New Validation Framework

Gary Karasiuk

2007-12-11  Rational software
Background

- The current validation framework has a number of known problems and limitations.
- I have been thinking about adding a new validation framework.
- This presentation starts to introduce that new framework.
Approach

- Old validators – work exactly like they used to
  - ✔ They still plug into the old framework
  - ✔ They are still run by the old framework
  - ✔ Good migration story
    - ✷ Can’t blame new framework if things don’t work
    - ✷ Migrate at your own pace – possibly never
    - ✷ Fine for low volume validators and third party validators

- New Validators
  - ✔ Designed to minimize what validator owners need to do
  - ✔ Use a new extension point
  - ✔ Implementation detail
    - ✷ Wrap an old validator
    - ✷ Supply a brand new, better validator
Validators get to Pick the Defaults

```xml
<extension
  id="html"
  name="$HTML_Syntax_Validator.name"
  point="org.eclipse.wst.validation.validatorV2">
  <validator
    build="true"
    class="org.eclipse.wst.html.internal.validation.HTMLValidator"
    manual="true">
    <include>
      <rules>
        <fileext caseSensitive="false" ext="html"></fileext>
        <fileext caseSensitive="false" ext="htm"></fileext>
        <fileext caseSensitive="false" ext="xhtml"></fileext>
        <fileext caseSensitive="false" ext="html"></fileext>
        <contentType
          id="org.eclipse.wst.html.core.htmlsource"/>
      </rules>
    </include>
  </validator>
</extension>
```
But, Users get the ultimate say …
New Validation UI

- Top level UI almost identical to existing UI
  - The level where you turn on or off validation
- New Validators will have additional configuration options on the “Settings” page:
  - File extensions
  - Natures
  - Facets
  - Content Type
  - Folder names
Users get more control over Filters
Adding Filters

Folder or File Name

File or folder: Browse File... Browse Folder...

Is the comparison case sensitive?

< Back Next > Erase Cancel

Project Nature


SIP Nature - com.ibm.siptools.SIPNature
- org.eclipse.wst.common.project.facet.core.nature
WRD Auto App Install Nature - com.ibm.ws.rapiddeploy.core.WRDApplInstallNature

< Back Next > Finish Cancel
API

Validation Framework

See:
- Description

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDependencyIndex</td>
<td>This service is used to specify the resources that a particular resource depends on.</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractValidator</td>
<td>The class that all Validators that wish to use version two of the validation framework must subclass.</td>
</tr>
<tr>
<td>ReporterHelper</td>
<td>This is a temporary class to ease the transition from the previous validation framework.</td>
</tr>
<tr>
<td>ValidationFramework</td>
<td>The central class of the Validation Framework.</td>
</tr>
<tr>
<td>ValidationResult</td>
<td>The result of running a validate operation.</td>
</tr>
<tr>
<td>ValidationResults</td>
<td>The combined results of validating multiple resources.</td>
</tr>
<tr>
<td>ValidationState</td>
<td>Keep track of arbitrary validation data, during the course of a validation.</td>
</tr>
<tr>
<td>Validator</td>
<td>Represents a validator.</td>
</tr>
<tr>
<td>Validator.V1</td>
<td>A validator that uses version 1 of the validation framework.</td>
</tr>
<tr>
<td>Validator.V2</td>
<td>A validator that uses version 2 of the validation framework.</td>
</tr>
<tr>
<td>ValidatorMessage</td>
<td>This class provides a way for a validator to return messages, that are easily converted into IMarkers.</td>
</tr>
</tbody>
</table>
Class AbstractValidator

java.lang.Object
  org.eclipse.wst.validation.AbstractValidator

public abstract class AbstractValidator
extends java.lang.Object

The class that all Validators that wish to use version two of the validation framework must subclass.

Author:
karasiuk

Constructor Summary

AbstractValidator()

Method Summary

void clean(org.eclipse.core.resources.IProject project, ValidationState state, org.eclipse.core.runtime.IProgressMonitor monitor)
The project is being cleaned, this method gives the validator a chance to do any special cleanup.

clean()

java.lang.String getDependencyId()
The validator is allowed to assert dependencies between various resources.

dependencyId()

abstract void validate(org.eclipse.core.resources.IResource resource, int kind, ValidationState state, org.eclipse.core.runtime.IProgressMonitor monitor)
Validate the resource.

validate()

void validationFinishing(org.eclipse.core.resources.IProject project, ValidationState state, org.eclipse.core.runtime.IProgressMonitor monitor)
This method will be called when validation is complete.

validationFinishing()

void validationStarting(org.eclipse.core.resources.IProject project, ValidationState state, org.eclipse.core.runtime.IProgressMonitor monitor)
This method will be called before any validation takes place.

