HeuristicLab

A Paradigm-Independent and Extensible Environment for Heuristic Optimization

Programming HeuristicLab

Basics

A. Scheibenpflug
Heuristic and Evolutionary Algorithms Laboratory (HEAL)
School of Informatics/Communications/Media, Campus Hagenberg
University of Applied Sciences Upper Austria

HEAL
Heuristic and Evolutionary Algorithms Laboratory

OBERÖSTERREICH

Prerequisites

- You should
  - know how to use HeuristicLab
  - have a basic understanding of what metaheuristics are
  - know how to write code
  - know C# or Java or similar languages
- This is not a user guide
- This is an overview
  - For details have a look at the source code

Introduction

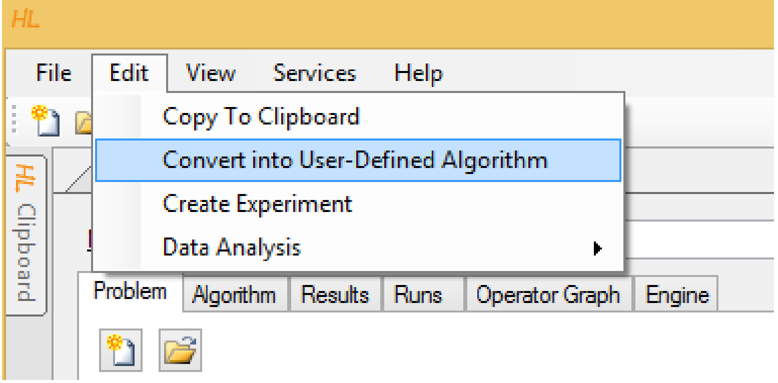
- HeuristicLab (HL) is quite a big project
- As of 3.3.12:
  - 5 VS solutions containing 173 projects
  - Lines of code: 670.526 + 890.638 (EXT) = 1.561.164 LOC
  - 368 unit tests
  - Quite a lot of feature branches in the SVN repository
- There are certain patterns/concepts that are used throughout all that code

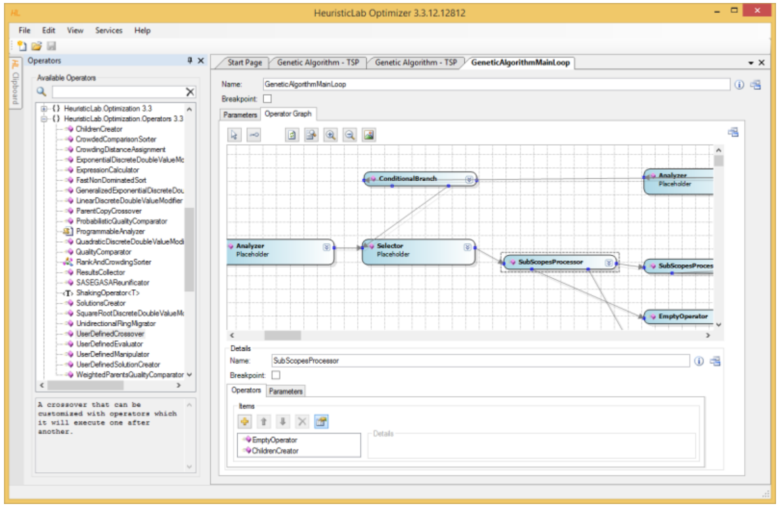
Extension Points

- HL can be extended in multiple ways
  - User-defined algorithm
  - User-defined problem
  - Programmable operators
  - Programmable problem
  - C# Script
  - Plugins

User-defined algorithm

- Start from an existing algorithm
- No programming skills required
- Useful for smaller modifications and prototyping
- Caution: Wiring is not active





```
A crossover that can be
customized with operators which
it will execute one after
another.
```

Name: GeneticAlgorithmMainLoop

Breakpoint: □

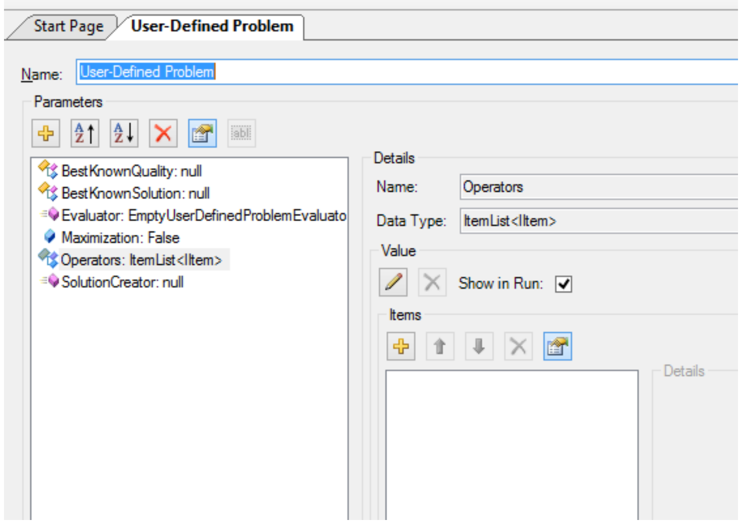
Name: SubScopesProcessor

Breakpoint: □

EmptyOperator
ChildrenCreator

User-defined problem

- Define a problem in the UI
- Use user-defined operators to fill the problems operator collection
- Usage of programmable operators also possible (e.g. programmable analyzer)
- No programming skills required



Start Page | User-Defined Problem

Name: User-Defined Problem

Parameters

- BestKnownQuality: null
- BestKnownSolution: null
- Evaluator: EmptyUserDefinedProblemEvaluato
- Maximization: False
- Operators: ItemList<IItem>
- SolutionCreator: null

Details

Name: Operators

Data Type: ItemList<IItem>

Value

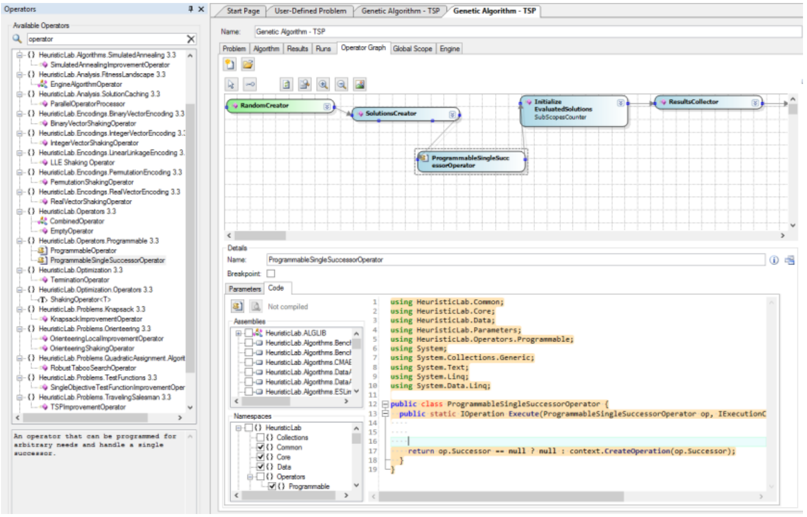
Show in Run: ☑

Items

Details

Programmable operators

- Used in user-defined algorithms and problems
- Used if there is
  - – no appropriate operator available
  - – creating a CombinedOperator is not desired
- Programming skills required



