talk

Rules Engine

.NET

with .NET

Rules Engine with .NET

.NET

.NET is the free and modern open-source cross-platform framework from Microsoft that enables developers to build applications and services for Cloud, Web, Desktop, Mobile, Gaming, IoT and AI. .NET has many improvements and new features each year with even-numbered long-term releases supported for three years and odd numbered short-term releases supported for eighteen months.

.NET with C# is also updated each year is an open-source programming language for modern software development with support for type safety, asynchronous programming and more! .NET allows developers to build applications for any device and platform such as Windows, Linux, MacOS, iOS, Android and more! Find out more about .NET including the latest version of .NET which comes with the latest version of C# by visiting dot.net.

Console

Console Apps allow .NET developers to create command-line applications that run on Windows, Linux and macOS, they have data streams to show output such as messages and error messages or exceptions to users and have a data stream to allow a user to input data such as keyboard input.

Console Apps are designed for single-window utility applications for Command Prompt, and you can Install .NET Desktop Development Workload for Visual Studio to create Console Apps.

Blazor

Blazor allows developers to use the power of .NET and C# to build full-stack web applications without writing any JavaScript, it is a modern front-end framework with Components based on HTML, CSS, and C# for building web applications.

Blazor Components can be hosted in browser with WebAssembly, server with ASP.NET Core or native applications. Install ASP.NET and web development workload for Visual Studio to create Blazor Web, Server or Hybrid Apps. Find out more about Blazor by visiting blazor.net.

What is a Rules Engine?

Rules Engine is a system that allows a set of actions to be defined that are based on specific conditions in an application, those specific conditions could be configured at runtime without needing to change business logic in an application. When creating implementations of functionality code is often coupled to use cases present at time of development, requirements can or will change so can create an implementation that remains the same regardless of use cases.

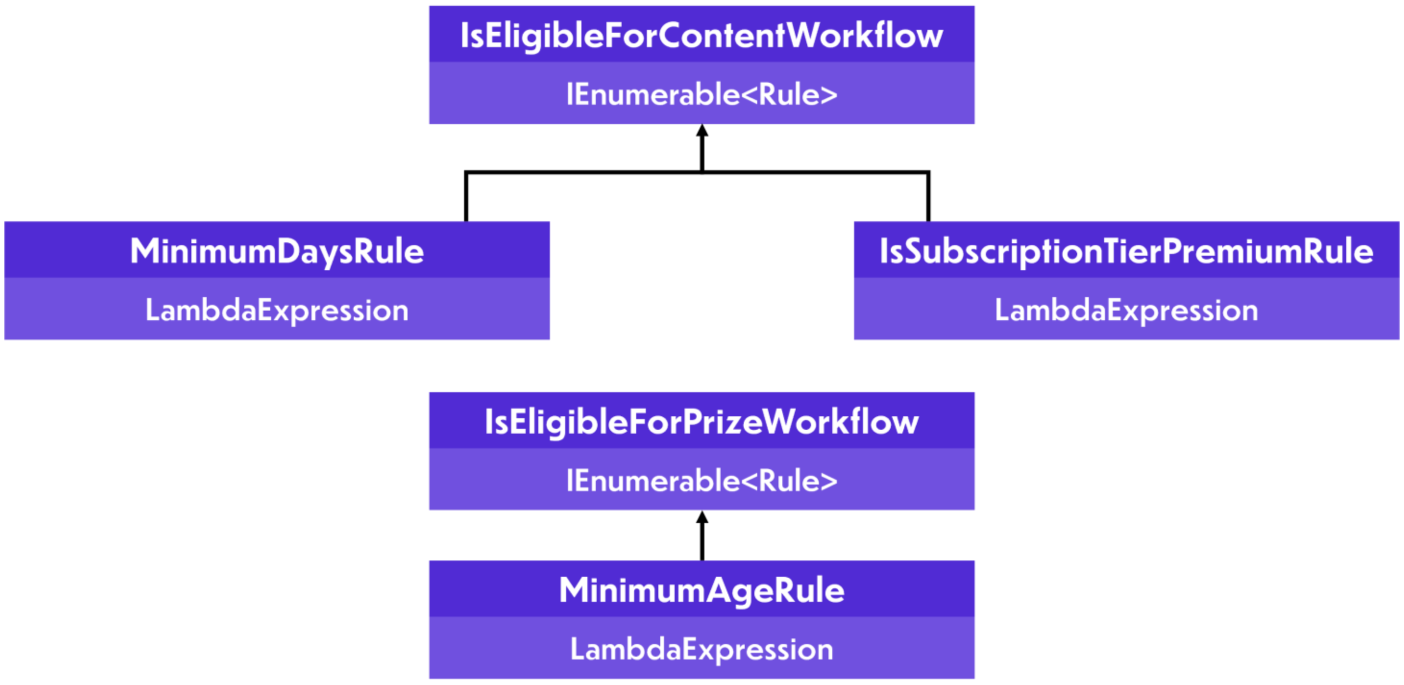
Rules Engines allow you to consolidate business logic if you need the same conditional rules in different places. They will perform the task of checking any inputs against one or more Rules in an application. Rules are a business policy with a set of triggers, conditions and effects applied by the Rules Engine. Triggers determine the context where a rule should be performed or where it should be used. Conditions contain logic to perform checks on the Input that either pass or fail and effects are outcomes when any rules are triggered, and conditions pass.

