Namespace LogicEngine

Classes

CompiledCatalog<T>

CompiledRule<T>

CompiledRulesSet<T>

Class CompiledCatalog<T>

Namespace: LogicEngine

Assembly: LogicEngine.dll

```
public record CompiledCatalog<T> : IAppliable<T>, IDetailedAppliable<T,
IEnumerable<string>>, IAppliedSelector<T, string>, IEquatable<CompiledCatalog<T>>
where T : new()
```

Type Parameters

`T`

Inheritance

object ← CompiledCatalog<T>

Implements

IAppliable<T>, IDetailedAppliable<T, IEnumerable<string>>, IAppliedSelector<T, string>, IEquatable<CompiledCatalog<T>>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CompiledCatalog(CompiledRulesSet<T>[], string)

Creates a new compiled catalog

```
public CompiledCatalog(CompiledRulesSet<T>[] ruleSets, string name)
```

Parameters

`ruleSets` CompiledRulesSet<T>[]

`name` string

Properties

Name

```
public string Name { get; }
```

Property Value

string

Methods

Apply(T)

Applies the catalog to an item by looping over the rules sets

```
public bool Apply(T item)
```

Parameters

`item` T

Returns

bool

DetailedApply(T)

Applies the catalog to an item and returns either a list of strings (the codes of the rules that are not satisfied) or a unit

```
public Either<IEnumerable<string>, Unit> DetailedApply(T item)
```

Parameters

`item` T

Returns

Either<IEnumerable<string>, Unit>

FirstMatching(T)

Returns the code of the first rule that is satisfied by the item, None if no rule is satisfied

```
public Option<string> FirstMatching(T item)
```

Parameters

`item` T

Returns

Option<string>

Class CompiledRule<T>

Namespace: LogicEngine

Assembly: LogicEngine.dll

```
public record CompiledRule<T> : IAppliable<T>, IDetailedAppliable<T, string>,
IEquatable<CompiledRule<T>> where T : new()
```

Type Parameters

`T`

Inheritance

object ← CompiledRule<T>

Implements

IAppliable<T>, IDetailedAppliable<T, string>, IEquatable<CompiledRule<T>>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CompiledRule(Func<T, bool>, string)

Creates a new compiled rule

```
public CompiledRule(Func<T, bool> executable, string code)
```

Parameters

`executable` Func<T, bool>

`code` string

Properties

Code

The code that represents the rule

```
public string Code { get; }
```

Property Value

string

Methods

Apply(T)

Applies the rule to an item

```
public bool Apply(T item)
```

Parameters

`item` T

Returns

bool

DetailedApply(T)

Applies the rule to an item and returns either a string (the code if the rule is not satified) or a unit

```
public Either<string, Unit> DetailedApply(T item)
```

Parameters

`item` T

Returns

Either<string, Unit>

Class CompiledRulesSet<T>

Namespace: LogicEngine

Assembly: LogicEngine.dll

```
public record CompiledRulesSet<T> : IAppliable<T>, IDetailedAppliable<T,
IEnumerable<string>>, IAppliedSelector<T, string>, IEquatable<CompiledRulesSet<T>>
where T : new()
```

Type Parameters

`T`

Inheritance

object ← CompiledRulesSet<T>

Implements

IAppliable<T>, IDetailedAppliable<T, IEnumerable<string>>, IAppliedSelector<T, string>, IEquatable<CompiledRulesSet<T>>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CompiledRulesSet(CompiledRule<T>[], string)

Creates a new compiled rules set

```
public CompiledRulesSet(CompiledRule<T>[] rules, string name)
```

Parameters

`rules` CompiledRule<T>[]

`name` string

Properties

Name

```
public string Name { get; }
```

Property Value

string

Methods

Apply(T)

Applies the rules set to an item

```
public bool Apply(T item)
```

Parameters

`item` T

Returns

bool

DetailedApply(T)

Applies the rules set to an item and returns either a list of strings (the codes of the rules that are not satisfied) or a unit

```
public Either<IEnumerable<string>, Unit> DetailedApply(T item)
```

Parameters

`item` T

Returns

Either<IEnumerable<string>, Unit>

FirstMatching(T)

Returns the code of the first rule that is satisfied by the item, None if no rule is satisfied

```
public Option<string> FirstMatching(T item)
```