operator

- {} HeuristicLab.Algorithms.SimulatedAnnealing 3.3
  - SimulatedAnnealingImprovementOperator
- {} HeuristicLab.Analysis.FitnessLandscape 3.3
  - EngineAlgorithmOperator
- {} HeuristicLab.Analysis.SolutionCaching 3.3
  - ParallelOperatorProcessor
- {} HeuristicLab.Encodings.BinaryVectorEncoding 3.3
  - BinaryVectorShakingOperator
- {} HeuristicLab.Encodings.IntegerVectorEncoding 3.3
  - IntegerVectorShakingOperator
- {} HeuristicLab.Encodings.LinearLinkageEncoding 3.
  - LLE Shaking Operator
- {} HeuristicLab.Encodings.PermutationEncoding 3.3
  - PermutationShakingOperator
- {} HeuristicLab.Encodings.RealVectorEncoding 3.3
  - RealVectorShakingOperator
- {} HeuristicLab.Operators 3.3
  - CombinedOperator
  - EmptyOperator
- {} HeuristicLab.Operators.Programmable 3.3
  - ProgrammableOperator
  - ProgrammableSingleSuccessorOperator
- {} HeuristicLab.Optimization 3.3
  - TerminationOperator
- {} HeuristicLab.Optimization.Operators 3.3
  - ShakingOperator<T>
- {} HeuristicLab.Problems.Knapsack 3.3
  - KnapsackImprovementOperator
- {} HeuristicLab.Problems.Orienteering 3.3
  - OrienteeringLocalImprovementOperator
  - OrienteeringShakingOperator
- {} HeuristicLab.Problems.QuadraticAssignment.Algorit
  - RobustTabooSearchOperator
- {} HeuristicLab.Problems.TestFunctions 3.3
  - SingleObjectiveTestFunctionImprovementOper
- {} HeuristicLab.Problems.TravelingSalesman 3.3
  - TSPImprovementOperator

An operator that can be programmed for arbitrary needs and handle a single successor.

Name: Genetic Algorithm - TSP

Name: ProgrammableSingleSuccessorOperator

Breakpoint: □

Assemblies:

- □ HeuristicLab.ALGLIB
- □ HeuristicLab.Algorithms.Bencl
- □ HeuristicLab.Algorithms.Bencl
- □ HeuristicLab.Algorithms.CMAE
- □ HeuristicLab.Algorithms.DataA
- □ HeuristicLab.Algorithms.DataA
- □ HeuristicLab.Algorithms.ESLin

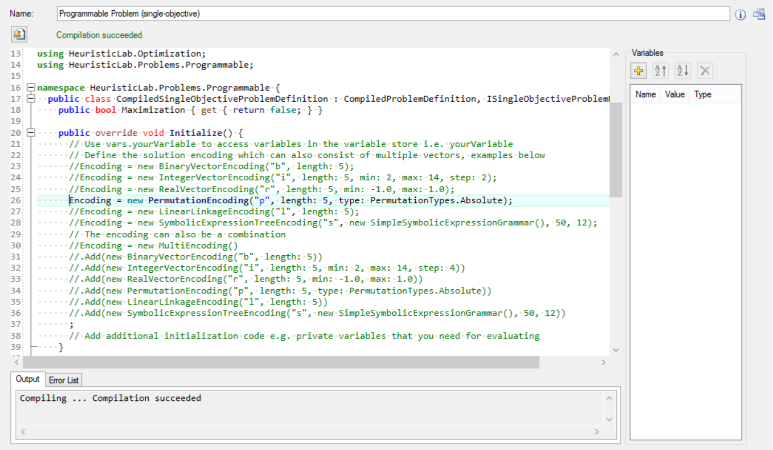
Namespaces:

- □ {} HeuristicLab
  - □ {} Collections
  - ☑ {} Common
  - ☑ {} Core
  - ☑ {} Data
  - □ {} Operators
    - ☑ {} Programmable

```
using HeuristicLab.Common;
using HeuristicLab.Core;
using HeuristicLab.Data;
using HeuristicLab.Parameters;
using HeuristicLab.Operators.Programmable;
using System;
using System.Collections.Generic;
using System.Text;
using System.Linq;
using System.Data.Linq;

public class ProgrammableSingleSuccessorOperator {
  public static IOperation Execute(ProgrammableSingleSuccessorOperator op, IExecutionC



    return op.Successor == null ? null : context.CreateOperation(op.Successor);
  }
}
```

Programmable problem

- Allows to define a problem in code in HeuristicLab
- Similar to user-defined problem, but with C#
- Only works if the encoding already exists
- Multi-encodings are possible
- Prototyping



Name: Programmable Problem (single-objective)

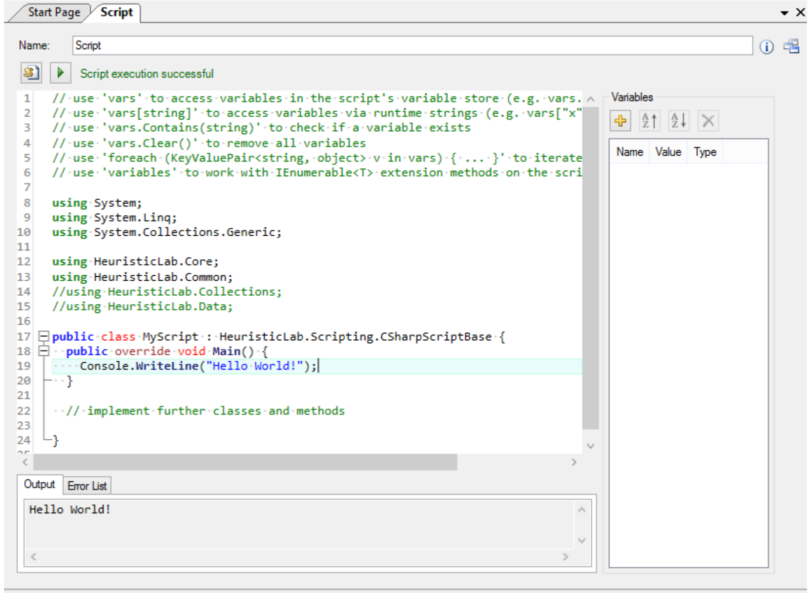
Compilation succeeded

```
using HeuristicLab.Optimization;
using HeuristicLab.Problems.Programmable;

namespace HeuristicLab.Problems.Programmable {
  public class CompiledSingleObjectiveProblemDefinition : CompiledProblemDefinition, ISingleObjectiveProbleml
    public bool Maximization { get { return false; } }

    public override void Initialize() {
      // Use vars.yourVariable to access variables in the variable store i.e. yourVariable
      // Define the solution encoding which can also consist of multiple vectors, examples below
      //Encoding = new BinaryVectorEncoding("b", length: 5);
      //Encoding = new IntegerVectorEncoding("i", length: 5, min: 2, max: 14, step: 2);
      //Encoding = new RealVectorEncoding("r", length: 5, min: -1.0, max: 1.0);
      Encoding = new PermutationEncoding("p", length: 5, type: PermutationTypes.Absolute);
      //Encoding = new LinearLinkageEncoding("l", length: 5);
      //Encoding = new SymbolicExpressionTreeEncoding("s", new SimpleSymbolicExpressionGrammar(), 50, 12);
      // The encoding can also be a combination
      //Encoding = new MultiEncoding()
      //.Add(new BinaryVectorEncoding("b", length: 5))
      //.Add(new IntegerVectorEncoding("i", length: 5, min: 2, max: 14, step: 4))
      //.Add(new RealVectorEncoding("r", length: 5, min: -1.0, max: 1.0))
      //.Add(new PermutationEncoding("p", length: 5, type: PermutationTypes.Absolute))
      //.Add(new LinearLinkageEncoding("l", length: 5))
      //.Add(new SymbolicExpressionTreeEncoding("s", new SimpleSymbolicExpressionGrammar(), 50, 12))
      ;
      // Add additional initialization code e.g. private variables that you need for evaluating
    }
```

```
Compiling ... Compilation succeeded
```

