Speaker 1

Today is session number two and we will continue to do our first steps on creature platform. As developers, we worked with our environment. We made our package recently and today we will continue with git with doing other preparations. I will explain how to operate with packages, how to integrate with git and we will work with objects and data. I will show you how to build data items, why we need them, how to transport solutions. So you will see how it works in general. I hope it will be clear for you. Okay, so in case of any questions, feel free to ask. I'll be really happy to answer your questions. As usual, video recording is being performed. I hope that. So you downloaded video files from our yesterday session. I hope you made your homework and prepared your environment.

Speaker 1

So now you are fully ready to continue. If you have any questions, feel free to ask Tushar any questions we have. Is there some issue during installation of the local environment? I'm getting this one error. Please type in the chat. It's related to some redis server. This is where so you posted the text. So it was failed to connect to the redis and so it's something like it was not possible to connect to radius server right now and it was multiplexer train board connection. So first of all, thank you for saying this. So first of all you need to look at your connection strings config to make sure that you have a correct host setting and db number. Also you must make sure that your regis is running. You can check it by running your windows services. I already checked.

Speaker 1

My port number is different like your. You should find your Regis service running. If it's running, it means it should be okay. You can also look at Regis logs. Maybe it will be more clear for you and which version of Regis are you running? You can. You can get this information. If you go to C program files Redis, you can run this special app called Redis Cli. Then you will see something like this. You can type info command. You will see a lot of technical information like version number, amount of ram used, a lot of system parameters, and also you will see set of used key spaces. So it's a database numbers used by your redis where I can find the version number in this information.

Speaker 1

C program files regis usually if you installed regis for Windows, you will find it here and then you can just run regis Cli exec file. It will be a command line utility for redis. So you can run different commands like select number of database db size. You can see size of the database that you selected. You can also use some dangerous commands like flashdb for clearing of your current database or even flash all for clearing of your all redis server storage. So sometimes it's necessary in rare cases where your redis already has some previous old date and you would like to start from scratch. So removing of all previously saved values. Normally creation manages redis by its own so you do not need any kind of assistance.

Speaker 1

And usually creation is caring about redis itself only in some rare cases you will need something like this, like flush all in order to reset your redis to clear everything and that's all. But it looks like you have a connectivity issue. So question about network name or question about port number or something like this. So is it possible to run your redis CLI and make connection to your redis? Yeah, I'm doing that. So when I started Radis Cli you can just type the comment flush all which will clear full regis and it will clear all the previously saved settings. And if you have different attempts to install creature, there is some slight chance that existing redis data may prevent you from normal start. So you can try to run flush all command.

Speaker 1

I would also type it in our chat so this may help you to normalize your redis operations. But the most common reason I think is about connection strings. So you should take care about the host and database number. You can try to choose another database number, maybe you have some other apps using the same number and you may try to run another one. I'm able to log in. Did it help to fix? Yeah, yeah. Okay great. Thank you Tushar for your question. I hope it was not so boring for others to hear it, especially if you if others will also experience same difficulties, it will be really helpful. So let's move on with today topics. If any other questions appears, feel free to ask at any moment of time.

Speaker 1

So it will give me a bit more entertainment because so once you have a question, I will be happy to answer. Today we'll continue with git, with package management, and also with working with objects in general and using data items. And possibly I will show you some advanced example with connecting database views and objects. So today we will mainly focus on general stuff like git and object and data management. First of all, let's remember where we finished last time. Last time we manually created new package. As I told you, the scenario for manual package creation will be necessary in examples where you do some classic UI customization. In general you can create packages and then you can use it for freedom UI apps, but it's not so common because when you create freedom UI app, system automatically creates a package for you.

Speaker 1

So creating package manually usually means that you are planning to do some classic UI stuff. Just to remind you, classic UI is a special old type of user interface which you can still find in many system sections like system users, like system settings, like web services. You will see some specifics and no possibility to shrink columns easily. And a bit later I will show you examples of Freedom UI and you will see how you can use such tools, how it will look like. So we have two different types of user interface. They have a totally different undercover frameworks inside and that's why it's so different and we make such difference between different types of creature pages. Okay, so dev classic will be intended to use for some classic UI stuff. I need some dummy example to be put into my package.

Speaker 1

Currently you can see that we can select all packages. As you remember we have 141. Now let me check number of packages at our d one database. Now we have 141. So one package was added because we added it manually. And if you select all packages you see contents of all your all system. You can use some search. You can use by search, by title or by code. You can use with different search condition and you can also make filtering by configurable item type that you can add to your package. In my case, if you select any package you will see its current contents, especially if you clear search bar. So you will see current contents. And when I select my package, by default it's empty.

Speaker 1

We can create some dummy stuff here just to demonstrate something that is not empty inside of our package. So let me show you what it will be. For example, we can make a test process, business process. It's a type of configurable item designed for automation of some business actions in our creation system. So now I need it only as a template done stuff just to make it to make my package not empty. When I design my items I must provide code and by default system expects me to provide some prefix for this code. This prefix can be also set at system settings. You can search for prefix for schemas and packages name. You can open it. You can see default prefix is USR.

Speaker 1

You can change this prefix at the beginning of your project or you can even remove it if you do not like. So supporting prefixes. In my case I do not need to disable it. So let it be as is. Prefix usually helps me to detect my

custom changes and differ them from base product stuff. So anything which has my prefix will be definitely made as part of my customization. So in general I consider prefix as useful thing. But in some projects developers decide not to support prefixes because they just tired of making them. So in my case, when I create a process, I must use this prefix the same as my system setting us our test process. We can name this stuff as test process. I need it only as a dummy stuff.

Speaker 1

So I do not really plan to make some business logic here. So my test process includes some start simple event and then terminate event. So this process, it's just doing nothing. I have no parameters, no special settings, I only have code and title. Then I can save this sample diagram. It will be saved into my package, as you can see here. So I can close it. Now my package is not empty. And you see this item added to this package. You may note this big star nearby the item name. And this star tells us that our change is performed and not fully properly saved. In case if we work in this creation. No, no external ID mode or it's also called file system development mode. It's kind of a long name for this. So file system mode.

Speaker 1

In the case if we work in this mode, this star is really important and helpful because it shows us that something was changed and not fully saved on disk. But if you will turn back to creature ide, this star will be almost useless because it will represent that your changes are not submitted to the SVN. And if you do not support SVN, so you will always have big set of such items with stars and it will not help you to detect some recent changes. So in our case, working with file system star means item was not fully saved on disk. Why we need to save it? Because creature shows us some changes and they must be submitted properly in order to make correct set of steps. Make correct development approach so we can submit everything to file system.

Speaker 1

Download packages to file system the reason why we have to do this is that some changes when we do them at configuration section are not initially and fully saved on disk. They are still created and saved in database first, because it was historically and originally the first storage place of creation items in database. And only several years later possibility to save items on disk was added. And that's why we have some kind of legacy and historical stuff. And when you develop some items, they are anyway created in database first. Okay, so we can run this action download packages to file system. It will include all our editable packages, all the contents, and save a set of files and folders on disk. When it finishes it will show us some report what have been done and you will see this report easily.

Speaker 1

And also this star nearby. The item name will disappear. Meanwhile you can see the type of the item. It was a business process. Now you can see this small report includes text that this USR process was modified on this so it was fully saved. Now you see the star disappeared. We can go to file system. Here is our app file system. Go to Tailsoft, the web Terrasoft configuration PKG. Then we will find our dev classic package. And now you will see that it's not empty. You see schemas. Now we have some special folder. We see some set of files. It's not so easy to read such files and luckily we will not need to modify them using external editors. This is special set of settings called metadata.

Speaker 1

Metadata is information about our configurable items stored in a special system inner format in a JSON string with a big number of different properties and values. And luckily we should not look here constantly. So just something that you need to know that it exists. So metadata represents in the structure of the item and for some types of configurable items it's not really designed to be edited out of creation. So only creature editor should be used for this. This metadata represents full structure of what we designed and we can now have this stuff saved on disk. The

reason why we saved it on disk is our attempt to organize a teamwork. So we want to make some changes then to make a common repository, save our changes there in order for our team members to make them possible to get our changes from common repository.