Parameters

`item` T

Returns

Option<string>

Namespace LogicEngine.Compilers

Classes

RuleCompiler

RulesCatalogCompiler

RulesSetCompiler

Class RuleCompiler

Namespace: LogicEngine.Compilers

Assembly: LogicEngine.dll

```
public class RuleCompiler : IRuleCompiler
```

Inheritance

object ← RuleCompiler

Implements

IRuleCompiler

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Compile<T>(Rule)

Compiles a given Rule into a CompiledRule<T>. It transforms the rule into a lambda expression, compiles it into a function, and wraps it in a CompiledRule<T> object. The method returns an Option, which contains the compiled rule if the compilation is successful, or None if it fails

```
public Option<CompiledRule<T>> Compile<T>(Rule rule) where T : new()
```

Parameters

`rule` Rule

Returns

Option<CompiledRule<T>>

Type Parameters

T

Class RulesCatalogCompiler

Namespace: LogicEngine.Compilers

Assembly: LogicEngine.dll

```
public class RulesCatalogCompiler : IRulesCatalogCompiler
```

Inheritance

object ← RulesCatalogCompiler

Implements

IRulesCatalogCompiler

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesCatalogCompiler(IRulesSetCompiler)

```
public RulesCatalogCompiler(IRulesSetCompiler rulesSetCompiler)
```

Parameters

`rulesSetCompiler` IRulesSetCompiler

Methods

Compile<T>(RulesCatalog)

ompiles a RulesCatalog into a CompiledCatalog<T>. It filters out any null RulesSets, compiles each valid RulesSet using the IRulesSetCompiler, and then constructs a CompiledCatalog<T> with the compiled rule sets and the catalog's name. The result is wrapped in an Option to handle cases where no valid rule sets are present.

```
public Option<CompiledCatalog<T>> Compile<T>(RulesCatalog catalog) where T : new()
```

Parameters

`catalog` RulesCatalog

Returns

Option<CompiledCatalog<T>>

Type Parameters

`T`

Class RulesSetCompiler

Namespace: LogicEngine.Compilers

Assembly: LogicEngine.dll

```
public class RulesSetCompiler : IRulesSetCompiler
```

Inheritance

object ← RulesSetCompiler

Implements

IRulesSetCompiler

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesSetCompiler(IRuleCompiler)

```
public RulesSetCompiler(IRuleCompiler singleRuleCompiler)
```

Parameters

`singleRuleCompiler` IRuleCompiler

Methods

Compile<T>(RulesSet)

Compiles a given RulesSet into an TinyFp.Option<A> of CompiledRulesSet<T> by compiling each individual rule in the set using the IRuleCompiler, filtering out any invalid rules, and then creating a new CompiledRulesSet<T> with the valid compiled rules and the original set's name. If no rules are valid, it returns None.

```
public Option<CompiledRulesSet<T>> Compile<T>(RulesSet set) where T : new()
```

Parameters

`set` RulesSet

Returns

Option<CompiledRulesSet<T>>

Type Parameters

`T`

Namespace LogicEngine.Extensions

Classes

EnumerableExtensions

Class EnumerableExtensions

Namespace: LogicEngine.Extensions

Assembly: LogicEngine.dll

```
public static class EnumerableExtensions
```

Inheritance

object ← EnumerableExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Filter<T>(IEnumerable<T>, IAppliable<T>)

Filters the collection based on a given IAppliable<T> implementation. It returns only the elements on which Apply(T) returns true

```
public static IEnumerable<T> Filter<T>(this IEnumerable<T> enumerable, IAppliable<T>
app) where T : new()
```

Parameters

`enumerable` IEnumerable<T>

`app` IAppliable<T>

Returns

IEnumerable<T>

Type Parameters

`T`

FirstOrDefault<T>(IEnumerable<T>, IAppliable<T>)

Takes an IAppliable<T> instance and returns the first element in the collection that satisfies the condition defined by Apply(T). If no such element is found, it returns the default value for the type T

```
public static T FirstOrDefault<T>(this IEnumerable<T> @this, IAppliable<T> app)
where T : new()
```

Parameters

`this` IEnumerable<T>

`app` IAppliable<T>

Returns

T

Type Parameters

`T`

Namespace LogicEngine.Interfaces

Interfaces

IAppliable<T>

IAppliedSelector<TIn, TOut>

IDetailedAppliable<T, TOut>

Interface IAppliable<T>

Namespace: LogicEngine.Interfaces

Assembly: LogicEngine.dll

```
public interface IAppliable<in T> where T : new()
```