C# Script

- Write programs from within HeuristicLab
- Access to
  - HeuristicLab APIs
  - data types
  - views
- Mainly used for
  - creating complex experiments
  - analysis
  - pre- and post processing
- Prototyping



Name: Script

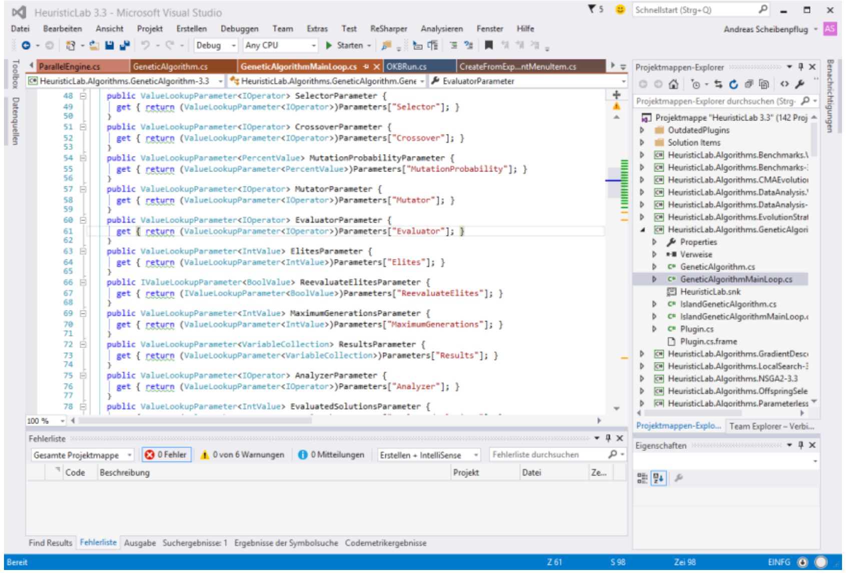
Script execution successful

```
// use 'vars' to access variables in the script's variable store (e.g. vars.
// use 'vars[string]' to access variables via runtime strings (e.g. vars["x"
// use 'vars.Contains(string)' to check if a variable exists
// use 'vars.Clear()' to remove all variables
// use 'foreach (KeyValuePair<string, object> v in vars) { ... }' to iterate
// use 'variables' to work with IEnumerable<T> extension methods on the scri

using System;
using System.Linq;
using System.Collections.Generic;

using HeuristicLab.Core;
using HeuristicLab.Common;
//using HeuristicLab.Collections;
//using HeuristicLab.Data;

public class MyScript : HeuristicLab.Scripting.CSharpScriptBase {
  public override void Main() {
    Console.WriteLine("Hello World!");
  }

  // implement further classes and methods

}
```

```
Hello World!
```

Plugins

- Are loaded into HL on startup
- Allow to add
  - algorithms
  - problems
  - operators
- Some features can only be added by creating plugins
  - data types
  - items
  - encodings
  - views
  - …
- Most universal way of adding functionality to HL



HeuristicLab 3.3 - Microsoft Visual Studio
Schnellstart (Strg+Q)
Andreas Scheibenpflug
Datei Bearbeiten Ansicht Projekt Erstellen Debuggen Team Extras Test ReSharper Analysieren Fenster Hilfe
Debug
Any CPU
Starten
ParallelEngine.cs
GeneticAlgorithm.cs
GeneticAlgorithmMainLoop.cs
OKBRun.cs
CreateFromExp...ntMenuItem.cs
HeuristicLab.Algorithms.GeneticAlgorithm-3.3
HeuristicLab.Algorithms.GeneticAlgorithm.Gene
EvaluatorParameter