validationStarting()
## Constructor Summary

**ValidationResult()**

## Method Summary

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add(ValidatorMessage message)</td>
<td>This is an optional method, that a validator can use to return error messages.</td>
</tr>
<tr>
<td>void</td>
<td>getDependsOn()</td>
<td></td>
</tr>
<tr>
<td>org.eclipse.core.resources.IResource[]</td>
<td>getMessages()</td>
<td>Answer any validation messages that were added by the validator.</td>
</tr>
<tr>
<td>int</td>
<td>getSeverityError()</td>
<td>Answer the number of error messages that were generated as part of this validation operation.</td>
</tr>
<tr>
<td>int</td>
<td>getSeverityInfo()</td>
<td>Answer the number of informational messages that were generated as part of this validation operation.</td>
</tr>
<tr>
<td>int</td>
<td>getSeverityWarning()</td>
<td>Answer the number of warning messages that were generated as part of this validation operation.</td>
</tr>
<tr>
<td>org.eclipse.core.resources.IResource[]</td>
<td>getValidated()</td>
<td>Answer all the resources that were validated as a side-effect of validating the main resource.</td>
</tr>
<tr>
<td>int</td>
<td>incrementError(int errors)</td>
<td>Increment the number of error messages that were generated as part of this validation operation.</td>
</tr>
<tr>
<td>int</td>
<td>incrementInfo(int info)</td>
<td>Increment the number of informational messages that were generated as part of this validation operation.</td>
</tr>
<tr>
<td>int</td>
<td>incrementWarning(int warnings)</td>
<td>Increment the number of warning messages that were generated as part of this validation operation.</td>
</tr>
<tr>
<td>boolean</td>
<td>isCanceled()</td>
<td>Was the operation canceled before it completed? For example if the validation is being run through the UI, the end user can cancel the operation through the progress monitor.</td>
</tr>
<tr>
<td>void</td>
<td>mergeResults(ValidationResult result)</td>
<td>Merge the message counts and messages from an individual validator into this result.</td>
</tr>
<tr>
<td>void</td>
<td>setCanceled(boolean canceled)</td>
<td>Indicate if the operation was canceled.</td>
</tr>
<tr>
<td>void</td>
<td>setDependsOn(org.eclipse.core.resources.IResource[] dependsOn)</td>
<td></td>
</tr>
</tbody>
</table>
Case Study

What did it take to convert the HTML Syntax Validator?

Not much

✔ Two files changed
  ✗ Plugin.xml – to use the new extension point
  ✗ HTMLValidator
Before

```java
public class HTMLValidator implements IVValidatorJob, ISourceValidator, IExecutableExtension {
    // Existing Java code...
}
```

After

```java
public class HTMLValidator extends AbstractValidator
    implements IVValidator, ISourceValidator, IExecutableExtension {
    // Existing Java code...

    public ValidationResult validate(IResource resource, int kind, ValidationState state, IProgressMonitor monitor) {
        ValidationResult vr = new ValidationResult();
        if (resource == null || !(resource instanceof IFile)) return vr;

        ReporterHelper reporter = new ReporterHelper(monitor);
        validateFile(null, reporter, (IFile) resource);
        reporter.updateResult(vr);
        return vr;
    }
}
```
Some methods to note:

**setDependsOn**

```java
public void setDependsOn(org.eclipse.core.resources.IResource[] dependsOn)
```

Update the resources that the validated resource depends on. This can be left null. For example, an XML file may depend on a XSD in order to know if it is valid or not. It would pass back the XSD file.

**Parameters:**

- `dependsOn` - if this is null then the dependency information is not updated. To remove the dependency information, an empty array needs to be supplied. A non-null parameter, replaces all the dependency information for this resource, for this validator.

**getValidated**

```java
public org.eclipse.core.resources.IResource[] getValidated()
```

Answer all the resources that were validated as a side-effect of validating the main resource.

**setValidated**

```java
public void setValidated(org.eclipse.core.resources.IResource[] validated)
```

Indicate that additional resources have been validated as part of this validate operation. Sometimes in the course of performing a validation on one resource it is necessary to validate other resources as well. This method is used to let the framework know about these additional validated resources, to possibly save them from being validated redundantly.

**Parameters:**

- `validated` -
Why move to the New Framework?

- Performance
  - ✔ Caching
  - ✔ Job Control
  - ✔ Content Type
  - ✔ Fine Grain Control

- Dependency Support
  - ✔ UI Dialog
  - ✔ Simplicity and Documentation
  - ✔ Framework Collateral
  - ✔ Logging and Auditing