Type Parameters

`T`

Methods

Apply(T)

Applies the item to the appliable

```
bool Apply(T item)
```

Parameters

`item` T

Returns

bool

Interface IAppliedSelector<TIn, TOut>

Namespace: LogicEngine.Interfaces

Assembly: LogicEngine.dll

```
public interface IAppliedSelector<TIn, TOut> where TIn : new()
```

Type Parameters

`TIn`

`TOut`

Methods

FirstMatching(TIn)

Returns the first matching item

```
Option<TOut> FirstMatching(TIn item)
```

Parameters

`item` TIn

Returns

Option<TOut>

Interface IDetailedAppliable<T, TOut>

Namespace: LogicEngine.Interfaces

Assembly: LogicEngine.dll

```
public interface IDetailedAppliable<T, TOut> where T : new()
```

Type Parameters

`T`

`TOut`

Methods

DetailedApply(T)

Applies the item to the appliable

```
Either<TOut, Unit> DetailedApply(T item)
```

Parameters

`item` T

Returns

Either<TOut, Unit>

Namespace LogicEngine.Interfaces.Compilers

Interfaces

IRuleCompiler

IRulesCatalogCompiler

IRulesSetCompiler

Interface IRuleCompiler

Namespace: LogicEngine.Interfaces.Compilers

Assembly: LogicEngine.dll

```
public interface IRuleCompiler
```

Methods

Compile<T>(Rule)

Compiles a rule into a compiled rule, None if the rule is invalid

```
Option<CompiledRule<T>> Compile<T>(Rule rule) where T : new()
```

Parameters

`rule` Rule

Returns

Option<CompiledRule<T>>

Type Parameters

`T`

Interface IRulesCatalogCompiler

Namespace: LogicEngine.Interfaces.Compilers

Assembly: LogicEngine.dll

```
public interface IRulesCatalogCompiler
```

Methods

Compile<T>(RulesCatalog)

Compiles a catalog of rules into a compiled catalog, None if the catalog is invalid

```
Option<CompiledCatalog<T>> Compile<T>(RulesCatalog catalog) where T : new()
```

Parameters

`catalog` RulesCatalog

Returns

Option<CompiledCatalog<T>>

Type Parameters

`T`

Interface IRulesSetCompiler

Namespace: LogicEngine.Interfaces.Compilers

Assembly: LogicEngine.dll

```
public interface IRulesSetCompiler
```

Methods

Compile<T>(RulesSet)

Compiles a rule set into a compiled rule set, None if the rule set is invalid

```
Option<CompiledRulesSet<T>> Compile<T>(RulesSet set) where T : new()
```