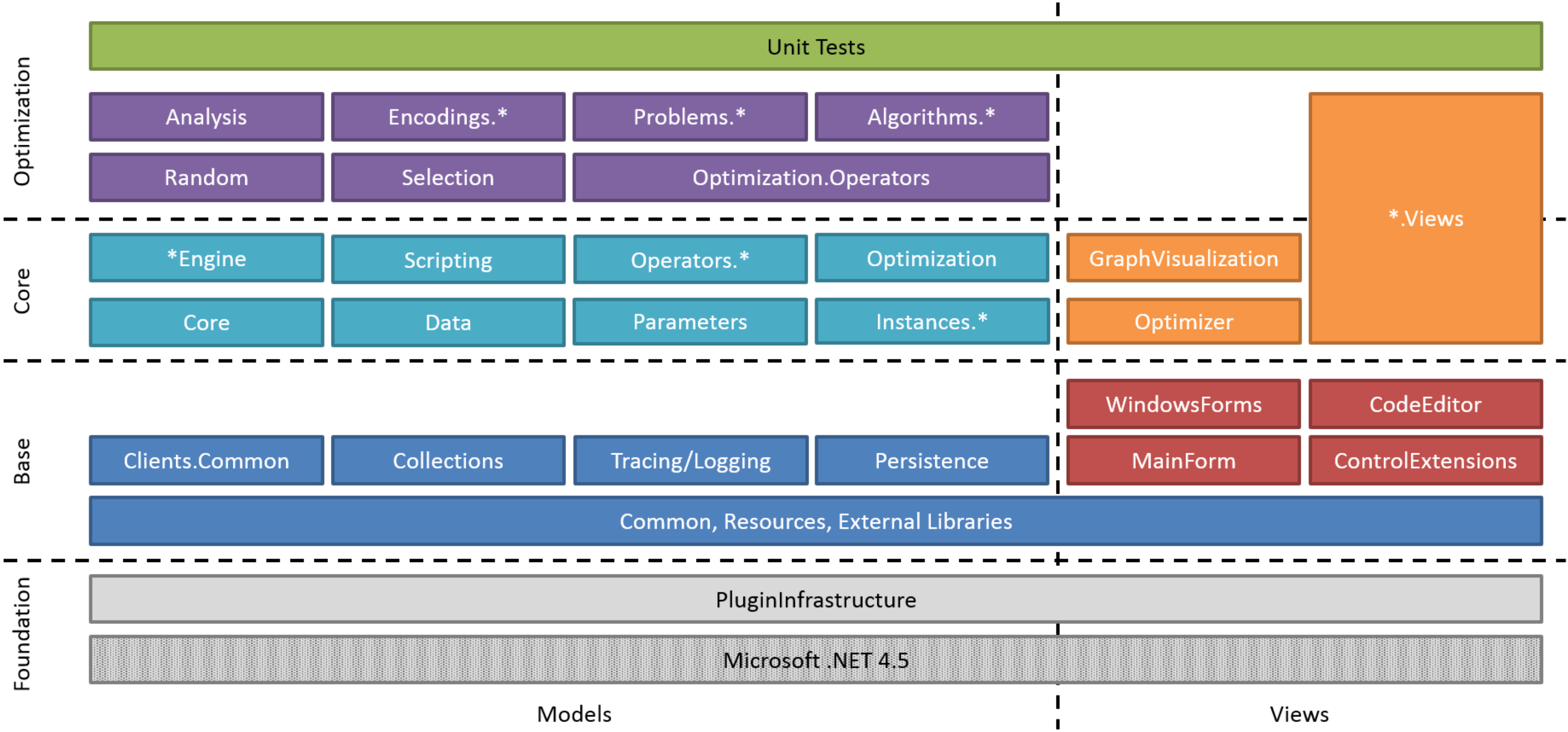
```
public ValueLookupParameter<IOperator> SelectorParameter {
  get { return (ValueLookupParameter<IOperator>)Parameters["Selector"]; }
}
public ValueLookupParameter<IOperator> CrossoverParameter {
  get { return (ValueLookupParameter<IOperator>)Parameters["Crossover"]; }
}
public ValueLookupParameter<PercentValue> MutationProbabilityParameter {
  get { return (ValueLookupParameter<PercentValue>)Parameters["MutationProbability"]; }
}
public ValueLookupParameter<IOperator> MutatorParameter {
  get { return (ValueLookupParameter<IOperator>)Parameters["Mutator"]; }
}
public ValueLookupParameter<IOperator> EvaluatorParameter {
  get { return (ValueLookupParameter<IOperator>)Parameters["Evaluator"]; }
}
public ValueLookupParameter<IntValue> ElitesParameter {
  get { return (ValueLookupParameter<IntValue>)Parameters["Elites"]; }
}
public IValueLookupParameter<BoolValue> ReevaluateElitesParameter {
  get { return (IValueLookupParameter<BoolValue>)Parameters["ReevaluateElites"]; }
}
public ValueLookupParameter<IntValue> MaximumGenerationsParameter {
  get { return (ValueLookupParameter<IntValue>)Parameters["MaximumGenerations"]; }
}
public ValueLookupParameter<VariableCollection> ResultsParameter {
  get { return (ValueLookupParameter<VariableCollection>)Parameters["Results"]; }
}
public ValueLookupParameter<IOperator> AnalyzerParameter {
  get { return (ValueLookupParameter<IOperator>)Parameters["Analyzer"]; }
}
public ValueLookupParameter<IntValue> EvaluatedSolutionsParameter {
```

100 %
Fehlerliste
Gesamte Projektmappe
0 Fehler
0 von 6 Warnungen
0 Mitteilungen
Erstellen + IntelliSense
Fehlerliste durchsuchen
Code
Beschreibung
Projekt
Datei
Ze...
Find Results
Fehlerliste
Ausgabe
Suchergebnisse: 1
Ergebnisse der Symbolsuche
Codemetrikergebnisse
Projektmappen-Explorer
Projektmappen-Explorer durchsuchen (Strg-
Projektmappe "HeuristicLab 3.3" (142 Proj
OutdatedPlugins
Solution Items
HeuristicLab.Algorithms.Benchmarks.\
HeuristicLab.Algorithms.Benchmarks-
HeuristicLab.Algorithms.CMAEvolutio
HeuristicLab.Algorithms.DataAnalysis.
HeuristicLab.Algorithms.DataAnalysis-
HeuristicLab.Algorithms.EvolutionStrat
HeuristicLab.Algorithms.GeneticAlgori
Properties
Verweise
GeneticAlgorithm.cs
GeneticAlgorithmMainLoop.cs
HeuristicLab.snk
IslandGeneticAlgorithm.cs
IslandGeneticAlgorithmMainLoop.c
Plugin.cs
Plugin.cs.frame
HeuristicLab.Algorithms.GradientDesc
HeuristicLab.Algorithms.LocalSearch-
HeuristicLab.Algorithms.NSGA2-3.3
HeuristicLab.Algorithms.OffspringSele
HeuristicLab.Algorithms.Parameterless
Projektmappen-Explo...
Team Explorer – Verbi...
Eigenschaften
Benachrichtigungen
Toolbox
Datenquellen
Bereit
Z 61
S 98
Zei 98
EINFG

Overview

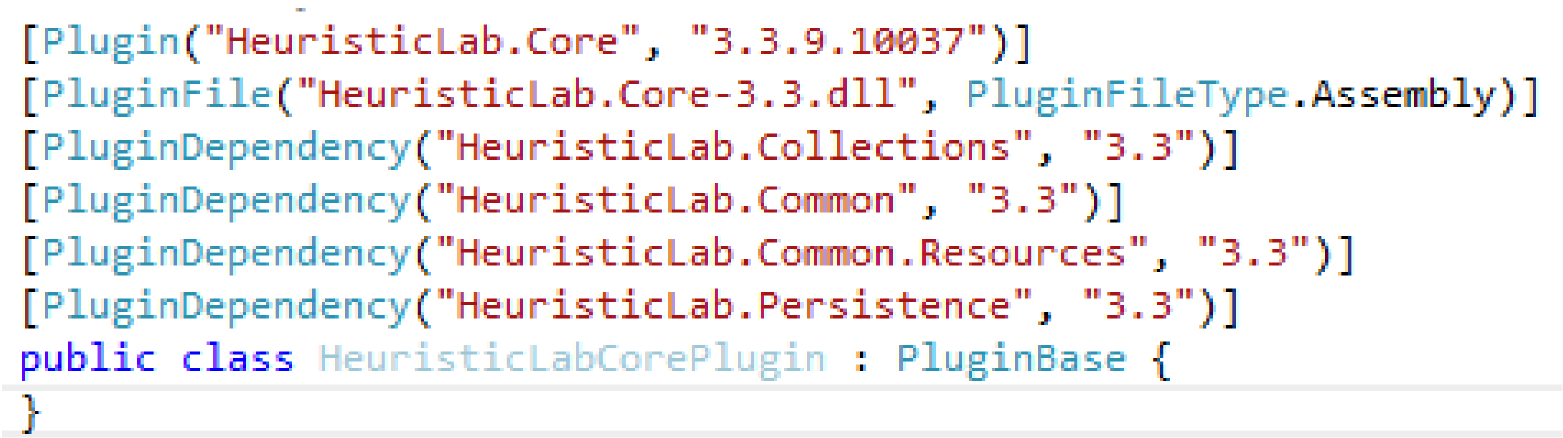
- Plugins
- HL Object Model
- Deep Cloning
- Persistence
- Items
- HL Data Types
- HL Collections
- Content and Views
- ViewHost

Where are we?