Speaker 1

And that's how we will exchange settings between different developer environments within one development team doing the same project. So now as you can see, we're going closer to file system and to understanding how we can exchange settings between developers. I need to tell you that creation has two different flows of changes, two different, let's say directions of how you can transport changes. First and the most necessary to know is delivery. So when you do some your solution or you have an existing solution and you want to save it and then import it somewhere else in order to test or to run in production, you can just export this package into file. You can export any package you see system will produce zip archive file. And this zip archive file can be found on disk here like this.

Speaker 1

And this zip archive file can be also uploaded on some other environments. Such uploading can be done with the settings. Then you go to application hub and then it's possible to add new application with install from file option. So that's how you can get it from file and load it. If you do it many times, you will install form file many times and package will overwrite previous package settings so you can do it multiple times if necessary. But this is delivery procedure which is designed to be used only for test and production purpose. But if your task is to share settings between developers, this export and import approach is not good because when you import such package at your another developer environment, you will see this package will be read only and also when it's imported it fully overwrites your contents.

Speaker 1

So it will be really hard to operate as a team where each team member can do his own, let's say contribution to the common repository. So such export and import is not designed for sharing settings between developers. We have to use a bit more sophisticated approach and that's why we turned our environment into a file system development mode. So the step one is to make sure everything from your changes is saved on disk and step two is to use any kind of version control systems to operate with set of your files and folders from your disk in order to deliver changes. In order to transport your changes from files and folders to some common repository and then other team members. Other developer environments can take such settings from common repository from other from common repository to another developer environments.

Speaker 1

So that's how it can be organized as a team. And now I will show you. I will use git as a version control system, the most popular for now. And in order to run git I need some kind of good git client. I prefer to use visual studio code as a git client. So it started visual studio code. Luckily it's free of charge so you can use it freely. Here is my visual studio code. Probably I need to close everything close folder so I need to make just from scratch and what I need to do.

Speaker 1

So first before you enable any folders, before you operate with your git, if you don't have git client, you should click on this button source control in order if you so if you don't have any git client installed, you will see only one button here to be called something like oh we have to update. Okay, maybe later one single button will be install git client. In this case you have to run your visual studio as administrator when you click on this button to automatically download and install official git client and after this you will see the same settings as I do. Also you probably will need to run a couple of commands related to your client username and email. But this is something very common.

Speaker 1

I think you can easily fix it if you face this difficulty with new terminal window and a bit of googling will tell you how to make full setup of your git client. In my case, this git client was already set so I already have git client normally working so I can now open folder and initialize repository in it. This is very important to understand how you should manage folders. So here we have a our application Terrasoft web app terrasoft configuration PkG folder and we have a lot of subordinate subfolders here. What are they? I will explain a bit later, but now you need to remember and understand that currently I have only one package, but very soon I will have more than one packages in my solution.

Speaker 1

So if I decide to make a repository root folder somewhere inside of my folder representing a package, then it will be really hard for me to add any other packages here. So I will need to keep several repositories for several packages which is also not so good. So possibly we can select upper level folder. This PkG folder will be my root working copy catalog working copy directory on my disk and then as many packages as I need I will be able to use inside of one single git repository and also we will have to take care to remove from the repository. So do not version a lot of other files here which are not so intended to be changed and they are not part of my solution. So I plan to use a PkG folder as a working copy root folder for my solution.

Speaker 1

I will do this with visual studio code and in Explorer tab I'll just click open folder. I need to find the corresponding folder from my disk. And now probably it's time to explain why we have so many files and folders inside of TKG. So when opened you will see a lot of subordinate. Yes, I trust the authors, but you will see a lot of subordinate files and folders here. So the answer of what it is quite complicated, especially as you are beginners. So probably you will need to watch this part of recording a couple of times to understand fully. First of all, you may notice that folder names here represent some packages names from your configuration. Let's go. Let's look here. Let's search calendar base. Can we find it here? Yes, here is calendar base. Okay, next one. Content builder.

Speaker 1

Content builder is also representing a page. Okay. CRT analytics dashboard okay, let's find it. Yes, you can see it, but total number of such folders is much less than 140. So it looks like such folders appear here as a result of presence. Some packages in configuration. But for some reasons not all the packages from our configuration form their own folders on disk. So the answer of what it is a bit complicated and it's called file contents. Let me explain what it is. So generally any package in creatio may include unlimited number of items with different types like objects. We will study them today. C sharp sources several site programmed code in C hash language a lot of JavaScript client modules representing page settings.

Speaker 1

For client side we have some data items, we will discuss them and they are used to keep some data records in your system, business processes and some other types of items. So in general any package may include unlimited number of different configurable items. That's okay. But also developers decided and architects of creature decided it would be nice to make possibility to add some additional file system stored functionality connected to the package and they decided to make so called file contents. So if you have a package, any package may have its own file contents and this file contents should be saved in PkG folder. Then in folder name inside of it which is the same as package name, and inside of it we have files folder and it may include some DLL's, it may include some additional files and bin folder which usually includes DLL's.

Speaker 1

And this file contents is usually having two types of files. It usually has some DLL's as a result of compilation and for

some cases it may include some JavaScript stuff, JavaScript components. So generally the answer is that in PKG folder we have file contents which is a legal valid part of some packages saved on disk and we should treat it as read only stuff. We should not remove or modify any part of base product packages file content. So we should just simply ignore all of this stuff. So now we are going one step closer. You already know that we are interested only in packages that we design, ignoring everything else that we can see in PKG folder. Okay, now it's clear. So let's go to our visual studio code which will work for us as a git client.

Speaker 1

Now we only opened some folder with big number of files. We are really interested only in some part of such files, but we are not interested in all the contents. So let's go to source control tab and now it's time for us to make a git init comment. So some of you already have some experience with git, you already know what it means. So we need to initialize a local repository to operate with git. This is a necessary so it's like mandatory step if you want to use git we need to initialize repository I prefer to use as much as possible of graphical user interface to operate with git because it's easier, it makes less chance of mistakes and much more visual. So we can see what's happening here.

Speaker 1

So I will run initializerepository which is the same as git init command and it will initialize my git local repository in PkG folder. So I just click this, it performs some steps and now we can see that at this source control tab our git client detects huge number of modifications. Of course most of them are at base product packages. So we do not plan to version such base product file content. We do not plan to submit it to version control system repository. So we need some tool to ignore this big part of PKG folder and we only need to keep and take care about packages that we decide to create ourselves. Now we have only one package that we are really interested in submitting. This package is called dev. Let me show you dev classic package.

Speaker 1

So this is our package that we plan to operate. We want to ignore everything else. Custom package is also editable, but we agreed that it's not a good idea to use custom package for big project development. So I prefer to ignore custom package as well and keep it empty or only with some temporary stuff which will be not submitted to version control and will be not shared with any other developers. So I have simply only one package here so far, but soon I will have more packages. I need to ignore everything else and from my previous experience I already have two for this. If you have no previous experience, you probably know that in git you can ignore some parts of your working copy folder by use of Gitignore tool. And I can find good example of Gitignore tool at my GitHub repository.

Speaker 1

So let's go to GitHub.com. I need to sign in. I use my Microsoft authenticator for this. Just a second. Authenticator helps me GitHub yeah. Oh, sorry, change number. Okay. Okay, this one. Okay. Successfully confirmed my person accessing GitHub. I recommend you also to register your account at GitHub. So what I need here is my repository. Confirm your account recovery. No no no, later. It's okay. So I have a lot of repositories at my account. Probably you can find it here. My repositories and what I need here, I need some example of a recent Gitignore file. Originally I had no examples, some of my colleagues proposed an example of Gitignore file originally made for visual studio, but now I have a better example so I will have less time to update it. So I already have some example of gitignore file.

Speaker 1

Of course I will share it with you. Gitignore file is a file with empty name and gitignore extension. And the main reason of this file is to provide set of macros or a set of text templates which will be used to find and exclude from the contents all stuff which we consider as not necessary. Usually it's designed to exclude log files, some settings files, some technical files generated by your integrated development environment, temporary files, project files, sometimes executables if you do not need them, debug files and so on. In my case, we will use this Gitignore file to keep set of base product packages names, to exclude our to exclude them from our versioned contents. So we'll

simply copy all of this code. I will make my own a new gitignore file inside of Pkg folder. The name of the file is Gitignore.

