there is quite a bit of options. I just expanded a few that we use quite a bit. You can work with these to send emails at certain points. A lot of times, we'll send them if the operation fails. We'll use those functions to send someone an email to let them know that they didn't go through. You've got to figure out what's going on.

Then the if-thens, we use those quite a bit. There's also the case and equals and while. So I just wanted to show you what is available, and I'm not going to get too much into how you do it.

AUDIENCE: You've got it mostly for validation?

TRENTON R. EARLEY: We use ours for validation. Yeah. You can actually manipulate the data as well. So yeah, good question. So I guess we have some time. We can go through one example. We've got plenty of time. I was talking too fast, I guess.

So a specific example, this if, and the way they have it set up inside Jitterbit, it's a little bit different than a normal if statement would be but, it's pretty easy to figure out. I don't have an example here.

So if the condition is met, so if x equals x, then after the first comma would be what you do if it's true, and after the second column would be what you would do if it's false. So you can manipulate data. You can validate data. You can do anything you want basically.

So I mean, it's pretty handy when you're doing transitions, especially if the database isn't set up to be the same data as what you have set up in you tenant, then you can manipulate the data a little bit as you're going through the mapping process to clean up the data a little bit.

AUDIENCE: Do you have like preformatting function and stuff, or like the [INAUDIBLE] string. Is that what you use you use that for?

TRENTON R. EARLEY: Yeah. I'm pretty sure that's--

AUDIENCE: [INAUDIBLE].

TRENTON R. EARLEY: Yeah. I mean, it's handy to clean the data up as you're passing data back and forth.

SHANE WEMHOFF: Trent?

TRENTON R. EARLEY: Yeah.

SHANE WEMHOFF: I'll elaborate on our Jitterbit integration. So Trent explained how we've integrated. So some of the examples of things that we've done with Jitterbit is we have multiple softwares that we use. Kind of paint a picture of what you can do with it is we have a software called MBS. It's our design software. I mentioned it earlier. We take and we have an order that we have built.

So you enter a whole bunch of data into this MBS. system. It's all of our building information, and we currently move it over in Excel spreadsheet. Our Excel spreadsheet becomes the order form. So what we've done is we've gone in, built a holding bin. We'd dump that data from MBS into there, use Jitterbit, and we'd peel it into Fusion.

We have a workspace in there called Frame Building Order. it's over 800 and I think 50 fields of information that we've Jitterbitted across from that. And things that we've done with that as we bring it across is we have feet dot and then it's fractions or it's inches. So you have 25.544 feet. We've taken that information as we bring it across, we make it 25 foot 4.25 inches so that way it reads like that on the order form before we bring it in.

So that's one example of what we've done with Jitterbit. Another example is our ERP system. JD Edwards. We have dates. We have dates inside Fusion. We have dates inside our ERP. They're important for our work orders, and so we've aligned those dates with each other. So we're taking actually, Fusion as this is where you put all your data. We want our ERP system to match Fusion.

So we don't want him going all over the place to try to enter stuff or put things in. We want them all to match. We want them to go to one place to look at all this information. So we brought this stuff over, and our dates match in Fusion here. Our work orders are right. When the load goes to pull, the pull dates all match. Our customer, when our CSR answers the phone, they can look in Fusion, and they know the dates are going to be the same that's in our ERP system.

The other thing that we've done is we have a database inside of our JD Edwards system of all kinds of addresses. Well, we don't want to build a whole other database inside of Fusion for all those addresses, thousands of addresses of all our customers. Everywhere we've ever shipped anything, we have all that data.

So we decided to go, instead of from Fusion to our ERP system, we said, hey, we want to take all our stuff in our ERP system, shove it into Fusion now. So we have forums that we've created that were Word documents, Excel spreadsheets that are out of date. So a lot of things that happen when you have Excel spreadsheet, Word documents, you try to keep everything up to date.

Someone's computer goes down. They bring in a new computer. Well, guess what? They've got a new version of Microsoft Word. This guy has an older version of Microsoft Word, so now your forms don't work like they used to because of the macros, we built those forms into Fusion. And the data that we were bringing in from our ERP system, we're using Jitterbit to bring our shipping addresses, our house order numbers, all that stuff in through that method.

And then the last thing that we're working on is we've built a lot of standalone programs. So we have piles of paper where someone stood around, manually was entering data off these stacks of paper into a program that you push a button and then it goes in and displays to all the screens inside of our ERP system. So you have your manufacturing screen, several screens, 8, 9 10 screens that all this data has been pushed on. So we built that.

And with Windows XP going away-- that's what all this stuff was built on, that type of platform-- we had to try to solve that. Well, we're going to use Jitterbit to take the information in Fusion and replicate that homegrown software that we've built. So that way we don't have to try to keep up with XP machines and Windows every time they change. Fusion adapts, Jitterbit adapts, so we don't have to sit there and try to keep up with all that.

And then the last thing that we're doing is we have a contract provision, same similar deal. We have a homegrown program. We're going to feed that, and it's going to communicate back and forth. We have data that needs to go this way and that way with that, and we built that.