Optimization
Core
Base
Foundation
Unit Tests
Analysis
Encodings.*
Problems.*
Algorithms.*
Random
Selection
Optimization.Operators
*.Views
*Engine
Scripting
Operators.*
Optimization
GraphVisualization
Core
Data
Parameters
Instances.*
Optimizer
WindowsForms
CodeEditor
Clients.Common
Collections
Tracing/Logging
Persistence
MainForm
ControlExtensions
Common, Resources, External Libraries
PluginInfrastructure
Microsoft .NET 4.5
Models
Views

Plugins

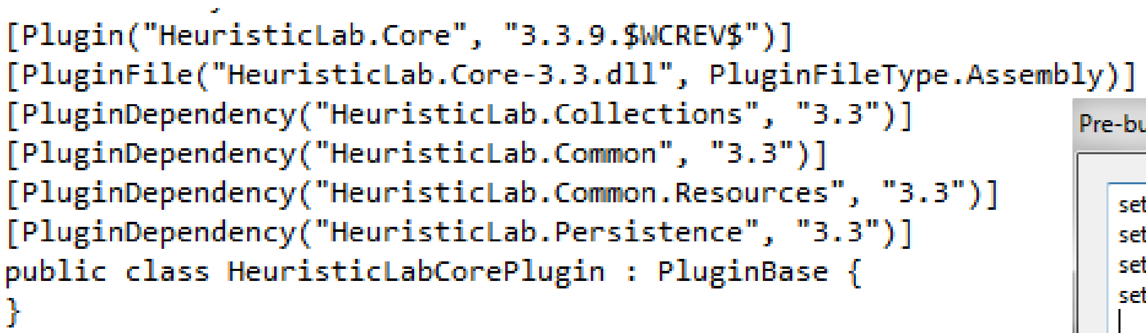
- Every plugin needs to contain a class that inherits `PluginBase`
- If an assembly contains such a class, it is a plugin and loaded by HeuristicLab



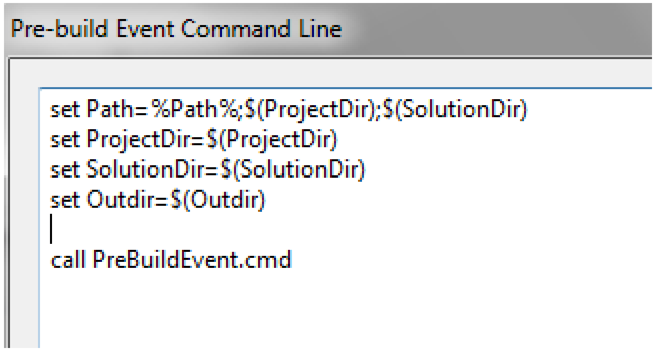
```
[Plugin("HeuristicLab.Core", "3.3.9.10037")]
[PluginFile("HeuristicLab.Core-3.3.dll", PluginFileType.Assembly)]
[PluginDependency("HeuristicLab.Collections", "3.3")]
[PluginDependency("HeuristicLab.Common", "3.3")]
[PluginDependency("HeuristicLab.Common.Resources", "3.3")]
[PluginDependency("HeuristicLab.Persistence", "3.3")]
public class HeuristicLabCorePlugin : PluginBase {
}
```

Plugins

- `PluginDependency` must reflect references
- Plugin Infrastructure does not have to be included as it is always needed
- We normally use SubWCRev for version information



```
[Plugin("HeuristicLab.Core", "3.3.9.$WCREV$")]
[PluginFile("HeuristicLab.Core-3.3.dll", PluginFileType.Assembly)]
[PluginDependency("HeuristicLab.Collections", "3.3")]
[PluginDependency("HeuristicLab.Common", "3.3")]
[PluginDependency("HeuristicLab.Common.Resources", "3.3")]
[PluginDependency("HeuristicLab.Persistence", "3.3")]
public class HeuristicLabCorePlugin : PluginBase {
}
```

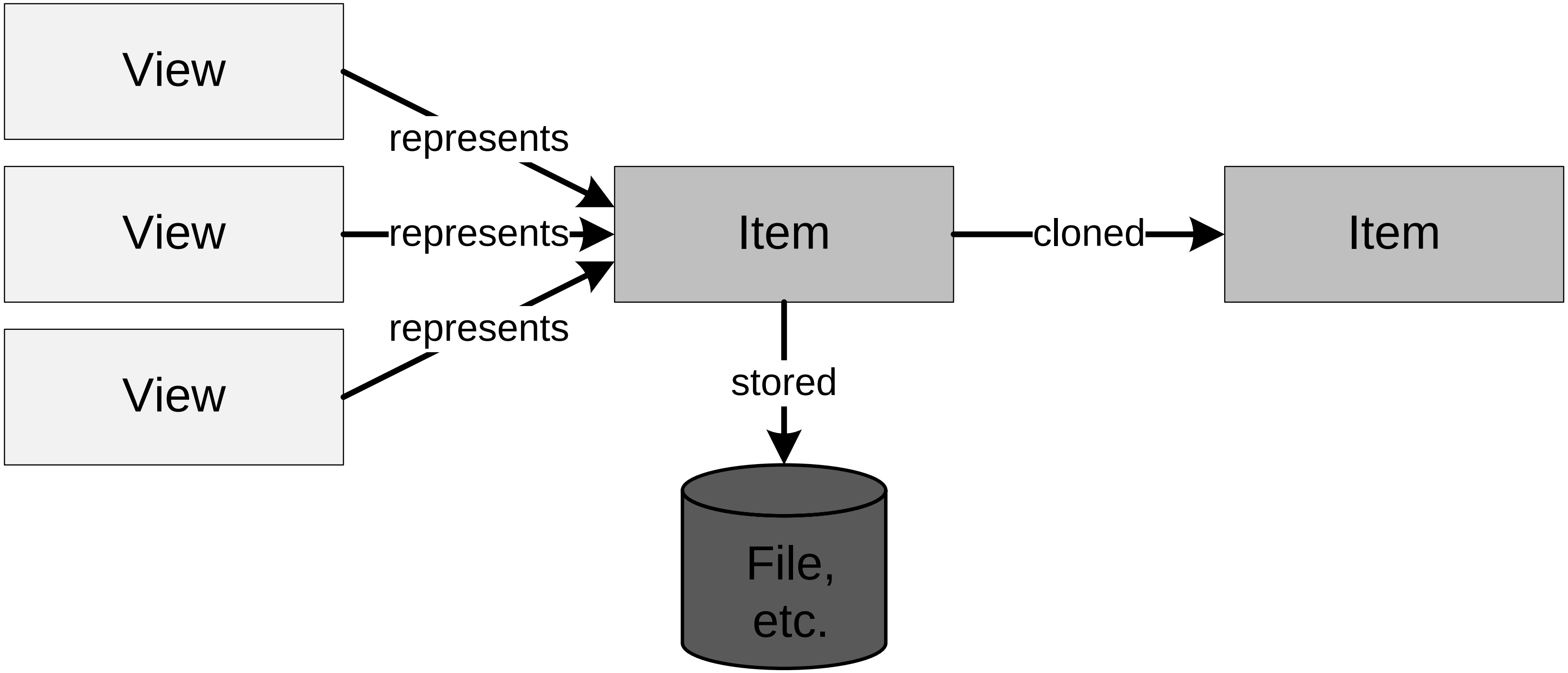


```
set Path=%Path%;$(ProjectDir);$(SolutionDir)
set ProjectDir=$(ProjectDir)
set SolutionDir=$(SolutionDir)
set Outdir=$(Outdir)

call PreBuildEvent.cmd
```