Speaker 1

If you will do this with your Windows Explorer, make sure you set for display of the full file names with extensions. In this case you will not fail and you will not create a gitignore txt file. So I will create new gitignore file, paste this stuff there and save it. Now you will see immediate change happening in my repository, in my working copy source control tab. Now you see gitignore file. Okay great. We can still. We can also edit it here if necessary. And we still have some packages appeared here. The reason is that my Gitignore example was from version 812. Currently we are running already new version. This version is eight one three, the latest available released version and in this 813 version some new changes and new file content appeared for base product packages.

Speaker 1

As you can see CrT omnichannel appear it here and I can add it to gitignore. Let me show you what happened and also I do not need to put all of the files here, so package name is enough and CrT security so I can type it manually CrT security and save it. Now you will see I cleared my versioned stuff. I only have my dev classic package here. I use the gitignore to remove big set of base product packages and base product file contents from versioned stuff. So Gitignore helps me to do with this. I have a small poll for you, could you please confirm do you have any experience with git and if yes please type plus in our chat or if you don't have any experience with git and you do it first time, then please type minus in our chat.

Speaker 1

So it will help me to understand your current level of skills and knowledge about version control. Because today developer I think must have some say basic experiences version control tools because it's quite inevitable to operate as a team. So thanks for all of you who replied you have some experiences git. Those who do not reply and please don't be shy understand that no one knows everything from the beginning, so everyone is studying every day. So some years before I also was not knowing about git at all. So this is something that you can learn and my task is to help you with it. So now we have settings for our git client to include only necessary parts of our code and we excluded everything else using gitignore.

Speaker 1

Also I wanted to show you in file system once we initialize git repository, you will see that new folder appeared on my disk folder name is git and this folder is hidden. As you can see the file attributes here show us it's hidden. So possibly I will not see this folder easily in my Windows Explorer, but here I already saw set show hidden items and also already set for showing file name extensions. Just not to be tricked by windows. So now you can see hidden folder dot git even in Windows Explorer. And dot Gitignore file has a dot gitignore extension, not text extension. So it is also important. Okay, so this is my local git repository. I did not perform any comments to it, so I did not submit any changes there. So it's time to make my first commitment. Let's do this.

Speaker 1

My first commit ever. So it's just something fun. Text and text for your commits is important to describe your changes in general. Like you did some new section, you did improvements integrations or you made some data fixes or anything else. So your commit text should help you to track history of your changes in future. And it also will help others in your team to understand what actually you are changing and what is the sense of your changes or why you submit some new changes. So when I do commit, it will run git commit action and it will save changes in my files into my local repository. My local repository now became a bit bigger, so it has a bit more files and folders now. You can track it if interested, but the inner stuff here is really hard to read.

Speaker 1

So it's about some git secrets, how it saves your comments, your changes and so on my git repository is still local so I did not share anything with my imagined team members. I need now I need remote global git repository that will be available for my team members. I will use GitHub.com as a central git repository host so it will be my git server and I can create new repository. I will have a lot of repositories here and I can easily create new one more new repository for our training session stuff. Oh have some followers. Nice. And it will be a public repository so anyone can take information from it. So you can also read this repository, you can use its contents, you can save it at your own and see what happens if you add your files to creation.

Speaker 1

So I need this common repository to organize sharing of settings with my colleagues. This is very easy. I will also I also avoid using commons. I user interface because it's easier and quite straightforward. I will create new repository to be something like guided dev May 2024 let it be so this name was not used inside of my account and no description public repository, no readme file. I'm too lazy for this. I already have gitignore so no need to add it once more and no license. So anyone can do anything with my code. Simply create repository. After doing this, it's time to think about some git commands to connect my remote repository with my local one. And we have set of commands that will help us to do this. So first comment is git remote add origin.

Speaker 1

So I will connect remote repository to my local git client and my local git repository. I can do it with a terminal window. Actually I can run any command line on my PC, but I prefer to do it here just to be a bit more consistent and to show you what happens directly not far from my visual studio code. So I just paste this code git remote add origin and as you can see my remote repository had this URL. Okay, let's do this. Then we have a set. Next comment, select a main branch on my local one. Git branch m main okay. And the most important command is git push which will transport changes from my local repository. So tools transport my commits from local repository to this global one so it will become publicly available. Git push it usually needs some Internet resource.

Speaker 1

You see, compressing was used here. And finally my remote repository was updated by set of commits by stuff that I committed from my local one. We can check it out. So now my repository is ready. You can see some files and folders here. If you go inside you will see that git keeps only files, only folders which are not empty. If we compare with our physical file system, you see for example data folder which is empty and it's not present here because git is still not supporting empty folders. But it's okay, so it's not a big trouble. If you really need empty folders, you can use some placeholder files. Creature is not sensitive for this. So even if we submit to version control, only meaningful part excluding empty folders, system will still be okay with it.

Speaker 1

And now we have some stuff from our package which is not empty. So we have some files and folders committed to local git repository and then to common repository. So now it's globally available and I can share my repository link for you. So you can also check it out if you have any git account and know any restrictions to access GitHub.com from your PC. So this is my repository and what the reason why I made a repository based on PkG folder. Now I have only one package and you can see this is a subordinate folder inside of my repository. If I have two more packages, it will be two more folders on this level. So having this kind of repositories helps you to keep minimum number of repositories and maximum number of packages inside.

Speaker 1

So I can easily add more packages to my solution and it will still be located at one repository. You can also see all

the history of my comments. You can see changes there. So we will work with it a bit later. Now I will demonstrate what usually developers do when they operate with git and how their, let's say set of steps should look like we are starting some steps at our development. For example, we open some configurable items, we do something very important. Okay, I'm not so lazy today. Let's do some read data. Let's say we will do some dummy thing. Read data from a contact object where name, full name for example equals to some value. No problem. Let's compare its value like supervisor. This is just something that is a demo setting, so I do not actually need it so much.

Speaker 1

But this is something that and I will read only one column 80 okay, so this is just example read contact. An example of what we can do as business logic automation inside of a process. Let's imagine we did something useful, something very important for us, and then we save our customization. This is an example. In real life, your package will include tens or even hundreds of items. You do some steps in your development. Finally you decide that some part of your solution is ready so you can save everything. Then you go to your configuration section and download all the packages to file system. The main reason is to get rid of this star and make sure that all of your recent changes are properly saved on disk and represented at file system level.

Speaker 1

Originally when you do customizations they are saved in database and only if you download on disk it will be properly and fully saved on disk. Some items are also saved on disk while you edit them. But this download action ensures that everything is properly saved. So you must do this action when finished. Some development steps. Okay, I have some changes now on disk star disappeared so everything is properly saved. Let's go to visual studio code and let's see what's changed here you see we have some changes in descriptor for our process. Something like date of modification changed. You can see changes in metadata. We have some new item added, we have some titles changed, some position changed. It's really hard for processes to detect changes like this, but this is something that we do. Just an example of some customization made.

Speaker 1

Also we have some new items appeared here probably because of our titles added to the process pieces. But okay so we have test process and so on. So we have set of file changes. We can commit it as a commit example one and we can do git commit into our local repository. Great. Now we have no changes. So we have some pending stuff that should be pushed to global repository. So we can again run this command git push or which is easier I think is to click on sync button which means pull and push. Pull and push. In my case it's like valid set of steps. So I prefer to do this pull and push and that's how system will automatically perform the same git push to my common git repository.

Speaker 1

Now we can go to git repository loaded and you see some history of changes. So we have two commits. Here is my commit history. You can, you can see and compare changes. Let me show you here you can compare changes in each file, in each separate file. What was changed there? I agree. Sometimes it's really hard to perform this type of comparison and it's because of the metadata stuff here. But anyway, technically you have tools to perform such comparison. You have tools to track changes and you can see what is changing, what is happening in your repository. You will see all users committing, you will see all history of commits. You will understand pretty much everything what happens with your package on file system level. And this storage now is a common repository. So let's imagine we have another developer.

Speaker 1

Another developer originally has no own package at all. Another developer just installed new environment and they have no package at all. So only one single developer should create a package, should commit it to file system and then to version control repository. Another developer should go to this common repository, another developer should

organize their local git repository in TKG folder and finally pull changes from this common repository. You can also make some clone, but I think it's something that probably will be a bit more tricky because you will have to do it in a proper place on your disk. So finally, task for second developer will be to operate with their own PKG folder, to make their own local git repository there and then to get this folder out from git to local file system.

