let's discuss what is lookup method injection in Spring Framework and why do we need it. Alright, I believe everyone know about Bean Scope in Spring Framework like Singleton, Prototype, Request and Session. If not, let me give you some heads up about Singleton and Prototype Bean Scope to understand why do we need this lookup method injection. So, let me go to my IntelliJ IDE. So, I created two Bean, UserService and WeatherService. So, if you observe here, UserService Scope is default which is Singleton and WeatherService Bean Scope is Prototype. So, to demonstrate Bean Scope, Singleton and Prototype, I created a test class.

So, you can see this is the test class and having two test methods. The first test method is for Singleton Bean Scope test scenario and second one for Prototype Bean Scope test scenario. So, if you will go to the first test scenario, here I am creating two different object of same UserService Bean whose scope is Singleton. Right, and then I am asserting hash code of each instance and I am expecting it should return true. So, let's run the test case then we will deep dive further. So, let me run this first test case. So, if you observe the test scenario is succeed. We are able to compare both the hash code and we are getting the result object. So, the reason behind it, when first time we create the object in line number 20, then that Bean definition cast in IOC container. So, when next time I am creating object with same ID, then immediately Spring search its metadata and return the instance which is already created. That is the reason both pointing to the same reference here. Hope this is clear for you. Since the scope of this UserService Bean is Singleton. Spring reuse it rather than create a new one. Now, let's move to the next test scenario that is prototype Bean scope. Ok, so let me minimize this. This is what the second test scenario prototype Bean scope and here if you observe again on line number 27 and 28. I am creating two different object of WeatherService and if you observe the WeatherService Bean scope is prototype here. That is what we defined. Right, now here I am asserting the hash code of both. The. Object and I am expecting it should return false. So, again let's run the test case and then we will understand further. Let me run this. So, if you observe test scenario is succeed and both the WeatherService 1 and WeatherService 2 hash code is not equal. The reason behind it, if Spring found any Bean with prototype scope, then immediately it creates a new object.

Hence, in both the line, line number 27 and 28, it create two different object. That is the reason. Hash code is not equal for both the instance. Ok, so if you define a Bean scope as a prototype, as many as request you will send, Spring will always create a different Bean or different object for that. But in case of Singleton, the Bean will initialized once. Ok, it means as many as you will send the request, you will get the same Bean instance or same reference. Hope this is clear for you. You understand how Singleton scope and prototype scope work in Spring context. Right. Now let's move to our actual discussion. Let's say, if interviewer ask you, what are various Bean scope available in Spring, then anyone can explain. But if interviewer tweak the question slightly and ask you, how can you inject one prototype Bean into a Singleton Bean. Then again, being an interviewer, I can say, maximum candidate unable to answer it. Even though someone answered it, but they don't know the solution or the approaches how to overwrite this. That is what we will demonstrate in this video. In today's session in multiple approaches. Ok. So without any further delay. Let's get started. So let's go to the each Bean. Let me close this test scenario. I'll also minimize this console. Now if you observe. This weather service is been scope as a prototype. and I have one simple method will return the today's temperature so the temperature is something the random number and the current timestamp okay this is what some get today's temperature method is there in weather service and please observe the scope of this particular bean is prototype now I am injecting this bean inside this user service bean so you can see here right I am just injecting but the scope of this user service is singleton so what happened when you will inject a prototype bean inside a singleton bean that is what will understand so first let's try to find out the problem then we'll go with the solution so here I am NOT doing anything in user service simply I am just injecting the weather service and I'm calling the same method then I created one controller class to demonstrate the output so here I am just calling the user service dot get current temperature from other API then again Nigeria the thread will sleep for one second then again i am doing the same call user service dot get current temperature from weather api so on the first call it should return us some random temperature and the current timestamp again after one second sleep it will return some different timestamp and different temperature right then i am just consolidate the result and i am returning it as a list so let's see let me run this application i will go to the i will just right click here run this particular application so you can observe it started on port 8080 let me go to the browser i will type localhost 8080 slash scopeokay i will just enter it so if you observe here we are getting the same output the timestamp didn't changed and the random number value also didn't change so if you observe here we are getting the same output the timestamp didn't changed and the random number value also didn't changed and the random number value also didn't changed so what i will do i will just copy this i will just go to the intellij id i just want to show you that how it behaves okay so in first call this is what the result then after one second rest or the thread slipped it should give us some different response right so this should be 42 this should be 43 because there is a pause for one second and this random value should be something in between 60 but the result is not coming here let me try once again i will just refresh this still the output is same but if you observe my timestamp changed to the 1344 but still we are getting here 42 right i'll again run it we are getting the same output that's what we found the problem it occurs because both the bin will initialized once that means prototype bin lose its behavior and act as a singleton okay if you will go and check the code this weather service which is the prototype and this user service which is singleton so when we injected this now this user service wrapped with this weather service okay i mean this weather service is now act as a singleton that is why you are getting the same output again and again i will just run it again you are getting the same there is no change right now how we can overcome this scenario or how we can override this bin behavior that is where lookup method injection concept came into the picture so i will show you multiple way how you can override this behavior so first let me go to the code let me stop the server so here rather than inject it directly what i will do i will just use application context application context then i will simply auto add this okay i will just come in this piece of code because i don't want to inject the prototype bin directly inside a singleton because we saw the behavior right the results are the same here the state is not changing here ok so what i will do i will just comment this as well i will just do return application context dot get bin so i will directly give the class name which is weather service dot class dot get to this temperature now let me run this so application started again just go to the postman or the browser since this is the get api i can directly hit this particular url now you can see here there is a change 48 and you can see here the changes on 01 02 which is second and the random value is 41 and 48. 03 okay now again i will just run it we are getting 16 then 17 and the random value is getting changed 925 let's run it again we are getting the different result 24 25 and then 59 40 which is the temperature right so we are good now we found the solution but the approach what we used is not recommended because here we are just bypassing the principle

