CS 240: Maven Transcript

[00:00:00] Now that we've talked about why command line builds are important. Let's talk about uh the tools that can be used to create automated builds.

[00:00:08] Most language environments have tools that are specifically targeted at automating the build process for projects that are implemented in that language.

[00:00:19] And so this slide lists some of the common tools that that people use.

[00:00:23] Uh For example, in Java, um there's tools like Maven and Gradel and in the C C++ world, we have made uh if you're a JavaScript programmer, you've certainly heard of NPM.

Start visual description. The professor demonstrates the use of various tools for automating the build process in different programming languages, such as Maven and Gradle for Java, Make for C/C++, and NPM for JavaScript. End visual description.

[00:00:36] Python has a set of tools that are very similar poetry pie builder.

[00:00:42] So those are language specific tools and then more universally, if you don't use a tool like that, you can just write shell scripts.

[00:00:49] So in any operating system, you can write uh shell scripts that run command line commands um in a batch kind of mode.

[00:00:58] And so you know, if you wanted to, you could automate builds using um shell scripts or you could use a combination of these tools listed above and then shell scripts as well.

[00:01:08] So in, in the concept is really just writing programs that automate your build process.

[00:01:13] But these tools make it really easy to, to, to do that.

[00:01:17] Now, we're going to focus on maven in uh our discussion here and the reason we're, we're doing that is because you're actually using Maven in the chess project, whether you, you knew it or not.

Start visual description. The professor explains how IntelliJ uses Maven under the covers to manage the project, compile code, and run test cases, even though the user interacts with IntelliJ's graphical interface. End visual description.

[00:01:29] Um I don't know if you've ever wondered, you know, how does the auto grader fetch my code for my git repository and then compile it and test it, compile, run it, test it and do all those things and it doesn't use Intel J at all to do that.

[00:01:45] Uh Intel J is just not involved in that process at all.

[00:01:48] And so how, how is the auto greater able to do that? Well, the answer is Maven, it turns out that your, your chess project is a maven project underneath.

[00:01:57] And what that means is that while you're, you're using Intel's graphical user interface to edit your code, run your code, debug it and so forth.

[00:02:07] Um Intelli actually under the covers is using Maven to manage your project.

[00:02:15] For example, it could use maven to um to compile your code, it could use Maven to run your test cases, uh things like that.

[00:02:24] And so um your chess project is actually using command line builds. It's just that intelligence is hiding that from you.

[00:02:33] So that as you work through its graphical interface, you usually don't see that.

[00:02:37] But uh if we look under the covers a little bit. We can see that uh you can do command line builds with your chess project, whether you knew it or not.

[00:02:45] And so let's do a demonstration of that um Real quick.

[00:02:48] So here I have a, a terminal that um if, if I uh look at the contents of this directory, this is just the root directory of my, my chess project.

[00:02:59] And it has the typical familiar structure. We have client, we've got server, we've got shared and so forth.

[00:03:08] Well, let's say I wanted to compile my project without using IntelliJ .

[00:03:13] Well, I can use Maven, which is a command line tool.

Start visual description. The professor shows how to compile a project using Maven from the command line by running the command mvn compile, which compiles all the source code for the project. End visual description.

[00:03:17] So I'm assuming here that I've installed Maven on my computer and that's easy to do if you don't have Maven on your computer.

[00:03:24] Uh Well, you actually do have it um whether you knew it or not.

[00:03:28] Um But you can easily install Maven if you don't have it.

[00:03:31] And the, the command that you run for Maven is just MBN.

[00:03:35] And in this case, I can just run MBN compile and that will compile all the source code for my project.

[00:03:48] So that just runs a Java compiler on all the um the java files.

[00:03:55] Let's make this a little shorter. So, we can see, but you can see here it's all green, which is good success, success, success, et cetera.

[00:04:03] And so it's all green. Of course, if things fail, you'll see lots of red Um So just with one little command line command, I can compile all my code.

[00:04:13] Um What else can I do? I can, I can test it if I want to run all my G unit tests on my code.

[00:04:20] Well, to do that, they're going to have to compile my code first, but once they've compiled my code, then they have to compile the test cases and then they can run the test cases and see if they work.

[00:04:29] And so when I run MBN test, that's going to, that's what's going to happen.

[00:04:38] So it's building the code, it's run the tests.

[00:04:42] Hopefully they all work.

[00:04:49] There's quite a few tests by the time we're done with this.

[00:05:17] OK? So, we can see that all the tests succeeded, which is great. So, this is, this is what the auto grader is doing with your project.

Start visual description. The professor demonstrates running all JUnit tests on the code using the command mvn test, which compiles the code, compiles the test cases, and runs the tests to verify if they work. End visual description.

[00:05:24] When it, when it passes you off, it's running and commands on your source code.

[00:05:28] And then uh that's how it verifies that your, your stuff works.

[00:05:32] Now, when I run maven, I tell it what I want it to do. So, the first command I ran was to even compile.

[00:05:44] So whatever you put on the command line here as an argument that's called a goal.

[00:05:48] So in this case, the goal is to compile the code. And so, I just type in compile.

[00:05:53] So whenever you run Maven, you have to give it a goal and that tells it what to go do, I could also put another goal on that command line, which is test.

[00:06:01] So you can put more than one goal on the command line.

[00:06:04] But basically, you're un maven and you just give it the names of the goals which represent the things that you want to do.

[00:06:09] So it's pretty simple.

[00:06:11] Um Another thing I can do with maven, another goal I can give it is to clean.

[00:06:17] Now, clean is a typical goal in a, a software project.

[00:06:20] And what clean means is go delete all the files that were created by the compiler and the other tools when my project was built.

[00:06:28] So anytime you build your project, it's going to compile your code which creates class files or object files or executable files or jar files or lots of files that are derived from your source code.

[00:06:39] So whenever you run the clean goal, you're basically saying, go delete all the, the derived files and, and just leave the source code.

[00:06:48] So one way to think of it is it goes and deletes everything that's not in your git repository.

[00:06:53] So if I run clean here, they'll just go wipe out all the drive files. So that's easy enough.

[00:07:03] So that's, that's how you run ma even from the command line.