Speaker 1

They can do it with git pull comments or git clone comments, whatever works better. And finally, creature is interested only in PKG folder and the contents here. So finally you will get this folder from git repository to some another developer environment. After you do this, you will have file system stuff like this. You probably will have empty folders for items that were not used so far, like SQL scripts and data will be empty for some time. Then second developer must tell creature that PKG folder file system has something new. Creature will not detect it automatically, so you need to push it. You need to like make a punch for your creation with the command update packages from file system.

Speaker 1

So second developer will perform this operation and second developer will see results that some new files, new folders appeared and even new package will appear at second developer environment. So that's how your package changes may travel from one environment to another. I mean from one developer environment to another developer environment. This is very important to mention we are talking about developer environments working with version control. Let's imagine another developer environment opens some changes or creates new items, finally saves on disk, then commits everything to file system, then performs git commit at GitHub client or sorry, at git client. Probably it's a visual studio code. And then perform Gil git pull and git push in order to update central repository after this repository was updated, first developer may also perform git pull and git push.

Speaker 1

And that's how first developer will receive all changes in the package files. And then first developer must run the separation update packages from file system. So that's how you can exchange settings between developer environments. And as you can see, it's fully 100% based only on files and folders. So no any special magic or special calls or special web services are involved into this. Only files and folders are enough to save changes to extend to exchange your solutions between developer environments. As you can see, it includes plenty of files. Let me show you, 30 files are here. In big projects you will have hundreds or even thousands of files, so it's much easier to miss something. But it offers you possibility of teamwork. You can change only some part of it.

Speaker 1

You will commit only changed items, changed parts to common repositories and everyone else can do the same. So that's how you can easily, let's say, modify two different items of the same package and work as a team. There are some warnings about this. Some items, like processes and also objects are not designed to be simultaneously edited by different developers at the same time. So you should avoid situations where you have two different metadata files and you should avoid situations when you will have to merge them into one. So here you will see the main metadata file, which is really hard to read and it's probably not so practical. You should not try to make any manual changes here, even in case if you got this situation.

Speaker 1

When two developers made changes to the same item, I think the best way is to sacrifice changes from one of the side and submit another side changes into repository and then ask first side to check out it and make changes again. So merge of metadata is not designed and probably will not be successful. So you should not try and it's better to avoid this. So you may communicate with your team members and organize your work so they will not interfere into each other settings. But in one package, as you already remember, you may have hundreds of items. So if developer one modifies item one, if developer two modifies item two, there will be no conflict. They will

successfully and happily work with the same package. So you should be just worried about simultaneous access with the same item for different users, different developers.

Speaker 1

If you have any questions about it, any other ideas, feel free to ask. I'll be happy to explain. So in general, now we made all major settings and you see how you can set up your common repository. Now you see how you can submit changes to this repository and we can move on with next topics probably before moving on, I offer you just five minutes break for coffee and then we will move on with working with objects topic and managing data. So we finished with the question of git integration and it's time for us to think about working with objects and working in general with creature data model. So what you need to know about it is that the basis of all creation data model is a configurable item called object.

Speaker 1

You can easily create objects in your configuration and also when you do customization using no code tools. For example when you operate with application hub or with section wizard, system is capable to create number of objects automatically. But in order to study it, you need to make some examples manually. You understand it better if you do it like by hands. And creature has a lot of out of the box existing objects, so maybe a couple of hundreds here. And a lot of them represent simple entities called lookups. Let's go to lookups and let's see what they are. So go to lookups. For example, we can sort. Okay, I prefer to look here. Select fields to display, switch to list view and it was originally entire view, so I plan to look at list view.

Speaker 1

And also I need created on column just to make more comfortable sort in this list so I can sort by name. For example, we can find a lookup called address types. So it's something simple, address types, you can open it and we will see set of records representing types of addresses in creature, like home address, business address, actual or lingo address, and so on. We have six records here. Such records are physically saved in the database and we can go to the database using our database management tool. Make new query, select everything from address type. You can see we have the same name of the database table working with it. So now you can see the same set of addresses stored in the database. And some additional options are also supported here. So what do you need to know?

Speaker 1

Is any object created in creation will finally turn into a table in database. So now we talk about general objects without special chemistry, without any tricks. So in general, if you create an object, it will have its own database representation as a table. One object means one table. Sometimes participants ask me, participants are asking me about is it possible to have one object for many tables at a time? So normally not. It's not designed to be so. If you need to operate many tables and to get some aggregated data from them, you may use database views and then you will use a specially designed object for it. But in general, when you create in one object and everything as usual system will create and manage one database table. We can find this database table by name and this is address type.

Speaker 1

Let me find this one. So you can see table structure, all the columns, you can see it's key field, you can see primary displayed text field and some others. We have a lot of base product lookups. You can see how many we have if you go to view in your lookup section, view summaries set up summaries, display number of records and now you see we have 120 lookups, an out of the box studio product. If you work with more complicated solutions like sales, service or marketing, you will have even more lookups. And also we can easily create our own lookups and we will need them for our future settings. Lookups are generally used at the contents of data when you have a lookup column and you need to select some values from it.

Speaker 1

So building more complex solutions usually starts from making your lookups and that's why we will do some. Now I plan to do some changes in my package and it will be necessary for my future solution in classic UI. So this will be just an example for you and I will show you how to create new object from scratch. And business sense of it will be some lookup for our future section that we will design. So let's imagine in future we plan to make a new section to keep some reality data, information about houses and apartments for sale and for rental. And obviously we need some lookups for them. For example, we will keep realty records and we need to know information about type with realty. So it will be lookup.

Speaker 1

We can call it real to type and I will show you how to create it from scratch. So select our package add object when you create new object, it means you are creating new entity in our system. New database table will be created. Working with replacing object will be covered a bit later. So now let's make new object. When creating new object, we can provide some properties for it. First of all we should take care about its code. As you can see, system tries to avoid duplicates and that's why it generates some unique name for our object. Of course it's not good name. We would like to make its name by our own. We still have to use prefix because we did not reject to do or to work with prefixes.

Speaker 1

And we will have this prefix USR and then we can provide code for our new object. It will be real t type. And also I plan to make two different solutions, one for classic UI, another for freedom UI. So for classic UI I will have a suffix classic. So let it be reality type classic. I do not want to share any lookups between two different solutions, so that's why I will make it fully separated and that's why all the lookups for classic part will be fully separated from lookups from freedom UI part. So this is our code. Code is very important. It must be made only of latin characters and underscore. And sometimes you may also use digits but I recommend you to use only latin characters and it must start from character. And code will be used to create database table.

Speaker 1

So system is capable to make set of theories to the database like create table, alter table and add columns and so on. So we only provide some necessary properties and system will take care about database storage itself. This is also very important so code will turn into database table title it could be anything and even more it can be translated into other languages. Currently in my environment there are two active languages so I can easily set values here. But in total we have probably 21 supported language. Let me quickly check. This is very interesting. CIS culture is a special object that keeps information about languages supported for translation into our system and it looks like we have now 23 languages. Sorry, let's see the most recently added ones. So wow.

Speaker 1

It's interesting because previously I only remember added Japanese at the beginning of 2024 and now we have some very recently added languages. Hungarian language was added recently. So greetings to Hungary and croatian language looks like Hr, it's probably Croatian Hr. Hr yes, it's croatian language. So you see more and more, let's say national languages are added here and now we have 23 supported languages in our system. Some of them may be disabled, others are active and if you want to use a certain language, you should activate it. The reason of activation and deactivation is saving system resources. If you do not need all 23 languages, it's better to keep most all of them as inactive. System will save system resources for not necessary languages. So that's why we have active option and you can manage active or inactive language in settings.

Speaker 1

Then you can find languages part and now you can see 23 languages and you can activate language if necessary by opening it, setting active and then save. But please. So this is a warning for you. If you activate the language, it will consume system resources and usually activation takes some time, maybe up to five minutes. So please be very careful with it. Now you see recently added croatian language and hungarian language. So this is something that is interesting and it's new for this version. Okay, let's go back. So if support of multiple languages is important for you, I recommend you to enable necessary languages and each time you develop some items, instantly provide corresponding translations for your items. In this case, it will be hard to forget and hard to miss something that must be translated into other languages.