of inverse of control. rather than spring create object for us we are just creating the object by our self which is not the recommended way now let's move to the second approach ok so there is a class given by spring that is called object factory so i will just use that object factory you can see here org dot spring framework dot bins so just use that then define the class which is prototype. ok. i will just define weather service object factory then again i will just comment this i will just do return weather service object factory dot get the object then call the method which is get today's temperature then again let me stop and re run our application so it started let's hit the endpoint let me check it's not started ok it's my bad i didn't auto edit. let's re run it again just go to the browser and then just hit the endpoint we are getting the different result right 48 49 there is a change on the time stamp as well as the temperature which is the random number let me run it again yeah we are getting the different time stamp as well as the random number now again there is a problem with this approach so in this approach we are using the object factory. ok. this method is always eager initialize even though we will not use this weather service bin still it will create the object and occupy memory in ioc container so this is also not recommended approach so let's move to the next approach so what i will do i will just create a method public i will return weather service object ok so let me define the method name get weather service bin. something like that then simply I will just return null and what I will do I will just annotate here lookup that is what the method introduced in string framework to use the lookup method injection now I will just simply comment this then what I will do next I will just do return get weather service bin dot get today's temperature now if you observe here I am just returning null okay so it means this should return me the null then this particular method call should throw me the null pointer exception but that is not the case what happened internally spring uses cglib library to dynamically override this method to provide its implementation that concept something similar to the dynamic proxy this is how the spring generate code so I will show you how internally spring will generate the code let me add this piece of code here we are returning the null in our code right but under the hood what spring will do it will just use application context dot get bin of your class if I will I will write it manually then again I am just bypassing the principle of IOC right so that is what the spring will does internally when you annotate a method with at the right lookup spring will create the object of it and will return its implementation or you can write it as a reference okay so I will just remove this now we are good let me run our application so it started just go to the chrome and just run this endpoint we are getting the different number right 4748 there is changes on the timestamp as well as the random number which is our temperature let me run it again yeah we are getting the result as expected now if we will go to the code. there is some certain limitation to use this lookup method injection so the bin which you are injecting using lookup annotation should not be final this should be public okay no final modifier should be applied for this particular bin and then the method which you define that should not be private or static or final these are the two limitation to use the lookup annotation to perform lookup method injection. so I hope you are clear about this lookup method injection and you know the complete context how you can inject a prototype bin into a singleton bin and also I explained how and when to use this lookup annotation in your spring application right so do let me know in comment section if you guys have any doubts that's all about this particular video guys thanks for watching this video. meet you in the next video. see you soon with a new concept.