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

Josef Ressel-Zentrum
HEUREKA!

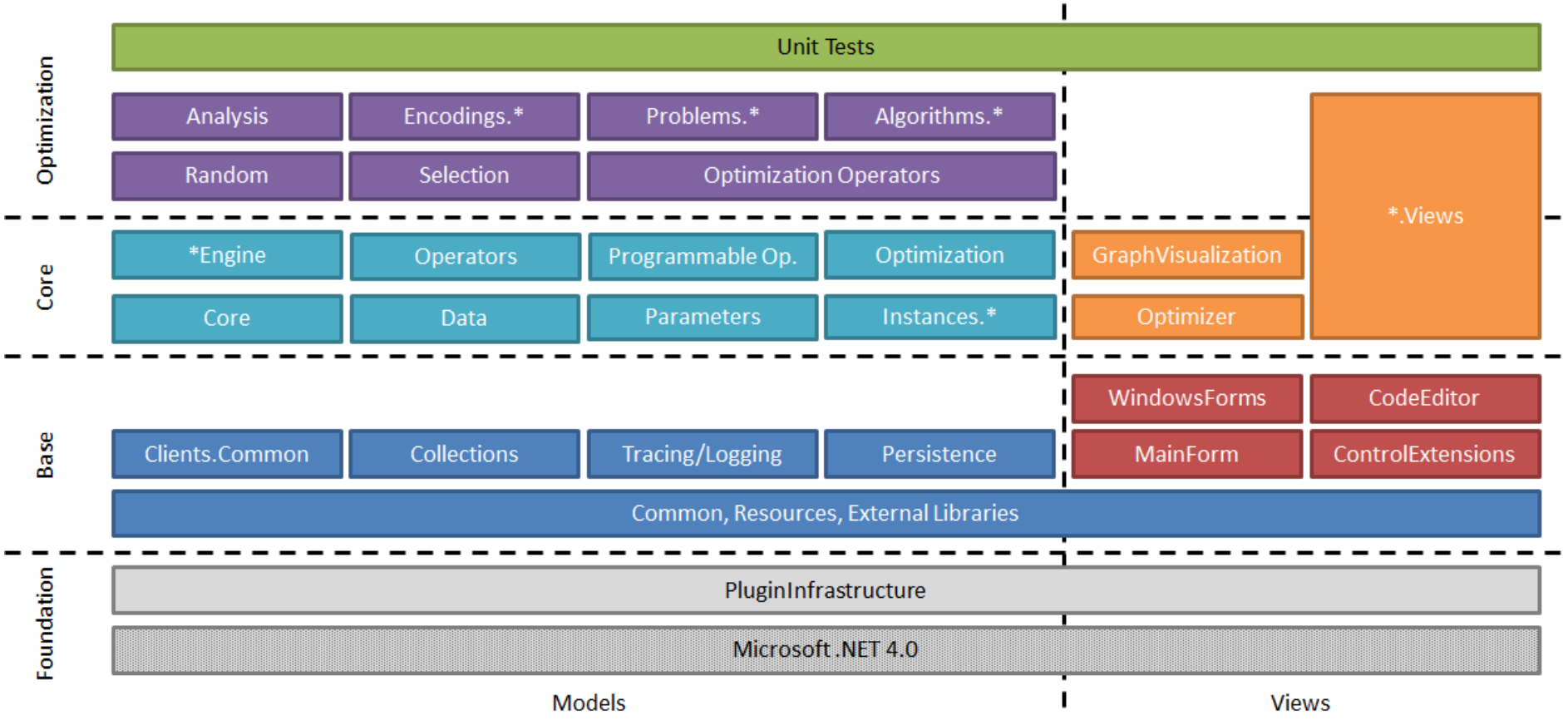
Overview

HeuristicLab

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

Where are we?