Speaker 1

We have also an alternative way to support multiple languages with the translation section, but I do not recommend you to work with it because when you initially go to translation section, it will take probably 15 minutes of time for initial preparation, so it's not so comfortable. And I think it's much easier to provide correct translations just while you develop it. So in my case it will be realty type classic UI. It will be my title if I want. I can support translations into other languages if necessary for my project. So code and title, they are important. Parentheses object is not mandatory, but it's also very important, especially if you make lookups. Inheritance helps us to get standard structure for objects, especially if we need some typical use of our object in our future.

Speaker 1

We plan to use reality type classic as a lookup so we can inherit structure for our object from some base objects. That can be helpful. In our case, the most commonly used Bayes object is base entity. It helps to inherit key field, historical columns and some behavior setting, some event handling that is necessary to operate in some lists and some cases. So base entities like common parent item for all creation objects, but also we have base lookup. Base lookup is used specially when you need to create new lookups, and that's why such two most commonly used items are displayed at the top and all others are shown in alphabetic order. So base entity and base lookup, the most requested, the most frequently used, are shown at the top. Okay, I need base lookup for this.

Speaker 1

I have to confirm my change, and once I do this now you see I have new inherited columns list. I have eight inherited columns. The most interesting is probably id as a key field. Its data type is unique identifier. You already saw examples of such unique identifiers in many creation tables. So such a hexadecimal big bar, big line of code is example of a grid globally unique identifier. Physically it's saved as a 16 bytes integer. So such data needs 16 bytes in memory to be saved. But it's very common for creature. So all creature objects are based on unique identifier key fields usually named as id. Also we have name as a text as it used as a primary displayed column name is mandatory. You can see required. Yes, the data type is text 250, which is very typical.

Speaker 1

And you should know that creature uses Unicode strings anywhere. So every text column you will find in creation is based on Unicode text. It means you can simply save any national characters there. Any kind of arabic letters, chinese letters, or even emoji can be saved into such text fields. So Unicode is supported. Also, we have so called historical columns created on the date of creation, which is date time date of creation of the record and created by. This is a reference to a contact. So this is a lookup column pointing to a contact of user who created the record modified on or performed last. So the time when last modification of the record was performed and modified by is column representing who what user contact performed modification.

Speaker 1

So we have description name and description are business columns, all others are system columns and process listeners is an integer column for a very rare cases when record level event handling is used in business processes. So what do you need to know? Inheriting from base lookup gets access and we get structure. So we inherit structure of system columns. The most important columns are id as a key column and name is primary displayed column and they are important to work with lookups. Also you can find in main object that some inherited properties are shown here id property which means key field. So this is not perfect title of the property. It should be named key field and id is used as a key field and displayed value is also not a perfect title. Previously it was called primary displayed value.

Speaker 1

So the main text column which is shown in drop down list of lookups. So name will be used for this purpose but you can change if you want nor any other special properties. We have some event handling made for this and inherited from parent one. We will study event handling a bit later. So what you need to know now when you do inheritance from Bayes lookup you automatically receive legacy of all columns that are intended to be used inside of lookups and usually it's enough. So usually you can create just code title inherit from base lookup and then save and publish and your object is ready. In my case I want to present and show you some additional features that could be interesting and also examples of how you can make your own custom columns in creation.

Speaker 1

So we'll go to columns list and obviously we can add our own columns to the object. We have plenty of options for different database types, a lot of text options. The minimum supported is text 50 characters, then we have 200 5500. We have unlimited length. We have rich text which supports fonts, colors and styles like italic, underscore and so on. We have special texts for phone numbers, for web links, emails and URL's and we have encrypted strings to keep passwords safely which will be not displayed as a text, to be always displayed as a set of stars and very rarely used CRC type which physically is an integer and it's used for checks. So it's like checksum column. I never faced a working example how we can actually use it.

Speaker 1

So just something like bizarre thing that possibly you will never use in your practice. So number is obviously integer and several types of decimals. Currency is also decimal with two digits after decimal point. The difference from standard decimal is that standard decimal is 18 digits and two after decimal point. Currency is 20 digits and two after decimal point. So 20 in total, two after decimal point. That's why we have so different. But you see we can choose any necessary precision for your solutions. Okay, date and time quite of those we have date time value separately, date separately, time some other types like boolean like unique identifiers like image link or color selection column file can be saved and lookups. Lookups are mainly so they are generally used and based on unique identifiers. But lookups also keep referential integrity.

Speaker 1

So when you have a lookup column system creates foreign key for you. So it keeps referential integrity which will not allow you to save into lookup column any value which is not faced in your lookup table. So we can try to make some examples like make a new text column named code title can be code text is 250. And as example I can show you how default values can work for us, especially how we can make autonumbering. So we can search for default value and we can set autonumber option. We can say prefix like cd which stands for code and number of digits would be two. First one will be cd zero one, cd zero two and up to CD 99 and then we will have the next number. So x CD 100 and so on.

Speaker 1

This is auto numbering which works pretty well and it doesn't require any programming. So this is a no code tool to make automatic numbering in your lines when you, when user import adds data. That's how we can do this and if

necessary can use custom indexing. You can also make your own custom indexes and even complex indexes. Unique indexes can be added, but generally creature takes care about indexing by its own quite well. It automatically creates indexes for lookup columns. So you probably will not face any significant performance issues in objects and data tables made by creation. So once we finished setup of our object, we have to save it. Save means saving only metadata. So inner data format to keep information about our configurable item.

Speaker 1

This type of item is also inherited from schema, but the word scheme I think it's hard to understand. So let's say configurable item which will be closer to its meaning and I think it's easier for you. So when we saved this item, it will be shown in our package contents. Now you can see database update required so object can be applied to the database but it's not performed each time you save, only when you click publish. Publish means apply changes to current application server and current database. In our case, apply changes means create new table with corresponding set of columns and provide necessary column settings and everything should be well done at the database to make our object fully operational. If we do not publish our object, we will not be able to use this object in our system.

Speaker 1

So publish is necessary to make our object operational and useful for us. Let's do this. Some years ago publish took a lot of time. It may took up to 1 minute of time. Now this operation is well automated and now we have no serious reasons to separate save and publish operation. Also when you click publish, save is performed automatically. So you may just use publish button. Forget about Save button and it will work for you because publish now works really fast. Okay, so when we finished with it we can close our page. By the way, this page is called object editor or object designer and now we have our object status is okay. So you see it's applied to the database level. We still have a sys which means not everything was saved on disk.

Speaker 1

But at this moment our object is already working so it can be used for us and we can try to use it in our environment. Let me show you how we can use it. We can go to studio workplace find lookups section. As you can see we have 120 existing lookups. One of the recent one. You can see something strange with no name here, but if we sort by date of creation. So this is list of registered lookups and we can see what recent registerings were done like white list of pages, opening restrictions, access to tag guarantees, or something like base product functionality recently added here. So we can register our object that we created as lookup in order to make it easily managed. And we can look at it and we can even fill it with some data.

Speaker 1

So when we click new lookup, it doesn't create new table. New lookup means register existing object as lookup. Our object is called realty type. We can name customers, we can provide custom title of this lookup. So name is separate from object as you can see, and we can save it. So this operation is just for registering object as lookup. Once we did it, we can click open content and that's how we can manage our lookup contents. The easiest way to manage object data is to register any object as a lookup. Then you will be able to see its contents. So we can create new record here and we can manually type some data like first one realty type which we work with. It will be probably house. So it's mean private house. We can save it.

Speaker 1

This is so called editable list and we are not using any separate edit page for it. So we just manage data directly in this list. Another one will be apartment. As you can see, numbering works for us automatically. The no code tool. Thanks developers for this. You can also use description if you want, but I'm not really needed. So we can also add more reality types. What else? It could be office space, it could be parking lot. And something very unusual like a castle. It's just an example. And I also wanted to show you some additional behavior here with the numbering. So

let's imagine you try to create new one, but then no data or something else. So you think that, oh, you don't want to create it. You see the code was generated, but then you just cancel.

Speaker 1

Okay, if you create one more, you see the number goes on, you cancel. If you create one more, you see numbering is going on and it will be not rolled back. The reason of why it performs. So when it generates system does not know are you planning to save it or not. So it generates new number, keeps currently created number somewhere in the database, in so called sequence in the database structure. And then we have no option to roll it back. Because while we are inputting such data, someone else also may create new records. So if system is expected to be used by thousands of users, we have no option to easily roll back such numbers. But it's not a big problem. Even if you have some gaps and missing numbers, it's not a problem with it.