Some additional remarks

- Plugins are signed with the HeuristicLab key
- Every plugin builds to `sources\bin` (output path of project should be "`..\..\bin\`" for all configurations adhering to standard HL folder structure)
- Default namespace and assembly name should/must match plugin description
- There should be x86, x64, Any CPU Debug and Release configurations
- "Copy Local" should be false for all Project/File references

HL Object Model

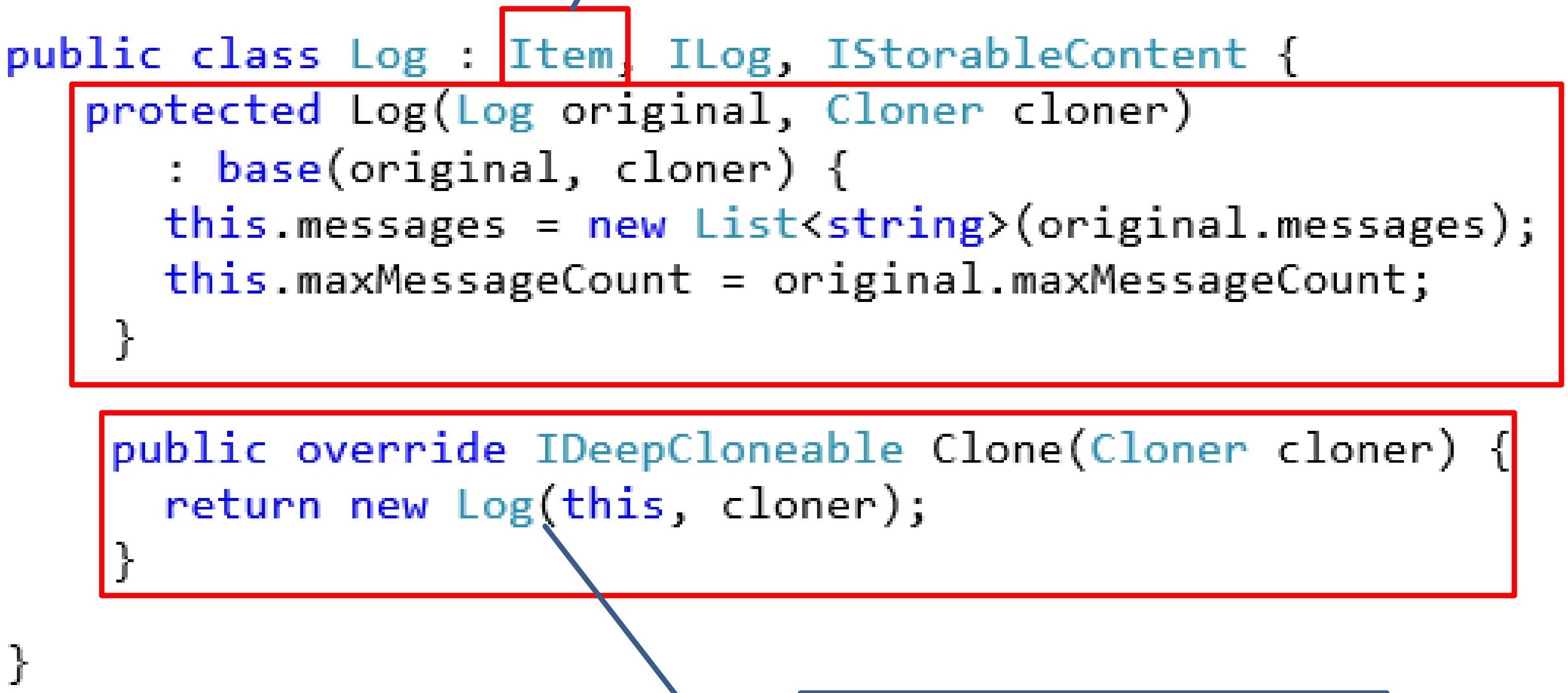


Deep Cloning

- Objects in HeuristicLab that store data and may be displayed in views/collection views should be deep cloneable
- UI allows “copying” of these objects
- Inherit from either `IDeepCloneable` or Item
- Implement interface and cloning constructor
- Actual cloning happens in the cloning constructor

