1. **SharpZibLib:** This library creates Zip, GZip, Tar and BZip2. If you have to deal with files, whether you are backing files up, moving files, taking files from a FTP site, unzipping them. Whatever it is, you can do it all using this library.
2. **FluentEmail:** It is a really simple Email system. It has the ability to connect to SendGrid, Mailgun, Mailtrap, MailKit, SMTP server. You can add templates to an email using this library. If you need standard Email stuff then FluentEmail is recommended.
3. **MailKit:** It is used when you need more in-depth Email client, we need encryption or other things and that’s where MailKit comes into play. It is built on top of mime kit and it’s a cross-platform.
4. **Papercut-SMTP**: They have a client that sits on your computer and when you are building your C# application that uses Email so whether you’re using FluentEmail or MailKit or whatever it is, what you do is you point your email system to localhost and then you run Papercut. So what it does is it acts like a SMTP server, so that you can test your heart’s content (your C# application) and it won’t sending any email out.
5. **EPPlus**: This is a tool for reading and writing Excel files and in business Excel files are everywhere. It is a paid library when you have to deploy your code for commercial purposes.
6. **Hangfire:** It’s all about executing jobs. For example, if every night we have to backup SQL Server so that was a scheduled job which says every night at this time I want you to backup SQL Server. Well there’s the same kind of jobs you might want to do in C#. You might have the C# job execute a SQL server backup or from the first library that SharpZipLib you could say let’s grab all these files and zip them up and archive them back them up somewhere every night. You can create a C# application to do that or synchronize file or clean up your database or archive client records or create reports and the list is endless.
7. **MassTransit:** Recently, we talked about Azure service bus and about using queues. I intentionally use Azure service bus because I found it was the easiest way of communicating that topic without adding another layer of complexity but there is other tools out there for example RabbitMq is another queuing system like Azure service bus but it’s not in Azure obviously. It’s just a third-party assistant installed pretty much anywhere and RabbitMq is another great option. There’s a tool called MassTransit and what it does is it basically abstracts away the differences between RabbitMq, Azure service bus and other queuing system so that you can write your code once and then support any of those systems. So it’s another layer on top of Azure service bus but think of it like .Net Standard where it creates this interface that then you can call any different piece down below.
8. **Polly:** It does some nice things with working with system that might have faults. It does little things not the huge massive things that are so clunky. It’s really focused on what it does for example retry policy that states many faults are transient and may self-correct after a short delay. Let’s say you’re trying to connect to an API and it doesn’t work. So you go to abc.com/api and you get an error (it times-out). That could be the end of the story but with Polly you can say I want you to try again that’s a retry policy. I can say that I want you to retry again in three seconds, so what the system will do is it will wait for 3 seconds and then it try it again to see if AKA “Maybe it’s just a blip”. This allows you to configure automatic retries. The next one is Circuit breaker. You can give a system a break. If you keep doing that retry over and over and over again but maybe it’s just not you maybe you have a lot of different spots in your application that are trying again and they’re all doing retries that can overload a system pretty quickly. So what the circuit breakers says is let’s do a timeout for a minute. Because if we overload the system retrying even if it comes back up it was a little shaky it’s will crash it right back down to his knees because of all of our retries.
9. **Serilog:** This is kind of the new de facto standard for .Net logging. It is usually used in place of log4Net. Log4Net is used to log data but it was mainly about logging text data and that’s kind of what loggers do. So you have an error and the error is file not found. So you might log your text file was not found and may be you put in parentheses what file was you looking for. But usually just one big text block and let’s say you store it in SQL database you’d things like date and time and you’d have what method had the error. Maybe you capture a call stack if there’s an actual exception and then you have your message and then maybe the level of the error. But that’s pretty much what you have captured in your log system. The problem with that is it’s very hard to figure out which files are problems because even if you put that in the message you’d have to do some kind of regex on a text field to see to pull out those file names. That’s not efficient, so the next level is what’s called structured logging where you have that error message but you also break out those pieces into searchable unique objects.
10. **Seq:** It is also used for logging. Logs are not useful if you don’t read them, don’t log what you are not going to read. If you say I am going to log every time a user does anything that would be great but are you going to read that because if you’re not then you’re wasting space and in a lot of ways you are taking up processing time in your own brain trying to find the real issues. Tim’s story, I used to work as an IT director and my network director at the time we decided we did a logging system that would monitor all of our different servers. We have maintained our entire active directory server, an exchange server, a couple of SQL servers and so on. So, whenever something goes wrong we want to know about it and so we bought and installed the logging system that would monitor all these different pieces of information and then what we would do is it could send an email to your inbox saying there’s a problem. So whenever anything bad goes wrong we should know about it. My network administrator dialled in these is the things that could be a problem. So if there’s no disk space we need to know about it, if the network is a little slow we need to know about it, if there’s a problem connecting to the server we need to know about it. When it was turned on we were getting hundreds of emails a day. So what he ended up doing was just saying send all these emails to a folder that he never looked at and we never saw errors because we had so many come in that we didn’t know which ones were real errors and which ones were not really a big deal. For example, a disk space we should know if we are getting down to 10% of our disk space but when the hard drive is a terabyte and you’ve got 100 gigabytes left and it’s an active directory server where it’s not changing that much size that often that’s not a huge deal so we don’t really care but we should turn that alert off but instead of doing that we just ignored that alert well that comes back again and so what happened was we had so many we didn’t see any and we actually had servers that had major problems that we didn’t catch because the alerting system was telling us too much information and it felt like good information but unless you’re going to act on it unless you read every error don’t have it tell you because it’s even worse than not knowing because by doing that you hide everything. So when it comes to logging only log what you’re going to read. Seq helps you read more and the way it does that is by telling you all the information about an error in a structured logging. It will say what the application was, the customer, Id, the environment the order id and all this great information that’s not information you read about every error but when you have an error that information is great to have to allow you to replicate that issue but what it also does is they have dashboards and you can filter and search the identity information but then you can analyze things over time that can be really helpful for things like may warnings aren’t a big deal but they’re indication of some kind of problem that can become an issue. If you track and see the number of warnings go up over time that might be something or might not be but it allows you to start looking into it more depth.