Speaker 1

So you see that numbering is generated using database tools. This is extremely reliable and physically sequence database object is used for it. Bas is telling us and about appointment and we watch the video. Thank you Bas. See you tomorrow. Thank you for your message. Any other questions please? You're welcome at any time. So now you understand, we input some data here. This data is physically saved in our realm database. Let me show it for you example. Select id name from USR realty type classic. This is the name of my object. I expect to see the name the same name of my data table. Now you can see the same five records with some ids are created. I can also search for like created on and also USR code. So here are my columns. You can see you can select all of the data.

Speaker 1

So this is physical data saved in my database. It's pretty simple. And of course, when you develop your big project, you will not pay so much attention to every single lookup. You will do it in much more automated way. But this is an example which will let you understand and will make you better feeling of what object is and how it works. That's why we discuss it in so technical details. If you need to change something in your object, you can open it for editing. You can find corresponding property and change it easily. If necessary, you can customize your columns. You may rename some columns. You can add new columns. For existing columns you can slightly change their properties. It will be not possible to change the column code, but you can change column title if you want.

Speaker 1

So for inherited columns, there are some limitations on what is available for you for change. So generally speaking you can manage your object, you can change them, then you can save and publish. That's how it will be applied in big scale solutions. You will create tens of different lookups and creation is capable to help you with it with automation of lookup creation, using application hub tools or using section wizard for classic UI. So it will be not so boring each time when you create new object. This is technical right now just because we need some practice, we need to look at object structure. So okay, we have our object ready. We have some data here, so how we can share it with our team members or how we can deliver it to test environment.

Speaker 1

Of course we need to download packages to file system if we want to submit everything to repository. But here is very frequent mistake that a lot of beginners usually do. This will be about data part, so saving anything to disk here is okay. And in our package we only have test dummy process and we have our object. If we submit these contents to version control repository, it will mean that our package only includes structure. So it will be including this object and some tests process. If other developers install our changes, they will not see contents of lookups and they will not see registered object as a lookup. So this is a very typical mistake of beginners.

Speaker 1

They see their environment, they fill in with data, they think that everything looks okay and then they just export their package install for example in test environment and then they realize they miss lookup contents and they also miss lookup registering data. So we have technically only two options how we can transport such data. The most recommended option is item called Data item is a special part of the package and second option which is also supported but it's harder to use is SQL script. SQL script can be used for any kind of data changes or any kind of SQL operations that you want to run. When your package is installed on some target environment is much harder to manage and in general it's much more powerful and much more dangerous.

Speaker 1

So you should only use SQL script if you have enough SQL experience and I always recommend you to check your code with someone else who also have experience in SQL just to make sure you will not do anything dangerous in your database. SQL script can be easily used for permanent serious damage for your system, so your creation will not work anymore if you do intentionally some delete of system data. So two options to transport data items or SQL scripts. I recommend you to use data items as much as possible and when you create data item this tool actually is a snapshot of some data records taken from your source developer environment. Obviously we need to specify what source object will be used for it.

Speaker 1

In my case, I should select for my object code which I recently used to register my lookup in lookup section. Unfortunately, you have to remember this code name by heart and this object is called lookup. This lookup object includes a list of registered objects in our lookup section and in general we have 121 records here. 121 records and each single record in the lookups section represents registering of the object as lookup. We need it because probably we would like to be able to go to this lookup section to find corresponding object to open its contents on test environment just to check out that we have everything correctly here. So this registering was done at developer environment and it makes sense to transport it as part of our solution to test and production. Such data is saved in a system object named lookup.

Speaker 1

That's why we go and search for the object named lookup. We can find it by name, by code and we can also provide some unique name here. For example, we can add something useful, reality type, classic, something like this. I do it just to make it unique and good looking and self explanatory. So it will be easy to understand why we created this data item. As you can see, system offered a set of columns, but we can also add a couple of more columns here. I'm interested in date of creation created on column here in the list of saved columns, data item is a snapshot of data, so we will take this data from our source database and save it as part of our package.

Speaker 1

Bound data tabs represent exact records which will be saved into our data item so we can manually add such records. In total we have 121 records but most of them are base product contents so we do not need to transport base product stuff. We expect it to be present at our target place target database. So we do not need all of them. We can use our created on date column to sort and easily find without user filters. You can also use filters, but I prefer to make it as easy as possible. So we can sort by descending order by date of creation all registered lookups and that's how we can easily find our reality types single record this is registering of the reality type object as a lookup.

Speaker 1

We can only select this data select button and that's how we saved only one data row in this data object. Then we just click save and our snapshot of data which includes only one data row will be saved into our package. Now you

see data item is created. This is only registering of an object that's not the contents. So for the contents of the object when we open it, we have five records here. This is our contents. We need to make one more data item. This is quite easy. Add again data and we select an object. We have to specify object code. Here are things developers because code is unique USR realty type classic, you can easily find it.

Speaker 1

I prefer to use codes because they are always unique and system keeps this uniqueness and titles sometimes are a bit ambitious, especially if you use non english language. So you may struggle with finding some corresponding title and I prefer to use codes. Okay, we can provide some common suffix just to make sure what we are doing here. If necessary we can add more columns, but the most important of course is bound data. And we can select all the records. All the records we have from source environment will be saved into our package as a special part, as a data item and we save it. And now our data item becomes part of our package. Now you see stars here which means our items are not fully saved. So we have to perform download packages to file system action in order to save our stuff.

Speaker 1

No need to use camera styler because it will be recorded. I will stop it. Okay, so changes were saved on disk, stars disappeared. We can find all the items in our package here, including data part. Now you see some data records are saved as a JSON data values and you can see translations into all our supported 23 languages here. So data items also include translations and it fully says on disk everything is correct. And now we have object and couple of data items representing registering of object as lookup and representing contents our object.

Speaker 1

So now our solution is full and if we export this package as Zip archive then if we import it somewhere on test environment to work and user will be able to find our new lookup in lookup section, user will be able to find its contents and user will see such data. Maybe it's time to practice with git and let's make one more commitment. Now you can see a lot of changes, 80 different changes. Mainly this is our data parts. Also we have new object here and physically it means big set of files and folders used inside of package folder. So we can name something like realty type classic lookup added. This is the sense of our change. We perform git commit to our local repository. Then we do git pull and push to submit our changes to the common repository.

Speaker 1

Great, everything is done now common repository also felt such changes. You see three commits and you can see comment history if interesting. Now you can see all the changes here. So it will be easy to check previous values, current values and so on. So what I need to tell you about data items. In addition, what you know now, data items can be used not only for inserts, they are also used for updates. Let me explain what I'm talking about. So originally we have five records for our data and let's imagine we already exported our change. We imported it somewhere to test and production and everything is okay. So our solution was successfully delivered to test and production and our users are happy. But sometimes later probably analysts or business owners decided to do some changes.

Speaker 1

For example, they decided to rename office space into just office. And we can do this only at developer environment because we need to share all such changes to all other environments including test and production. So such changes should be started from developer environment first and then someone decided to rename parking lot into just parking. So now we have some changes made at developer environment and we need to populate them to all other systems. Data items can help us with it, but you need to know how to properly set it up. Let's open this data item for the contents of our lookup. You can also sort by date of modification to see the most recently updated items first. So we can see this one, the most recent. Let's open it now I will tell you a bit more about how system operates with data.

Speaker 1

So let's imagine such data items are so such data items is being loaded at target environment. System is capable to prevent duplicates and it uses key columns to detect presence of the record which is being processed right now. As you can see we have key for id. So id column is set as key and when system will load within the procedure of loading your package, it will take first row, it will take this id and then it will check do we have such data record at our target database or not? If not, yeah, it's okay, it will just insert all the columns that we have for this data record and it's not a big deal, not a problem.

Speaker 1

But if record exists system is capable to update existing record and here you will face something which is a bit tricky and you may fail for the first time. And you will have to look carefully at the settings of your columns because system will perform update only for columns marked with the first update option and by default as you can see no updates are asked here. So when record will be found system will just skip it because no columns should be updated. So if you changed names right here and you know that such data was already traveled somewhere and you need to update it. So you go to your correspond data item, you make sure you set forced update option for columns that you need to update.

