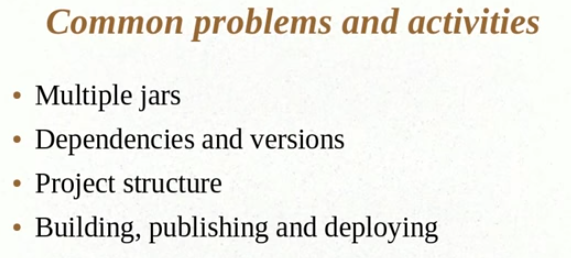
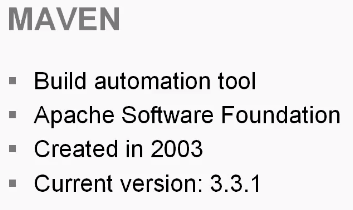
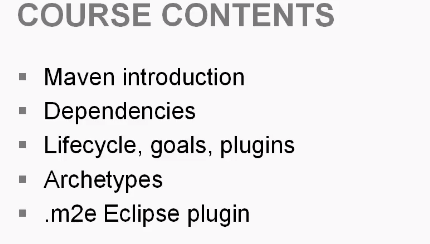


Helps to generate reports

It helps in dependency management





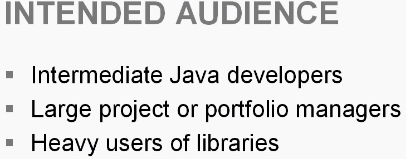


**Archetypes:**

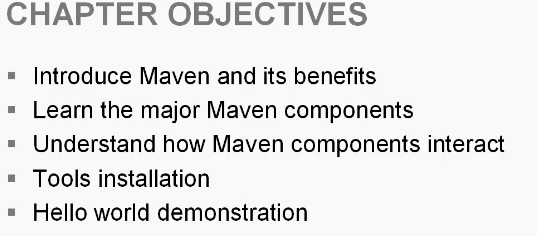
* project templates that can help us generate projects more quickly
* create some reusable project that we can share across our organization.

**m2e eclipse plug-in:** integration within eclipse that helps us when we're working with maven

.net or C sharp and that won't be a problem but the ecosystem has really lent itself to java developers



XML: to configure maven you will see that we have a pom.xml file and that's the configuration file read by maven and it primarily uses xml





Maven is a build tool. You might be familiar with something like make or ant .These tolls take our source code and generate some sort of library maybe it's a jar file or an ear file or a war file. They package our code into that distributable library