HeuristicLab



Plugins

HeuristicLab

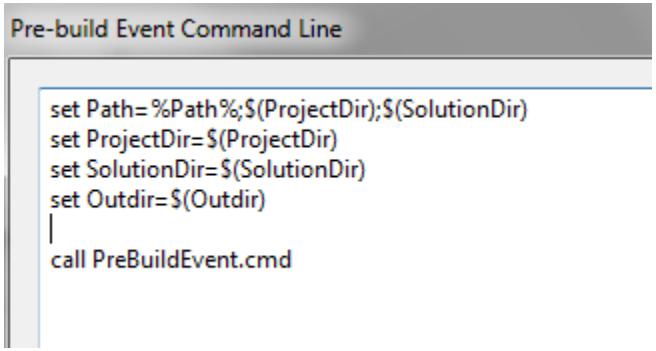
- Every plugin needs to contain a class that inherits PluginBase
- If an assembly contains such a class, it is a plugin and loaded by HeuristicLab

```
[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
- PluginInfrastructure does not have to be included as it is always needed
- We normally use SubWCRev for version information

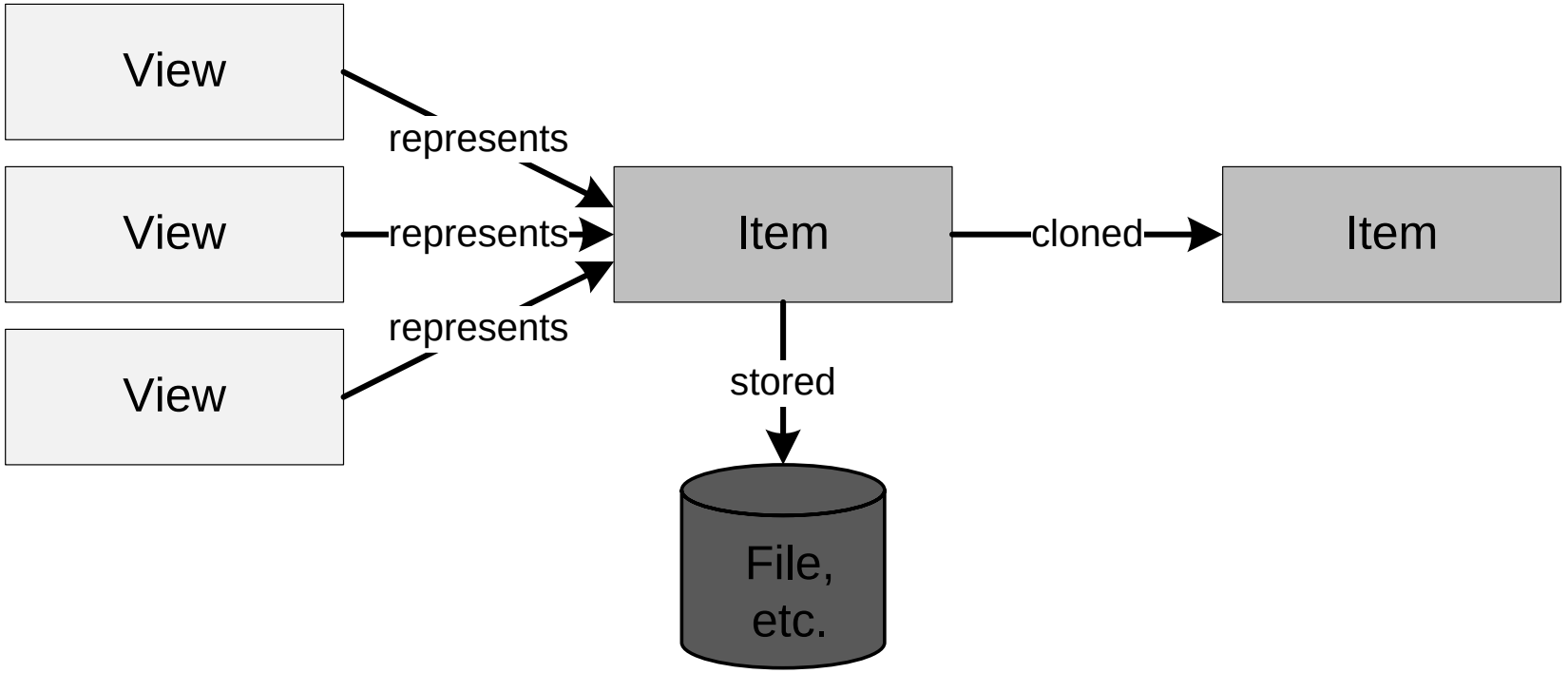
```
[Plugin("HeuristicLab.Core", "3.3.9.$WCREV$")]
[PluginFile("HeuristicLab.Core-3.3.dll", PluginFileType.Assembly)]
[PluginDependency("HeuristicLab.Collections", "3.3")]
[PluginDependency("HeuristicLab.Common", "3.3")]
[PluginDependency("HeuristicLab.Common.Resources", "3.3")]
[PluginDependency("HeuristicLab.Persistence",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



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

HeuristicLab

- 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

HeuristicLab

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
- StorableConstructor 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

HeuristicLab

```
[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
- Most Items are marked as IContent to allow displaying in views
- Use [Item] attribute to set name and description

Items

HeuristicLab

```
[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

HeuristicLab

- 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
- DataTypes 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

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

HeuristicLab

```
[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

HeuristicLab

- 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

HeuristicLab

- 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

Useful Links

HeuristicLab

http://dev.heuristiclab.com/trac/hl/core/wiki/UsersHowtos

http://dev.heuristiclab.com/trac/hl/core/wiki/Publications

heuristiclab@googlegroups.com

http://www.youtube.com/heuristiclab