Solr.xml (supported through 4.x)

Solr.xml

Deprecated in 4.4 and unsupported as of 5.0, see Solr.xml 4.4 and beyond

To enable support for dynamic SolrCore administration, place a file named *solr.xml* in the solr.home directory. Here is an example *solr.xml* file:

```
<solr persistent="true" sharedLib="lib">
 <cores adminPath="/admin/cores">
  <core name="core0" instanceDir="core0" />
  <core name="core1" instanceDir="core1" />
 </cores>
</solr>
```

You can also specify properties in solr.xml which can be used in the solrconfig.xml and schema.xml files.

```
<solr persistent="true" sharedLib="lib">
 <property name="snapshooter" value="/home/solr-user/solr/bin/snapshooter.sh" />
 <cores adminPath="/admin/cores">
  <core name="core0" instanceDir="core0">
    <property name="dataDir" value="/data/core0" />
  </core>
  <core name="core1" instanceDir="core1" />
 </cores>
</solr>
```

The properties can be container scope (i.e. specified after <solr> but outside of a <core> element) in which case it is automatically inherited by each core. Therefore, they can be used in any of the cores' configuration files.

The properties can also be defined in a core's scope (inside the <core> element) in which case they can be used only in that core's scope. If a property by that name already exists in the container scope then it will be overridden.

Besides them, a few properties are automatically added in the core scope. They are:

- `solr.core.name` – The core's name as defined in solr.xml
- `solr.core.instanceDir` – The core's instance directory (i.e. the directory under which that core's `conf/` and `data/` directory are located)
- `solr.core.dataDir` – The core's data directory (i.e. the directory under which that core's index directory are located)
- `solr.core.configName` – The name of the core's config file (solrconfig.xml by default)
- `solr.core.schemaName` – The name of the core's schema file (schema.xml by default)

Such properties can be used inside solrconfig.xml and schema.xml files by specifying an expression with optionally, a default value.

```
// Without a default value
${snapshooter}
// With a default value
${snapshooter:./solr/bin/snapshooter.sh}
```

The above expression will evaluate to the value specified in solr.xml for the property name "solr.snapshooter". If no value is defined in solr.xml, it will check if a system property by that name exists otherwise it will use the specified default value. If no default value is specified, a runtime exception will be thrown and the core may fail to startup.

solr

The `<solr>` tag accepts two attributes:
***persistent** - By default, should runtime core manipulation be saved in *solr.xml* so that it is available after a restart.
***sharedLib** - Path to a directory containing .jar files that are added to the classpath of every core. The path is relative to solr.home (where solr.xml sits)

- ⚠ Solr4.0 **zkHost** - Used for SolrCloud, zookeeper server list.

cores

The `<cores>` tag accepts the following attributes:
***hostPort** - The port solr uses to access cores. For example, you can set hostPort="8080" when working with Tomcat.
***adminPath** - Relative path to access the CoreAdminHandler for dynamic core manipulation. For example, adminPath="/admin/cores" configures access via http://localhost:8983/solr/admin/cores. If this attribute is not specified, dynamic manipulation is unavailable.
*** ⚠ Solr1.4 *defaultCoreName** - The name of a core that will be used for requests that don't specify a core. If you have one core and want to use the features specified on this page, then this provides a way to keep your URLs the same.
*** ⚠ Solr1.4 *shareSchema** - The value can be 'true' or 'false'. This ensures that the multiple cores pointing to the same schema.xml will be referring to the same IndexSchema Object. This makes loading the core faster. Ensure that no core specific property is used in your schema.xml.