Parameters

`set` RulesSet

Returns

Option<CompiledRulesSet<T>>

Type Parameters

`T`

Namespace LogicEngine.Internals

Enums

OperatorType

Enum OperatorType

Namespace: LogicEngine.Internals

Assembly: LogicEngine.dll

```
public enum OperatorType
```

Fields

`Contains = 11`

Item parameter contains specified constant

`ContainsKey = 15`

Item parameter dictionary property contains the given key

`ContainsValue = 17`

Item parameter dictionary property contains the given value

`Equal = 1`

Item parameter is equal to specified constant

`GreaterThan = 6`

Item parameter greater than specified constant

`GreaterThanOrEqual = 7`

Item parameter greater or equal than specified constant

`InnerContains = 29`

Item parameter array contains another item parameter

`InnerEqual = 23`

Item parameter equal to another item parameter

`InnerGreaterThan = 24`

Item parameter greater than another item parameter

`InnerGreaterThanOrEqual = 25`

Item parameter greater or equal than another item parameter

`InnerLessThan = 26`

Item parameter less than another item parameter

`InnerLessThanOrEqual = 27`

Item parameter less or equal than another item parameter

`InnerNotContains = 30`

Item parameter array does not contain another item parameter

`InnerNotEqual = 28`

Item parameter not equal to another item parameter

`InnerNotOverlaps = 32`

Item parameter array does not overlap another array parameter

`InnerOverlaps = 31`

Item parameter array overlaps another array parameter

`IsContained = 21`

Item parameter is contained into specified constant array

`IsNotContained = 22`

Item parameter is not contained into specified constant array

`KeyContainsValue = 19`

Item parameter dictionary has the given value for the specified key

`LessThan = 8`

Item parameter less than specified constant

`LessThanOrEqual = 9`

Item parameter less or equal than specified constant

`None = 0`

`NotContains = 12`

Item parameter not contains specified constant

`NotContainsKey = 16`

Item parameter dictionary property does not contain the given key

`NotContainsValue = 18`

Item parameter dictionary property does not contain the given value

`NotEqual = 10`

Item parameter not equal to specified constant

`NotKeyContainsValue = 20`

Item parameter dictionary has not the given value for the specified key

`NotOverlaps = 14`

Item parameter does not have intersections with specified enumerable constant

`Overlaps = 13`

Item parameter has intersection with specified enumerable constant

`StringContains = 4`

Item parameter string contains specified constant

`StringEndsWith = 3`

Item parameter string ends with specified constant

`StringRegexIsMatch = 5`

Item parameter string matches specified regex

`StringStartsWith = 2`

Item parameter string starts with specified constant

Namespace LogicEngine.Models

Classes

Rule

RulesCatalog

RulesSet

Class Rule

Namespace: LogicEngine.Models

Assembly: LogicEngine.dll

```
public record Rule : IEquatable<Rule>
```

Inheritance

object ← Rule

Implements

IEquatable<Rule>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Rule(string, OperatorType, string, string)

```
public Rule(string property, OperatorType @operator, string value, string code)
```

Parameters

`property` string

`operator` OperatorType

`value` string

`code` string

Properties

Code

Code to return if the rule is not satisfied

```
[DataMember(Name = "code")]
public string Code { get; init; }
```

Property Value

string

Operator

Operator to apply to the property

```
[DataMember(Name = "operator")]
public OperatorType Operator { get; init; }
```

Property Value

OperatorType

Property

Name of the property the rule is applied to

```
[DataMember(Name = "property")]
public string Property { get; init; }
```

Property Value

string

Value

Value to compare the property to

```
[DataMember(Name = "value")]
```

```
public string Value { get; init; }
```

Property Value

string

Methods

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

string

A string that represents the current object.

Class RulesCatalog

Namespace: LogicEngine.Models

Assembly: LogicEngine.dll

```
public record RulesCatalog : IEquatable<RulesCatalog>
```

Inheritance

object ← RulesCatalog

Implements

IEquatable<RulesCatalog>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesCatalog(IEnumerable<RulesSet>, string)

Creates a new rules catalog

```
public RulesCatalog(IEnumerable<RulesSet> rulesSets, string name)
```

Parameters

`rulesSets` IEnumerable<RulesSet>

`name` string

Properties

Name

```
public string Name { get; }
```

Property Value

string

RulesSets

Rules sets that make up the catalog

```
public IEnumerable<RulesSet> RulesSets { get; }
```

Property Value

IEnumerable<RulesSet>

Operators

operator +(RulesCatalog, RulesCatalog)

This represents the logical OR between two catalogs

```
public static RulesCatalog operator +(RulesCatalog catalog1, RulesCatalog catalog2)
```

Parameters

`catalog1` RulesCatalog

`catalog2` RulesCatalog

Returns

RulesCatalog

operator *(RulesCatalog, RulesCatalog)

This represents the logical AND between two catalogs

```
public static RulesCatalog operator *(RulesCatalog catalog1, RulesCatalog catalog2)
```

Parameters

`catalog1` RulesCatalog

`catalog2` RulesCatalog

Returns

RulesCatalog

Class RulesSet

Namespace: LogicEngine.Models

Assembly: LogicEngine.dll

```
public record RulesSet : IEquatable<RulesSet>
```

Inheritance

object ← RulesSet

Implements

IEquatable<RulesSet>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RulesSet(IEnumerable<Rule>, string)

Creates a new rule set

```
public RulesSet(IEnumerable<Rule> rules, string name)
```

Parameters

`rules` IEnumerable<Rule>

`name` string

Properties

Name

```
public string Name { get; }
```

Property Value

string

Rules

Rules that make up the set

```
[DataMember(Name = "rules")]
public IEnumerable<Rule> Rules { get; init; }
```

Property Value

IEnumerable<Rule>

Operators

operator *(RulesSet, RulesSet)

Combines two rule sets into one

```
public static RulesSet operator *(RulesSet set1, RulesSet set2)
```

Parameters

`set1` RulesSet

`set2` RulesSet

Returns

RulesSet