Deep Cloning

Item implements IDeepCloneable



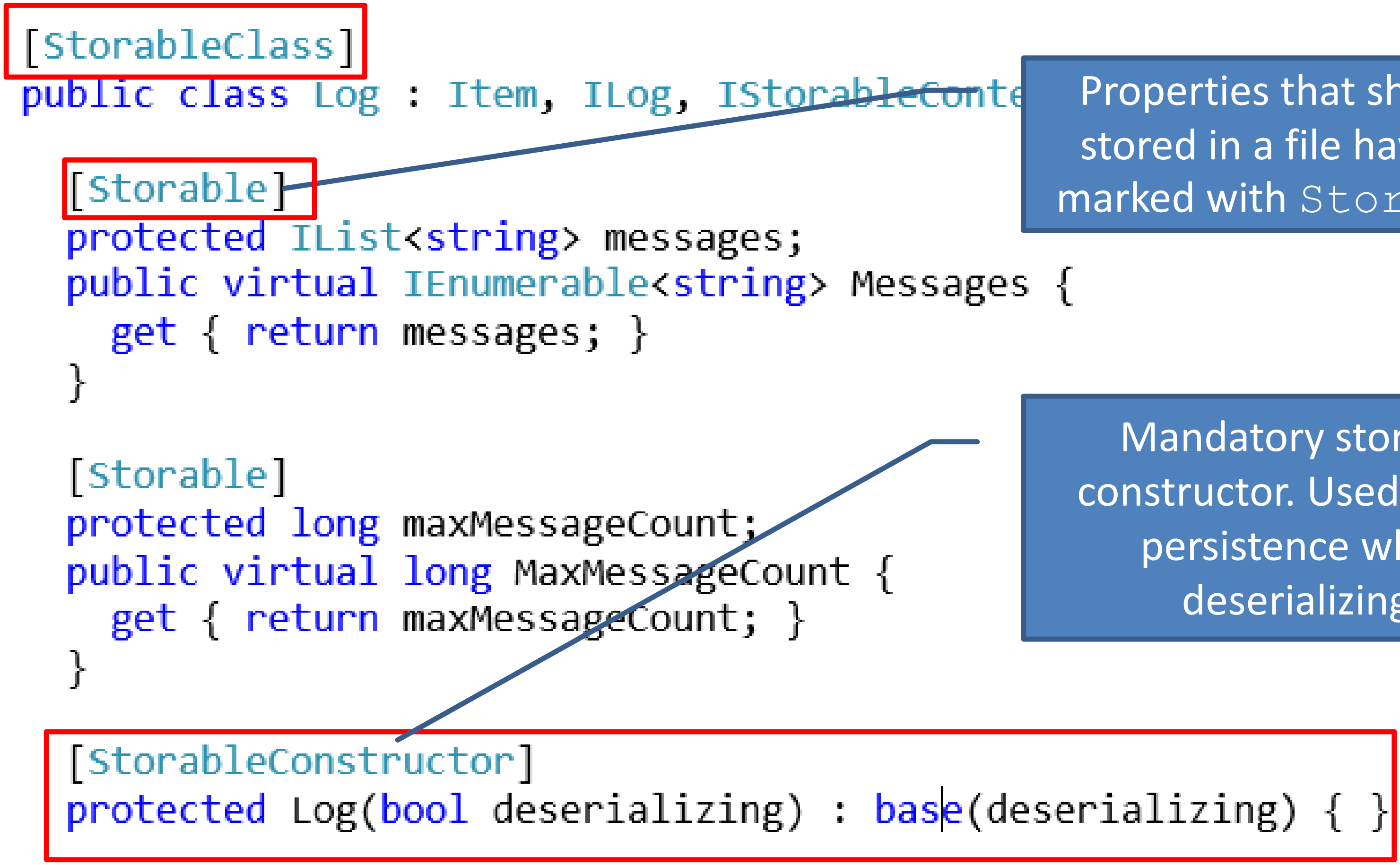
```
public class Log : Item, ILog, IStorableContent {
  protected Log(Log original, Cloner cloner)
     : base(original, cloner) {
     this.messages = new List<string>(original.messages);
     this.maxMessageCount = original.maxMessageCount;
   }

   public override IDeepCloneable Clone(Cloner cloner) {
     return new Log(this, cloner);
   }

}
```

Call cloning constructor which implements the cloning

Persistence

- HL provides it‘s own serialization mechanism
- A class that should be serializable has to be marked with the `StorableClass[]` attribute
- Properties that should be serialized have to be marked with the `Storable[]` attribute
- Storable constructor has to be implemented
- Optional: Define hooks with attribute `StorableHook[]` to react on loading/saving events
- Implement `IStorableContent` to signal that this is a root object

Persistence



```
[StorableClass]
public class Log : Item, ILog, IStorableConte

  [Storable]
  protected IList<string> messages;
  public virtual IEnumerable<string> Messages {
    get { return messages; }
  }

  [Storable]
  protected long maxMessageCount;
  public virtual long MaxMessageCount {
    get { return maxMessageCount; }
  }

  [StorableConstructor]
  protected Log(bool deserializing) : base(deserializing) { }
```

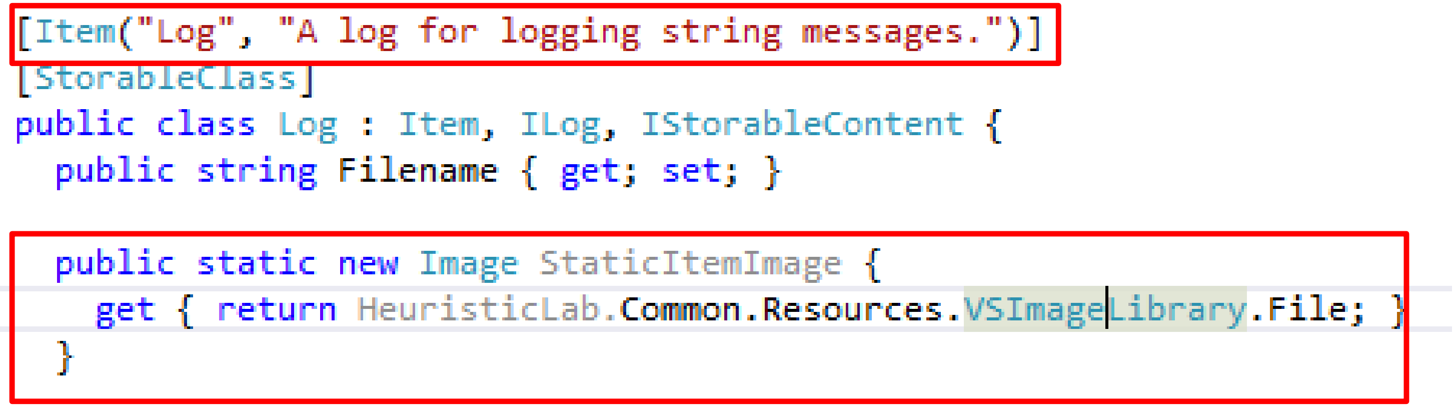
Properties that should be stored in a file have to be marked with `Storable[]`

Mandatory storable constructor. Used by the persistence when deserializing.

Items

- Items have
  - a name
  - a description
  - an icon
  - `ToStringChanged` and `ItemImageChanged` events
- All Items are DeepCloneables and Storable
- Items are marked as `IContent` to allow displaying in views
- Use `Item[]` attribute to set name and description

Items



```
[Item("Log", "A log for logging string messages.")]
[StorableClass]
public class Log : Item, ILog, IStorableContent {
  public string Filename { get; set; }

  public static new Image StaticItemImage {
    get { return HeuristicLab.Common.Resources.VSImageLibrary.File; }
  }
```

HL Data Types

- Located in `HeuristicLab.Data` (and corresponding views in `Data.Views`)
- Wrap standard .NET data types and provide functionality necessary for UIs:
  - `ValueChanged` event
  - Parsing of strings
  - Validation
- Data types include
  - `IntValue`, `DoubleValue`, `PercentValue`, `StringValue`,…
  - Ranges, Arrays, Matrices

Collections

- Located in `HeuristicLab.Collections/Core` (and `Core.Views` for the corresponding views)
- Same as with data types, provide UI friendly wrappers for .NET collections (e.g., additional events)
- There are Lists, Arrays, Sets, Dictionaries and read-only collections
- Most are designed for Items

Data Types and Collections

```
results.Add(new Result("MWIPS", new IntValue(intRating / 1000)));
```

