UpdateRequestProcessor

/!\ Solr1.3

UpdateRequestProcessors can be mixed and matched in UpdateRequestProcessorChains that define how update requests are processed.

- Configuring Update Request Processors
- Selecting the UpdateChain for Your Request
- Distributed Updates
- Processor Customization Examples
  - Field Mutating Update Processors
  - Implementing a conditional copyField
  - Script UpdateProcessors
- Full list of UpdateRequestProcessor Factories

Configuring Update Request Processors

solrconfig.xml files can contain any number of UpdateRequestProcessorChains...

```
<updateRequestProcessorChain name="mychain" default="true">
  <processor class="com.example.MyCustomProcessorFactory" >
    <lst name="name">
      <str name="n1">x1</str>
      <str name="n2">x2</str>
    </lst>
  </processor>
  <processor class="solr.LogUpdateProcessorFactory" />
  <processor class="solr.RunUpdateProcessorFactory" />
</updateRequestProcessorChain>
```

Note the entirely custom class in this example called "com.example.MyCustomProcessorFactory". This is an example of a class that you might create yourself by writing Java code that extends the UpdateRequestProcessorFactory class.

Almost all processor chains should end with an instance of `RunUpdateProcessorFactory` unless the user is explicitly executing the update commands in an alternative custom `UpdateRequestProcessorFactory`.

At most one processor chain may be configured as the "default". if no processor is configured as a default, then an implicit default using `LogUpdateProcessorFactory` and `RunUpdateProcessorFactory` is created for you. Supplying a default processor chain may be the only way to affect documents indexed from some sources like the dataimport handler.

Selecting the UpdateChain for Your Request

Once one or more update chains are defined, you may select one on the update request through the parameter `update.chain`

Example: `http://localhost:8983/solr/update/xml?update.chain=mychain`.

⚠️ **Note:** for Solr versions prior to Solr3.2 you need to use `update.processor` instead ⚠️

You may also choose to set a default chain for a certain UpdateRequestHandler:

```
<!-- referencing it in an update handler -->
<requestHandler name="/update/processortest" class="solr.JsonUpdateRequestHandler" >
  <lst name="defaults">
    <str name="update.chain">mychain</str>
  </lst>
</requestHandler>
```

Distributed Updates

Begining with DistributingUpdateProcessorFactory] all UpdateRequestProcessorChains which include `RunUpdateProcessorFactory`, but do not include an implementation of the [http://lucene.apache.org/solr/4_0_0/solr-core/org/apache/solr/update/processor/DistributingUpdateProcessorFactory.html will have an instance of `DistributedUpdateProcessorFactory` automaticly injected immediately prior to the `RunUpdateProcessorFactory`.

In single-server instances, `DistributedUpdateProcessorFactory` is a No-Op, but for Solr Cloud instances, it determines where the in the chain the request gets forwarded to the leader (and other all other nodes in the shard). Processors prior to `DistributedUpdateProcessorFactory` in the chain will be executed only on the first node to recieve the command, processors after the `DistributedUpdateProcessorFactory` will be executed on every node in the appropriate shard(s).

Processor Customization Examples

Field Mutating Update Processors

Beginging with FieldMutatingUpdateProcessorFactories] various [http://lucene.apache.org/solr/api/org/apache/solr/update/processor/FieldMutatingUpdateProcessorFactory.html are available that can be mixed and matched to accomplish a variety of goals.

Implementing a conditional copyField

Here is a quick example that adds the 'cat' 'popular' if the value of 'popularity' is > 5

```
package my.solr;

import java.io.IOException;

import org.apache.solr.common.SolrInputDocument;
import org.apache.solr.request.SolrQueryRequest;
import org.apache.solr.response.SolrQueryResponse;
import org.apache.solr.update.AddUpdateCommand;
import org.apache.solr.update.processor.UpdateRequestProcessor;
import org.apache.solr.update.processor.UpdateRequestProcessorFactory;

public class ConditionalCopyProcessorFactory extends UpdateRequestProcessorFactory
{
  @Override
  public UpdateRequestProcessor getInstance(SolrQueryRequest req, SolrQueryResponse rsp, UpdateRequestProcessor
next)
  {
    return new ConditionalCopyProcessor(next);
  }
}

class ConditionalCopyProcessor extends UpdateRequestProcessor
{
  public ConditionalCopyProcessor( UpdateRequestProcessor next) {
    super( next );
  }

  @Override
  public void processAdd(AddUpdateCommand cmd) throws IOException {
    SolrInputDocument doc = cmd.getSolrInputDocument();

    Object v = doc.getFieldValue( "popularity" );
    if( v != null ) {
      int pop = Integer.parseInt( v.toString() );
      if( pop > 5 ) {
        doc.addField( "cat", "popular" );
      }
    }

    // pass it up the chain
    super.processAdd(cmd);
  }
}
```

Script UpdateProcessors

SOLR-1725, to be committed for Solr 4.0 final release, adds a ScriptUpdateProcessor. There is a StatelessScriptUpdateProcessorFactory included to configure into your solrconfig.xml configuration file. The ScriptUpdateProcessor allows for Java scripting engines to be used during the Solr document update processing, allowing dramatic flexibility in expressing custom document processing before being indexed. (it also allows hooks to commit, delete, etc, but add will have the most common usage). More coming soon!

Full list of UpdateRequestProcessor Factories

The nearly-complete list of the factories can be found by looking at the Javadoc for UpdateRequestProcessorFactory and navigating through the various children and grand-children subclasses. Notice that it will only show children classes in the *solr-core* group. This does not include factories that come from other modules, such as UIMAUpdateRequestProcessorFactory . Interesting mention for the Solr Classification Update Request Processor SolrClassification UpdateRequestProcessor

The complete list has also been compiled for the convenience by the Solr-Start project.