Source: How to Design Software - Rules Engines by Joseph Gefroh - link.medium.com/IryJavuhznb

Microsoft Rules Engine

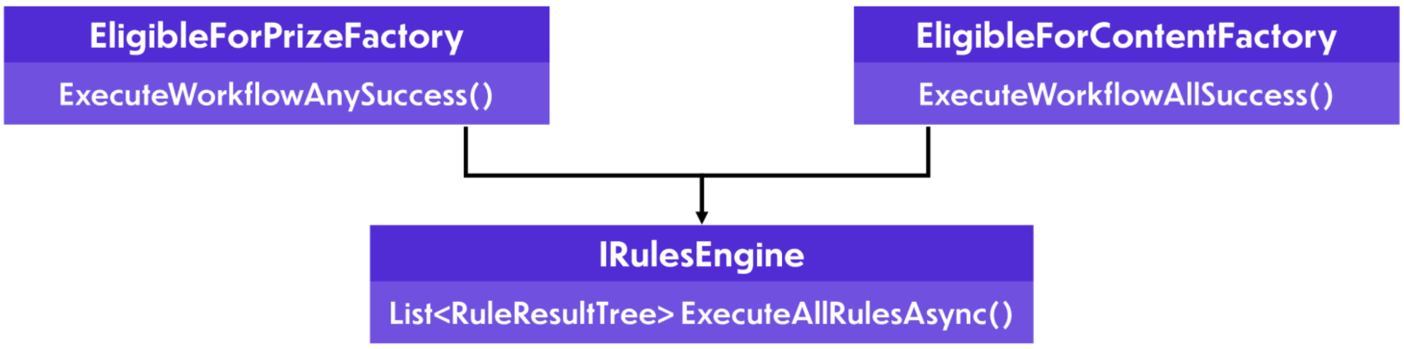
Microsoft Rules Engine is a highly extensible open-source .NET library for building rule-based systems, it supports JSON-based rules using extensive dynamic C# expressions. Microsoft Rules Engine also supports multiple input and dynamic object input, and it can be extended with class or type injection. For more information about Microsoft Rules Engine visit github.com/Microsoft/RulesEngine.

Workflows & Rules



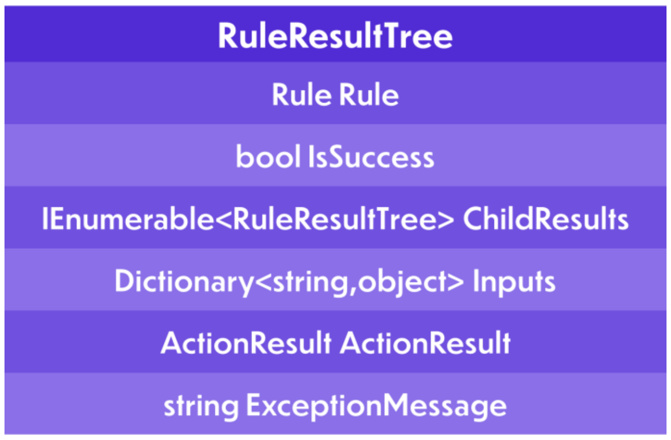
Microsoft Rules Engine defines Rules using C# Lambda Expressions that can be evaluated with one or many input values that if return true will pass or if they return false will fail. Rules can also be provided with functionality such as helper methods or custom types and a collection of rules can then be used together in a workflow. Rules such as MinimumDaysRule can be used to check the registration date of a user and MinimumAgeRule can ensure a user is over a certain age, rules such as IsSubscriptionTierPremiumRule can check the subscription tier of a user with a custom type as the expected value.

Engine & Factories


EligibleForPrizeFactory
ExecuteWorkflowAnySuccess()
EligibleForContentFactory
ExecuteWorkflowAllSuccess()
IRulesEngine
List<RuleResultTree> ExecuteAllRulesAsync()

Microsoft Rules Engine can then execute all rules in a workflow and then you can use a LINQ expression to evaluate the results and factories can be used to encapsulate one or more workflows to be executed for the Rules Engine.

Results & Outcome


RuleResultTree
Rule Rule
bool IsSuccess
IEnumerable<RuleResultTree> ChildResults
Dictionary<string,object> Inputs
ActionResult ActionResult
string ExceptionMessage

Microsoft Rule Engine once the execution has completed and gone through all the rules in the workflows then the output will be a collection of RuleResultTree models, these models contain the rule, if it has passed, any child rule results and inputs being checked along with any exceptions that occurred for a rule. You can control the flow of an application by proceeding on the outcome of any rule that returns success or when all rules return success.

Summary

.NET is the open-source cross-platform framework from Microsoft for building modern applications or services for any device or platform. Rules Engine allows a set of actions to be defined based on specific conditions within an application. Rules can then be evaluated to see when conditions, which could be configurable, are met. Outcomes can then be performed in an application once any conditions have been met. Developers can leverage existing Rules Engine such as Microsoft Rules Engine Library.

About

How to Design Software – Rules Engines - link.medium.com/IryJavuhznb
dot.net
blazor.net
github.com/Microsoft/RulesEngine

tutorialr.com/talks/rules-engine-with-dot-net
github.com/tutorialr/rules-engine-with-dot-net-talk