SHANE WEMHOFF: One thing that Trent did-- sitting here talking, thinking about it, is we had a big magnet board, and we kept everything up to date with this magnet board. So everyone got their little markers out, drawn on the magnets, sticking their names up there. And any time that something needs to move on that board, you had to get up out of your desk and move that magnet.

Well, all of this data is inside Fusion. All this stuff that we're doing up here, all the customer names, all of our shipping addresses, all of our shipping dates are inside there. So why do we need that magnet board? You have to take the magnets off. You got to wash them, clean them, all that good stuff. A lot of time.

Trent went out, and he took all that data, dumped it into a database. I should probably let you explain this more because you created it. And then from that database, we created a green board inside of-- we called the magnet board the green board-- the green board the inside of--

TRENTON R. EARLEY: Internet.

SHANE WEMHOFF: --the internet connection. So we used Jitterbit take that data out Fusion, dump it into a MySQL server.

TRENTON R. EARLEY: Yeah, MySQL database, yeah.

SHANE WEMHOFF: And he created an internet page that now is live, and it's color coded. And so that way our team leaders can see where everything is at, or people working on our boards can see where everything is at. And they don't have to get up out of their desk, write their name on a little magnet. Our one lady that does our mailing, she always had to go clean all these magnets and wash them off. Don't have to do that anymore.

So just some examples of how this integration with Jitterbit and some of the scripting, how it's helped us be more efficient and take some of the manual work out of what we used to do.

TRENTON R. EARLEY: So that's basically it. I just want to go over these last few slides. They went on their own. So we have this discussion forum and the idea station. You can go to these websites to post to the idea station or get on and ask for help. Basically, when I was first teaching myself how to use the PLM software, I used this a lot. It's a really good resource.

The ideas station is great too because if you have an idea of something that would help improve Fusion, put your ideas in the idea station or whatever. And then you give people kudos if you think they have a great idea. You can comment on it. You can just let basically the AutoDesk team know that, hey, this is a great idea. I've used this too. Let's talk about this. So the idea station is a great way to try to get things going, get on Marten's, love to get them to change some stuff in there.

So this is a blog that it's informative. If you want to know what's going on inside Fusion, it's pretty useful. PLM TV, they have training videos and stuff on their PLM TV, the YouTube channel, so that's helpful too, especially if you're new to Fusion and you don't know what you're doing. It helps a lot.

And yep, the answer bar. Actually, Shane and I will go over here next, so if you guys have any questions, we'll be over there. We can talk about stuff I can elaborate on the code a little more if you need help with what I was showing you. And the slides will be available or they are available, so if you want to look at them again to just see what's going on with what we have going on, they're all available.

So that is all I have for you guys. Do you guys have any questions? We got plenty of time, I think. Yeah.

AUDIENCE: Just one question about the-- when you connect for a relay in the workspace or the others in the workspace. When you're pushing information [INAUDIBLE], it should be a one-to-one connection. Do you have [INAUDIBLE] scripted in the background to make sure that no one is connecting [INAUDIBLE] another workspace, and then stuff gets overwritten.

TRENTON R. EARLEY: See, that's the thing with JavaScript, you got to be careful with how you're programming it potentially do some damage because you can overwrite data that you don't want overwritten. If this action calls this and does this, it's going to do it, but that was do they have to be of like data types?

So he wouldn't be able to write a single-line text into a date field or if you're working with pick lists, which we use a lot for usernames and stuff like that, you wouldn't be able to insert data into a pick list if it's not in that pick list already.

AUDIENCE: Yeah, but if you have workload on one workspace and [INAUDIBLE] or whatever, maybe another workspace, and then you connect two autos to the same [INAUDIBLE] workspace. So if you still use pick list, but you have two different items pointing to the same item in a different workspace.

[? AUDIENCE: ?] There is a possibility to list. Trent is using the scripting to control his business needs to connect. And with from an administrative perspective in life cycle, there's a way to say this [INAUDIBLE] must be unique across the entire item, so if you only [INAUDIBLE]. And let the system do its own security constraint and let the [INAUDIBLE].

AUDIENCE: Yeah, basically that was a question. Do I have it scripted?

[? AUDIENCE: ?] Oh, no, no.

[INTERPOSING VOICES]

[? AUDIENCE: ?] There's a conditional valuation, unique, must match this one with the other from another [INAUDIBLE].

AUDIENCE: OK, fine.

TRENTON R. EARLEY: See, that's the good thing about Fusion too is a lot of this stuff is already built in for you to use, it's just JavaScripting, if you need to do something more specific, that's when we use JavaScripting. That's why I said keep it simple too because if you're overthinking it, or if you're trying to do too much, then maybe try to find a different option, and there's a lot of useful stuff already built in. So this is just an option to just have more versatility with what you're trying to do.

[? AUDIENCE: ?] Try the forum. If you have a question on that, try the forum. And [? usually the ?] response will say, hey, is there a better way of using scripting?

[INTERPOSING VOICES]

TRENTON R. EARLEY: Any other questions? Well, that's all we have. Thank you, guys.