- ⚠ Solr4.0 **hostContext** - the host context of solr url.
- ⚠ Solr4.0 **zkClientTimeout** - Timeout for connection of zookeeper server. It's used for SolrCloud environment for ensure the connection status.
- ⚠ LotsOfCores] **transientCacheSize** - The limit for the LRU cache of "lazily loaded" cores. More at [LotsOfCores
- ⚠ Solr1.4 **adminHandler** - FQN(Fully qualified name) of a class that inherits from CoreAdminHandler. For example, adminHandler="com.myorg.MyAdminHandler" would configure the custom admin handler (MyAdminHandler) to handler admin requests (as opposed to org.apache.solr.handler.admin.CoreAdminHandler , that is the default admin handler if one is not specified). To illustrate an use case for the same - suppose if there is a need to get some statistics from different cores in a solr instance - we would proceed as follows.
  - Define a new action called 'mystat' that could be accessed from the client as below. http://localhost:8983/solr/admin/cores?action=MYSTAT
  - ⚠ Solr4.0 Note that to load up your custom handler, you may need to use the sharedLib attribute mentioned in the `<solr>` tag section.
  - Define the implementation of that action as

```
import org.apache.solr.handler.admin.CoreAdminHandler ;

class MyAdminHandler extends CoreAdminHandler {

  /**
   * @return true, if the changes need to be persisted by the CoreContainer. (use only if solr.xml would be
changed because of this action. )
   *         false, otherwise.   (Use this if unsure or having a read-only access to the CoreContainer like
collecting statistics)
   *
   */
  protected boolean handleCustomAction(SolrQueryRequest req, SolrQueryResponse rsp) {
     CoreContainer container = super.getCoreContainer();
     SolrCore mycore1 = container.getCore("core1");
     SolrCore mycore2 = container.getCore("core2");
     SolrParams params = req.getParams();
     String a = params.get( CoreAdminParams.ACTION );
     if (a.equalsIgnoreCase("mystat"))  {
         // TODO: populate 'rsp' as necessary.
      }
  }
}
```

⚠ Solr4.0 Note that you may need to declare the two constructors for it to be instantiated successfully.

There are other methods in CoreAdminHandler that could be used to override default action-s, but for most of the common cases they would not be necessary but left best to the experts.

```
class MyAdminHandler extends CoreAdminHandler {
    //Available for override , but unnecessary except for the rare case.
     protected boolean handleAliasAction(SolrQueryRequest req, SolrQueryResponse rsp) ;
     protected boolean handleCreateAction(SolrQueryRequest req, SolrQueryResponse rsp) ;
     // etc.
}
```

See SOLR-1106 for more details.

The `<core>` tag accepts the following attributes:
***name** - The registered core name. This will be how the core is accessed.
***instanceDir** - The *solr.home* directory for a given core.
***config** - The configuration file name for a given core. The default is 'solrconfig.xml'.
***schema** - The schema file name for a given core. The default is 'schema.xml'.
***dataDir** - The data directory for a given core. The default is <instanceDir>/data . It can take an absolute path or a relative path w.r.t instanceDir . ⚠ Solr1.4

- **loadOnStartup** - true|false. Whether the core should be completely loaded upon startup. More at LotsOfCores ⚠ Solr4.1
- **transient**- true|false. Whether the core is allowed to be swapped out or not. More at LotsOfCores ⚠ Solr4.1 ***properties** - The core properties file name.This can be an absolute or relative path(relative to instanceDir) ⚠ Solr1.4

property

The `<property>` tag accepts two attributes:
***name** - The name of the property
***value** - The value of the property

solr.xml Permissions

It is important to note that persistent=true functionality **'replaces**' solr.xml it does not edit it. This means that the directory the file is in needs to allow the web server to replace the file. If the permissions are set incorrectly it will give 500 errors and throw IOExceptions. Additionally, all comments are wiped from the file on save.

Example

Solr ships with an example running two cores together setup. To run this configuration, start jetty in the `example/` directory using:

```
java -Dsolr.solr.home=multicore -jar start.jar
```

This will start solr running two cores: *core0*, and *core1*. To access each core, try:
http://localhost:8983/solr/core0/select?q=&#42;:&#42;

http://localhost:8983/solr/core1/select?q=&#42;:&#42;

To access the admin pages for each core visit:
http://localhost:8983/solr/core0/admin/

http://localhost:8983/solr/core1/admin/