maven is also a dependency management tool and this is the favorite feature of many maven users.We build these applications and these applications rely upon third-party libraries so maybe you use spring or hibernate well using maven we change the way we obtain those dependencies and the way we manage them next we'll talk about how maven is a project management toll now this isn't one of the strongest project management tools I've seen but maven does allow us to include some information about the software we're building you know such as the version number or maybe who was working on it and it also provides just a good overview of what is being used within our project next we see that maven provides a standard approach to building software this is kind of a theory behind maven if we have five projects were building and we are using maven in each one of those projects we will see a consistent approach to each project and that gains us some efficiencies just because the projects are so consistent now I've have two more bullets here and there simply to show you how we use the tool maven is primarily a command-line tool so you open up a command prompt and you have made it on your path and you're issuing commands to maven and it's doing some sort of work for you that helps you build your application so that's one way but we can also use the ide there is integration for major ides such as eclipse or netbeans and we'll take a look at that later on in our course so let's take a closer look at build tools because if you don't understand a build tool you won't understand maven we talked about how there's this primary goal of taking source code files and making these deployables artifacts so our jars are Wars our ears that's the primary goal but there's also a lot more to that maybe we have certain other steps that go along in our build maybe we want to run some unit tests maybe we want to generate some javadocs there's all sorts of things that we could want to occur when we build our software products it's important that when we perform that build we hit all of those steps and the best way to do that is to have an automated build process or something that is repeatable and that's where are built tools come in they allow each developer to essentially kick off the same build it's not manually performed so we know steps are not missed and we know that a developer can easily come onto a project and as long as they know how to use our build tool they can essentially build the project from source so we get some advantages there now what you also see built into a lot of build tools and into the build files is that are artifacts are then sent off to a server in the case of a war or an ear file or you can also store that artifact like a jar file somewhere where it can be obtained and these are all things that may even does as a build tool now you may think well that's great but you know I can step through and compile my classes in my ide and I can use the Java compiler to compile i can use it to build myself a jar file or a war file well the idea is to step back out of your IDE or from using those rudimentary tools and put it into one tool that can be used by any developer so maybe you're using eclipse and you're able to build a jar file using it but somebody else may be using netbeans you're both able to build a jar file but the idea behind a build tool is to make the build ide agnostic and to make it repeatable now one of the great things about maven is there is integration with other build tools like Hudson like bamboo so by using maven you can then in turn work with other tools that are going to provide you with other benefits so let's move on and discuss dependency management as i mentioned this is one of the most popular aspects of mavin we talked about pulling in something like the spring framework when we use maven we reach out to an online repository and we pull down the framework from that repository so we don't have to go out and search on the website and download the jar file and pulled in we can do that through maven now the great part about that is if the library were pulling in let's say spring has defined its own pom file then we know Springs dependencies because just like our application depends upon spring spring can also depend upon other libraries and when we use maven maven will reach out and grab the dependencies of our dependencies and those are known as transitive dependencies and then we'll also see a thing called dependency scoping within maven sometimes you may not want to use a dependency at a certain point so maybe a dependency use in my test is not use when I actually deploy my web application so you can specify the scope of a dependency and that will conditionally included at certain points moving on there's also this project management aspect of maven and it's pretty simple week have a palm file which we'll talk about later but it's an xml file and it can list pretty much all of the information we need to know about our project and some of the things we can specify are the version of our project the developers on the project the website for the project so we can find a catalog that information you also see that we can reach into source code repositories and we can generate changelogs depending upon the diffs between different versions of our source code maven can also be used to generate documentation about the software we're developing and this is all automated during that build process we can also create Java documents that will provide information about the code that we have developed so for developing a library that other people are going to be using a javadoc could be a very useful artifact for that individual also we can generate reports that give us all sorts of information about our project so maybe we run a clover code coverage report and that can tell us what has been tested what hasn't been tested and those are all useful from that project management perspective now finally we're going to talk about how maven is used to standardize our builds so as we discussed if we have multiple projects using maven there's this uniformity across those projects and the way it's achieved is through patterns now don't go off and think about like factory patterns or builder patterns this is very simple its kind of having like a standard directory structure so maven knows where we're going to place different things and this leads into convention over configuration maven is opinionated in ants you may say well all of my source code is in this directory mavens approach is you're going to place all of your source code within this directory so it dictates a lot of the structure of our project and that achieves a lot of Conformity across a portfolio of projects so you're going to see a can distance e throughout them and you're also going to see a consistency in the path taken to build these projects so it believes strongly that there is going to be a compile phase a test phase deploy phase and those are going to appear in the life cycle of every project and we're going to discuss the lifecycle little bit later but it's just good to introduce this concept now so you have an understanding that a lot of maven projects are going to look the same and they're going to follow the same processes have the same type of layout and that's really where the power is in maven and this leads into the philosophy of mavin now there's a very short article on mavens website that basically describes their motivations for creating maven and it was actually kind of because they had all these Apache project so maven is an Apache project and it was created because all of the Apache projects were kind of different if i was a developer on one project and i moved to another I had to learn a completely different build system and it might be laid out differently and maven was created to solve all that so I would urge you to go into your working files and follow the link to the philosophy of mavin as we move forward in the course we will explore a lot more of these topics in detail