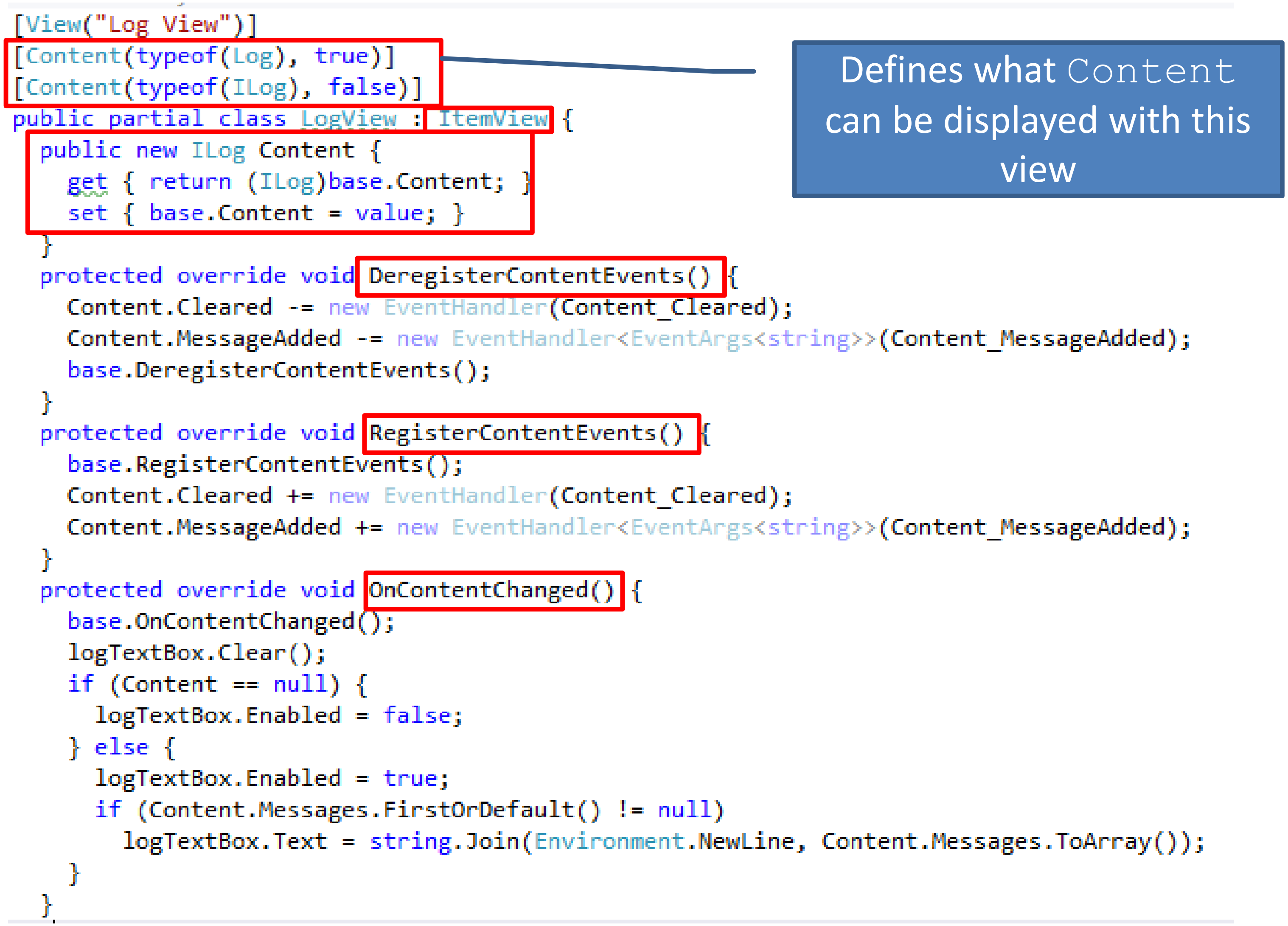
```
DoubleValue doubleValue  = new DoubleValue();
doubleValue.Value = resultValue.Value.Average();
```

```
[Storable]
private ItemList<ICovarianceFunction> terms;
public CovarianceSum()
: base() {
  this.terms = new ItemList<ICovarianceFunction>();
}

terms.Select(t => t.GetNumberOfParameters(numberOfVariables)).Sum();
```

Content and Views

- HL provides views for all data types, collections and much more (including input validation and updates)
- Views display (and manipulate) `Content`
- Use `Content[]` attribute to define the type of `Content` a `View` can display
- Inherit `UserControl` from `AsynchronousContentView` or `ItemView`
- `Content` is set by HeuristicLab or manually
- React on events (e.g., `OnContentChanged`, `(De)RegisterContentEvents`, …)

Content and Views



```
[View("Log View")]
[Content(typeof(Log), true)]
[Content(typeof(ILog), false)]
public partial class LogView : ItemView {
  public new ILog Content {
    get { return (ILog)base.Content; }
    set { base.Content = value; }
  }
  protected override void DeregisterContentEvents() {
    Content.Cleared -= new EventHandler(Content_Cleared);
    Content.MessageAdded -= new EventHandler<EventArgs<string>>(Content_MessageAdded);
    base.DeregisterContentEvents();
  }
  protected override void RegisterContentEvents() {
    base.RegisterContentEvents();
    Content.Cleared += new EventHandler(Content_Cleared);
    Content.MessageAdded += new EventHandler<EventArgs<string>>(Content_MessageAdded);
  }
  protected override void OnContentChanged() {
    base.OnContentChanged();
    logTextBox.Clear();
    if (Content == null) {
      logTextBox.Enabled = false;
    } else {
      logTextBox.Enabled = true;
      if (Content.Messages.FirstOrDefault() != null)
        logTextBox.Text = string.Join(Environment.NewLine, Content.Messages.ToArray());
    }
  }
```

Defines what `Content` can be displayed with this view

Displaying Content

- Manually:

```
Log log = new Log();
LogView logview = new LogView();
logview.Content = log;
```

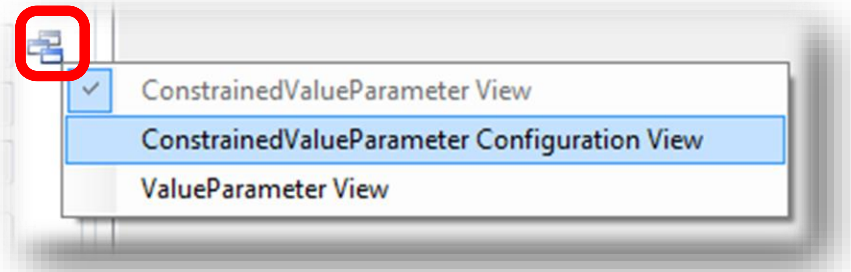
- In an own tab using discovery:

```
MainFormManager.MainForm.ShowContent(log);
```

- Using a `ViewHost`

ViewHost

- `ViewHost` is a special `ContentView` that changes it‘s appearance based on the type of `Content`
- `Content[]` attribute marks a view for a certain content type
- `ViewHost` looks up the view based on the `Content` type and uses it to display the `Content`
- Useful for views that can contain different `Content` types or collection views


✓ ConstrainedValueParameter View
ConstrainedValueParameter Configuration View
ValueParameter View

Useful Links

http://dev.heuristiclab.com/trac.fcgi/wiki/Documentation

http://dev.heuristiclab.com/trac.fcgi/wiki/Research

heuristiclab@googlegroups.com

http://www.youtube.com/heuristiclab