Speaker 1

Also you may check some other columns if you're not sure and you also want to update so you know the source of truth is your developer environment. So if something else was changed at your target system then you have to update it anyway. So first update is not set by default. If you expect to perform update operation you should take care about it and once you do this your update will work. Also I need to tell you that in our real database now we have office and parking and if you carefully look here you will see that we still have old values here. This is a proof of this concept, what I told you previously. That data item is a snapshot of your database contents taken at a certain moment of time. So when this data item was created we had older values here.

Speaker 1

That's why we can see it here. If you want to actualize it, to make it actual and read it from your database, you can just click actualize data. When you do, this system will refresh data item contents and now you see office and parking exactly as in our database. Now let's check column settings correct. And now we can save it. So if you create your data item like this and if you took care about the first update option, then you changed some source data records. In your source database you should actualize data, make sure your data records are fully properly updated and then just save it in your package. Of course your date and time of the item will be changed.

Speaker 1

And next time when you export your package and import in some target environment because of change the date of modification, this item will be processed and corresponding updates will be performed. The same happens with our systems shared with a git. So we need to save everything in file system. Then we will commit our changes to our common repository and it will be an example of how we can do some updates in data and populate it on other environments. Now you can see we have only small part of changes here. So we have only some data changes but they are present. So it will be lookup data updates. Or maybe it's better to say realty type data updates. Okay, so we can commit it, pull and push and that's how our changes will be correctly saved. And it's about updates.

Speaker 1

You may ask me about the delete scenario, what happens if in our source database we finally realize that castle will be never used and should be deleted from developer environment and for test and production. Unfortunately data

items cannot help us to delete such data. So if you really need to remove your data from target environments, you have the only way for this. This is SQL script. So as example I will show you how we can write such script. We can use search by name or by id and we can create this SQL script in our package. Add SQL script, you should be really extremely careful with it. We'll provide its name like Usr Del Castle realty type.

Speaker 1

Of course in real life you probably never use this, but this is some like a training example for you so you can create SQL scripts for some specific needs. Installation type determines the way the moment of time when your script will be executed in the scope of package loading procedure. Before package is at the very beginning. After package means after new tables will be created and processed. After schema data, it's after package data was loaded and uninstall app is the very rare case when you will need to run some scripts when your application is removed from the system. So whether it be at the very end after schema data. We're not using backward compatibility because there are so strict conditions. So it makes your SQL scripts almost useless and backward compatible with just an informational property. So we can just keep it off.

Speaker 1

So I will write some script like delete from USR realty type classic. This is our name of the object and then where name equals something like this. So this is an example of a script which intentionally removes some data and this script will remove nothing if no data found. This is easy example. I do not use id and I'm using this free name just to make it simple and easy to understand. Such scripts are dangerous so you should be very careful when doing such things. You can validate it, you can check okay, no problems. You can save such script and also which is interesting. You can even run such scripts on your environment. As you can see status needs to be installed in database means that this script was not executed at this database.

Speaker 1

So if we run it now, you can go to actions in write part here, then install, which means run SQL script, you can do it carefully and if script finishes with an error it still will have status and also you will see its properties and last error message text will show you last error which happened with running the script. In my case it was successful, so my script executed with no errors. And now we can check our lookup columns, lookup contents. Now you see Castle disappeared, so it was physically removed from our source database. And also when this script will go to our package and will be executed on some target environments, also the script will remove necessary records from our target environments.

Speaker 1

You should be careful because if someone already started to use such data that you plan to remove, you will have some cascade problems with it. So you will have referential integrity exceptions. You will need to look at connected data and ideally you should think of it in your script, you should process such scenarios and you should think of what to do with connected data entities. And also the script is not saved on disk, so if I will click download packages to the file system, I will have everything saved on disk. Our session comes to an end today and it looks like we have no enough time to work with a complex example of virtual objects. And I will show it a bit later, but today I plan to make couple of more preparational steps for us.

Speaker 1

So now we saved everything on disk and later we will start with the building of our new applications and we will need some base product parts which are very common to be used. I'm talking about contacts and accounts and also need some additional tools. So my plan is at the end of today's session to show you how you can add some necessary frequently used marketplace add ons that will really help us to operate in future. Here my example with the lookup is almost done, and also when we removed some data from the lookup, you see our snapshot still keeps this castle. So this castle is no longer present in our database and it makes sense to actualize data now. You see, I

removed it even though if I do not remove it, my item was not changed, so it's not correct.

Speaker 1

I think that if I keep some data here which is not present in real database, this is mistake, so I should avoid such mistakes. I should actualize data, not to keep any non existent records in my contents of the package. So actualized data saved it. Now everything is consistent. So my castle fully disappeared from my conscience here and we can also save it again to disk. So my plan for today's session and is to show you how to install some necessary marketplace add ons into your environment. We will need it later so I have opportunity to show you. Let's go to settings, then go to application hub. This is a special part of managing our system with so called applications. I will discuss and explain what application is tomorrow.

Speaker 1

Currently we will use this application hub only for inserts so only to add some useful items into our environment. So add new application marketplace powered so we can find a lot of application there. And also we have some base product tools made by creation. Now I need customer 360 because it enables contacts and accounts section in my system. So simply speaking, when you install customer 360 you will have new sections in your system. You will have accounts and contacts designed in freedom UI. This is very useful, very commonly used. Almost any project implementation on creature uses context and accounts, so we will definitely need it out of the box stuff of our studio system does not include this. I will explain why a bit later when we talk about composable apps. So let's install customer 360.

Speaker 1

There is a high degree of integration and seamless connection between my local website and marketplace dot creation.com. So I just click install. You see some automation works. It asks me to confirm. Okay, no worries, confirm one more confirmation about app that I plan to install anyway. Confirm. Let it be. So it will take just several seconds, possibly up to 30 seconds to install this app and as a result expect to see accounts and contacts sections added to my system. They will be displayed in Freedom UI. So you will see the first Freedom UI page at our sessions today and main reason why I do this is I need contacts and accounts in my future. Examples when you added something to your app it makes sense to download installation logs. This includes a lot of technical information, but sometimes it could be really helpful.

Speaker 1

And this installation logs can be kept for some time on your disk just for future reference. Because even in case of successful installation, sometimes you may face situations where you see something that you do not expect and working with such logs can be efficient to check it out and to see some technical details about your installation. At least we can see how much time it took. So 1437 till 1510 it looks like 33 seconds. So it took only 33 seconds to install this app. And all the technical steps that were done here are listed. So if interested you can see and find no warnings, no errors. You can see even some SQL scripts were installed. And also I just forgot to say that SQL script is database dependent. So when I created my own script.

Speaker 1

It was only for let me show you properties here. It only for Microsoft SQL type of the database if you need a similar script for postgres, you have to create another SQL script. That's why base creature product if you will filter by SQL script only. It always includes three types of SQL scripts for each business action that we need. For example, let me scroll down and find any kind of procedures like create omniscient tables for postgres, separately for Oracle, separately for Microsoft SQL. Three different SQL syntaxes are supported. And that's why if your solution is intended to be supporting several different database engines, you have to make several scripts for it. But mainly when you develop your projects, you will know for sure target database type. So you only you can focus only on corresponding focus corresponding target database type.

Speaker 1

It could be most frequently it could be Microsoft SQL or postgres. If you expect to deploy your solution in creature cloud, it's highly likely that you will have PostgreSQL in the cloud condition. So it makes sense to use the same type of the database for your own local development. But it's not mandatory. If you develop on Microsoft SQL and then deploy your solution on postgres, it will also work if you support all necessary database dependent settings like necessary scripts. Also for postgres, not only for Microsoft SQL. Main part of configuration is not database specific. So when you develop an object, it will normally work in any type of supported databases. Okay, it was just a distraction. So let's go back to our logs. Now you see it took only 33 seconds to install. It's interesting to see results.

Speaker 1

Probably we will need to reload main page just to suppress any kind of browser cache issues that possibly can appear. So now we can see a bit more workplaces. Originally we had only one studio. Now we have all apps workplace studio and we have new one which is called customer 360. We have context Freedom UI section here you can easily check if the section is Freedom UI or not because only Freedom UI section offer you, they offers you. So to shrink list columns easily in classic UI you cannot do this. So let's go for example into lookup section. You can't do this. You only can shrink columns by going to set up summaries, no, to set up columns that field to display. And that's how you can do such shrink manually. In freedom UI you can do it easily.

