AzureException extends RuntimeException

2003 tries to get me

Onkobu Tanaake

2022-06-27T21:00:00

The project I'm working for updated Azure cloud resource libraries. None of the pre-com.azure-variants uses HTTP Keep-Alive – worth another rant. So everything ought to be better with the improved com.azure things. Then came a review and I spotted some odd warnings of my code checker with runtime exceptions.

Java knows two fundamental types of exceptions. One type is the so called checked exception. For example I/O-operations mostly throw these so the developer knows that there is something that can go wrong and needs to be dealt with. The other group are ancestors of RuntimeExceptions. For example all errors like out of memory or illegal argument errors don't need to be handled explicitely. They simply kill the VM if nobody cares.

When accessing a table storage the old library throws a checked exception TableServiceException. The code must be prepared for. With the move to com.azure libraries TableServiceException is marked as unncessarily handled as a checked exception (in throws declarations). That puzzles me because the big difference is the situation after the exception was thrown. A RuntimeException normally is something serious the code cannot recover from thus no checking and preferrably a global handler shuts everything down or the VM simply dies with a stack trace.

All in all some beginner designer inherited TableServiceException from HttpStatusException. It is even less right because this exception is also thrown when an entity could not be found. Or it is caused by a wrong URL. This is sort of an HTTP error. But it does not seem like an error for me, when an entity is not found. Shure it causes HTTP.404 in a REST interface. But wouldn't an empty ResultSet be sufficient? Going further up in the hierarchy you pass an AzureException that finally inherits from RuntimeException. Is it something with my eyes or did somebody with a degree in 2022 derive a general AzureException from RuntimeException?

Whoever was payed for this totally broken design – never use exceptions (in Java) to control business logic – got away with it. It is released as an official Azure library to access table storage, CosmosDB, EventHub and ServiceBus. Sure it works but at which cost? The previous libraries forced developers to correctly handle exceptions. The new one has protective comments and requires a lot of testing and logging. It felt like concluding ORA error codes from hidden structures in exceptions of Oracle JDBC libraries of 2003.

My recommendation: don't use the (Azure) Cloud yet if your JDBC or MQTT-stack works well. Give something extra to your software engineers supporting you to stay competitive and don't fear of missing out.