Speaker 1

So that's how you can easily detect which type of UI you are working with. But also you can remember position of buttons. In freedom UI, managing buttons are usually the right part and fonts are different. The style of the list is a bit different functionality of buttons in list and you can see you can hide and freeze buttons, freeze columns in lists. So freedom UI is much more rich with UI capabilities and it's considered as a future of creature. And classic UI is now considered as a legacy which has a lot of implementations. So probably thousands or even tens of thousands different solutions are currently running all over the world in classic UI. And so creature supports both types of Ui so far. This is example of a classic UI, sorry, freedom Ui list.

Speaker 1

You can open edit page, you will see page reach of controls with oh no, no. This is a classic UI stuff. So this is still example of it's not a glitch, it's kind of behavior connected with a cache and server side. So we have to log out. We will need to log in. Trying to use my name. What's happened with my saved users? Yeah, supervisor okay, I want to use this. Okay probably developers changed some behavior and login page which now made a bit harder to use saved passwords in browser. Okay, so I logged in again. Contact section. Now you will see edit page with no any cache issue. This is example of a freedom UI edit page. You still see some freedom UI capture like possibility to shrink lists more rich components with colors, customizations and so on.

Speaker 1

But so we will study how to build such pages. You will know and if some one of you already passed our customization course, you already know how to do such things. So this is example of a freedom UI user interface accounts also shown as a list and edit page. So contacts represent physical persons with names, emails, addresses and so on. Accounts represent companies like Samsung, like Tesla, like others. So some companies that you may work with and you need to keep information about them in your system. This was example of customer 360 installed into our app. Into our system. I have a couple more tools to be installed. Let me quickly show it for you. It would be end of today's session, so not so hard.

Speaker 1

It's still a bit strange that application hub is empty here, let me reload this page because something unusual. Yes, now it's better. Now you see customer 360 is already installed and we will study what an application is, how it works. Now we will just install a couple more apps from our marketplace because we will need them in future examples. So marketplace powered. Here I need tool named data binding. I will search by the word bind easily finding this data binding tool. Select it, click install button. Thanks for automation. Here confirm and install. This installation will take a bit longer because it will need to compile my environment during this installation process. So this takes a bit longer but it's still okay. And I need this data binding tool because it helps to create data items easily.

Speaker 1

So data items can be automated and you can create them with several clicks. And it's easier than to make everything manually. So data binding tool takes some time to install. It's pretty awesome. Old tool, I think it exists maybe for five years and it's designed for an older version of creature and it also has a lot of support in classic UI. But now mainly we needed to support our creating of data items and this help. So this tool helps to save our time and to reduce boring part of the work. You can download installation log to see how long it took to load this. Let's check it out. Our session is almost over, so now you see 20 415, then 20 516. It looks like 1 minute and 1 second and like 61 seconds took to install this solution. No errors, no.

Speaker 1

Any warnings here? Perfect. Everything is okay. Everything is okay. So our app was installed and one more final thing so we can reward this. And you see data binding item, one more item which I want to install from Marketplace is a special tool called maintenance tools for creation maintenance. This maintenance tools was made by partner which previously was named Siso. You probably saw its icon, now it's named neo technologies from Switzerland, and they use just a standard creature API. And this tool has two major functions, clear redis cache. But you don't really need it often. But the most important and useful feature is restart the app restart application. Unfortunately, creature doesn't have this function out of the box available in user interface. And partners from Neo technologies decided to fix this issue and they just made a UI button to do this.

Speaker 1

So this app simply adds a couple of buttons into our creation user interface to make it easy to restart the app or to clear redis cache. I will use this restart app, we will need it, we cannot avoid it unfortunately. And in our examples you will see it. So you see 2708 started, 20 718 finished. So it took only 10 seconds for installation. You probably will ask me about why am I so obsessed with installation time? This is important because this is 10 seconds of downtime for your system. If you do some kind of installations on your production, your end users will not be able to operate with your system during these installation procedures. So we still have non zero downtime.

Speaker 1

And this is important to track how long it took for creation to install something, especially if you do it in the middle of working hours. So my installation was successful. We can also go back again to application hub. Now you see one more app, maintenance tools and I can show you where this maintenance tools can be found. It will be into our settings page. Physically it's called system designer, but now developers decided just to call it settings. At the end of this page you will see new buttons. Originally we had only advanced settings, but also now we have clear redis and restart application. We will need restart application later and unfortunately some, let's say study like learning examples will require this restart application action and we cannot avoid it. In some cases you will see it later and today's session is over.

Speaker 1

Thank you for your time. Today we moved forward with working with objects. We discussed how to build data items, how to operate with lookups. We also worked with Git. And tomorrow I plan to show you one more example with

database view and we will move on with building user interface. I plan to focus on Freedom UI, mainly on freedom UI and if we will have some significant demand on Quest AQI stuff. For example, you expect to work or support existing Quest AQI solutions. We will have to take time also to discuss some classic UI tools, but mainly my focus will be on Freedom UI and programming inside of it. So thank you for your time. Today our session is over in case if you have any questions, feel free to ask.

Speaker 1

Now your homework, as usual, is to watch this video to represent all the steps or produce all the steps for today's session. Working with your package, creating new lookup, new object, register it as lookup and practice with data items. So this will be your homework and tomorrow we will continue doing more complicated things. Thank you. Thank you for your time. Questions, paddy? Yeah, yeah, I do. Just with the apps installed. Are they installed then similar to how you would set up a packet on the backend? Oh yes. Physically they represent as packages. If you reload our configuration section, you will see that we have a couple of more packages here. And for example, the package named GLB data binding represents our data binding tool and maintenance tools. Let me find it. It's called CSL maintenance tools.

Speaker 1

Is changes customizations in our settings page to enable couple of buttons. Yeah. And customer 360 c 3300 okay. CRT customers, it's surprisingly low. I need a bit more. Okay, let's search just for 360. Yeah. Customer 360 represents three packages in one archive. So we will study how application works, how it is created and what is inside. Tomorrow you will see that application finally keeps some packages behind it and when we load such applications, physically will load one or several packages. Here you see customers 360. It includes several packages. So that's how it works. And physically application is a bit artificial thing. Physically we operate with packages. So anyway you can see everything which was customized here as a package and the package contents inside of configuration section. Thanks. Thank you. Any other questions please? Yes basil, thank you.

Speaker 1

When I am going to advanced setting, the configuration page doesn't work, it just reload. It's because you're probably using Windows home. If you windows home you have to close previous page. In this case, configuration page will be working normally. Yeah. Let me, I have no options how to make it better, unfortunately. Probably you can consider you can upgrade to Pro Windows Pro version. In this case you will have no such issue. Sure. This is kind of limitation connected with WebSocket and this is something that makes home different from pro and it's not so comfortable. I agree. Thank you. Thank you. Okay, and it looks like Tushar had a question. Yes tushar please.

Speaker 1

My question is like if we have a new lookup, like if we have two, three more lookups and we want how we can then update it and like a system designer, do we have to create the data file again? Or we can update in the existing data file which you already showed. So probably you are talking about new data records added to existing lookups table, am I right? No. Like if you suppose if total number of lookups are 121 right now, if we add like five new lookups, then how we can bind those? Oh, if you have five new lookups, you have finally to create ten new data items or maybe less than ten because one data item for lookups can include many records for registering.

Speaker 1

So the minimum amount of data items in your case will be six, one for five lookup records and five others for each separate data contents of your lookups. So simply speaking, any lookup should include data item for registering of the lookup and data item for the contents of the lookup. In this case it will be the most comfortable for your end users because they will always find all your lookups in production and lookups section and they will be able to open it to see their contents normally and data items guarantee such transport of records. Okay, thanks. Thank you Tashar. And that's all for today. If you will have more questions. I understand that probably questions will appear

later when you try to practice. So you may use my email. Please send messages. I will try to find time to answer you.

Speaker 1

So our tomorrow session will be as usual and we will continue working with objects and we will start to work with user interface programming. Thank you very much for your time. See you tomorrow. Bye. Good luck. See you tomorrow. Bye. Thank you. Tomorrow. Thank you. Bye.