What's new in grepo v1.5

- Introduction
- New features
  - Custom grepo spring namespaces
    - query-hibernate namespace
    - query-jpa namespace
    - procedure namespace
  - OSGI artifacts
  - Simple Logging Facade for Java (slf4j)
  - Generic Statistics (gstatistics) component
- Upgrading from v1.0
  - Location of default config files has changed
  - Procedure repositories must extend from GenericProcedureRepository

Introduction

This article summarizes the main features of the grepo framework v1.5 release. For a full list of changes see the Jira Release Notes.

New features

Custom grepo spring namespaces

Grepo now provides custom namespace(s) which should make configuration of repository beans (DAOs) easier.

query-hibernate namespace

In order to use the grepo query-hibernate namespace your spring configuration file has to declare the query-hibernate namspace as follows:

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:gquery="http://grepo.codehaus.org/schema/query-hibernate"
    xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://grepo.codehaus.org/schema/query-hibernate
http://grepo.codehaus.org/schema/grepo-query-hibernate.xsd">

<!-- your beans here -->

</beans>
```

repository-factory

You can define an *abstract* factory bean like this:

```
<gquery:repository-factory id="repositoryFactory" />
```

This is the equal to the following configuration:

```
<bean id="repositoryFactory"
  class="org.codehaus.grepo.query.hibernate.repository.HibernateRepositoryFactoryBean"
abstract="true" />
```

repository

You can define concrete repository beans like this:

```
<gquery:repository id="userRepository" factory="repositoryFactory"
        proxy-class="demo.repository.UserRepository" />
```

This is equal to the following configuration:

```
<bean id="userRepository" parent="repositoryFactory">
    <property name="proxyClass" value="demo.repository.UserRepository" />
</bean>
```

repository-scan

You can also let grepo automatically detect concrete repository beans using the repository scanning feature. This feature is similar (based on) Spring's component-scan feature. You would use it like this:

```
<gquery:repository-scan base-package="demo.repository" factory="repositoryFactory" />
```

Using the configuration above, grepo will scan the package (and all sub-packages) *demo.repository* and automatically create repository beans which are based on the *abstract* factory bean '*repositoryFactory*'.

Note: The repository interfaces (or implementations) have to be annotated with Spring's *Repository* annotation and furthermore must extend/implement from the *GenericQueryRepository* interface in order to be detected by the repository-scanner.

Similar to spring's component-scan feature you can use the *include-filter/exclude-filter* elements to refine repository scanning. For example:

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gquery="http://grepo.codehaus.org/schema/query-hibernate"
  xmlns:gcore="http://grepo.codehaus.org/schema/core"
  xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://grepo.codehaus.org/schema/core
http://grepo.codehaus.org/schema/grepo-core.xsd
      http://grepo.codehaus.org/schema/query-hibernate
http://grepo.codehaus.org/schema/grepo-query-hibernate.xsd">

  <gquery:repository-scan base-package="demo.repository" factory="abstractFactory">
      <gcore:exclude-filter type="regex"
expression="demo.repository.notscannedpackage.*" />
  </gquery:repository-scan>

</beans>
```

query-jpa namespace

In order to use the grepo query-jpa namespace in your spring configuration file has to declare the query-jpa namspace as follows:

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:gquery="http://grepo.codehaus.org/schema/query-jpa"
    xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://grepo.codehaus.org/schema/query-jpa
http://grepo.codehaus.org/schema/grepo-query-jpa.xsd">

<!-- your beans here -->

</beans>
```

repository-factory

You can define an *abstract* factory bean like this:

```
<gquery:repository-factory id="repositoryFactory" />
```

This is the equal to the following configuration:

```
<bean id="repositoryFactory"
  class="org.codehaus.grepo.query.jpa.repository.JpaRepositoryFactoryBean"
abstract="true" />
```

repository

You can define concrete repository beans like this:

```
<gquery:repository id="userRepository" factory="repositoryFactory"
       proxy-class="demo.repository.UserRepository" />
```

This is equal to the following configuration:

```
<bean id="userRepository" parent="repositoryFactory">
    <property name="proxyClass" value="demo.repository.UserRepository" />
</bean>
```

repository-scan

You can also let grepo automatically detect concrete repository beans using the repository scanning feature. This feature is similar (based on) Spring's component-scan feature. You would use it like this:

```
<gquery:repository-scan base-package="demo.repository" factory="repositoryFactory" />
```

Using the configuration above, grepo will scan the package (and all sub-packages) *demo.repository* and automatically create repository beans which are based on the *abstract* factory bean '*repositoryFactory*'.

Note: The repository interfaces (or implementations) have to be annotated with Spring's *Repository* annotation and furthermore must extend/implement from the *GenericQueryRepository* interface in order to be detected by the repository-scanner.

Similar to spring's component-scan feature you can use the *include-filter*/*exclude-filter* elements to refine repository scanning. For example:

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gquery="http://grepo.codehaus.org/schema/query-jpa"
  xmlns:gcore="http://grepo.codehaus.org/schema/core"
  xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://grepo.codehaus.org/schema/core
http://grepo.codehaus.org/schema/grepo-core.xsd
      http://grepo.codehaus.org/schema/query-jpa
http://grepo.codehaus.org/schema/grepo-query-jpa.xsd">

  <gquery:repository-scan base-package="demo.repository" factory="abstractFactory">
      <gcore:exclude-filter type="regex"
expression="demo.repository.notscannedpackage.*" />
  </gquery:repository-scan>

</beans>
```

procedure namespace

In order to use the grepo procedure namespace in your spring configuration file has to declare the procedure namspace as follows:

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:gproc="http://grepo.codehaus.org/schema/procedure"
    xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://grepo.codehaus.org/schema/procedure
http://grepo.codehaus.org/schema/grepo-procedure.xsd">

<!-- your beans here -->

</beans>
```

repository-factory

You can define an *abstract* factory bean like this:

```
<gproc:repository-factory id="repositoryFactory" />
```

This is the equal to the following configuration:

```
<bean id="repositoryFactory"

class="org.codehaus.grepo.procedure.repository.GenericProcedureRepositoryFactoryBean"
abstract="true" />
```

repository

You can define concrete repository beans like this:

```
<gproc:repository id="demoProcedure" factory="repositoryFactory"
        proxy-class="demo.repository.DemoProcedure" />
```

This is equal to the following configuration:

```
<bean id="demoProcedure" parent="repositoryFactory">
    <property name="proxyClass" value="demo.repository.DemoProcedure" />
</bean>
```

repository-scan

You can also let grepo automatically detect concrete repository beans using the repository scanning feature. This feature is similar (based on) Spring's component-scan feature. You would use it like this:

```
<gproc:repository-scan base-package="demo.repository" factory="repositoryFactory" />
```

Using the configuration above, grepo will scan the package (and all sub-packages) *demo.repository* and automatically create repository beans which are based on the *abstract* factory bean '*repositoryFactory*'.

Note: The repository interfaces (or implementations) have to be annotated with Spring's *Repository* annotation and furthermore must extend/implement from the *GenericProcedureRepository* interface in order to be detected by the repository-scanner.

Similar to spring's component-scan feature you can use the *include-filter/exclude-filter* elements to refine repository scanning. For example:

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gproc="http://grepo.codehaus.org/schema/procedure"
  xmlns:gcore="http://grepo.codehaus.org/schema/core"
  xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://grepo.codehaus.org/schema/core
http://grepo.codehaus.org/schema/grepo-core.xsd
      http://grepo.codehaus.org/schema/procedure
http://grepo.codehaus.org/schema/grepo-procedure.xsd">

  <gproc:repository-scan base-package="demo.repository" factory="abstractFactory">
      <gcore:exclude-filter type="regex"
expression="demo.repository.notscannedpackage.*" />
  </gproc:repository-scan>

</beans>
```

OSGI artifacts

Grepo artifacts (jars) now include osgi relevant information in it's *MANIFEST.MF* file. The following is an example of the *MANIFEST.MF* of the grepo-core artifact:

```
```

```
Manifest-Version: 1.0
Archiver-Version: Plexus Archiver
Created-By: Apache Maven
Built-By: dguggi
Build-Jdk: 1.5.0_17
Bundle-Name: grepo-core
Import-Package: org.aopalliance.aop;version="[1.0.0,2.0.0)",org.aopall
 iance.intercept;version="[1.0.0,2.0.0)",org.apache.commons.lang;versi
 on="[2.0.0,3.0.0)",org.slf4j;version="[1.5.0,2.0.0)",org.springframew
 ork.aop.framework;version="[2.5.0,4.0.0)",org.springframework.beans;v
 ersion="[2.5.0,4.0.0)",org.springframework.beans.factory;version="[2.
 5.0,4.0.0)",org.springframework.beans.factory.annotation;version="[2.
 5.0,4.0.0)",org.springframework.beans.factory.config;version="[2.5.0,
 4.0.0)",org.springframework.beans.factory.parsing;version="[2.5.0,4.0
 .0)",org.springframework.beans.factory.support;version="[2.5.0,4.0.0)
 ",org.springframework.beans.factory.xml;version="[2.5.0,4.0.0)",org.s
 pringframework.context;version="[2.5.0,4.0.0)",org.springframework.co
 ntext.annotation;version="[2.5.0,4.0.0)",org.springframework.core.io;
 version="[2.5.0,4.0.0)",org.springframework.core.type;version="[2.5.0
 ,4.0.0)",org.springframework.core.type.classreading;version="[2.5.0,4
 .0.0)",org.springframework.core.type.filter;version="[2.5.0,4.0.0)",o
 rg.springframework.dao;version="[2.5.0,4.0.0)",org.springframework.st
 ereotype;version="[2.5.0,4.0.0)",org.springframework.transaction.supp
 ort;version="[2.5.0,4.0.0)",org.springframework.util;version="[2.5.0,
 4.0.0)",org.w3c.dom;version="0.0.0"
Bundle-ManifestVersion: 2
Bundle-RequiredExecutionEnvironment: J2SE-1.5
Bundle-Vendor: Generic Repository Framework
Bundle-SymbolicName: org.codehaus.grepo.core
Tool: Bundlor 1.0.0.RELEASE
Export-Package: org.codehaus.grepo.core.annotation;version="1.5.0",org
 .codehaus.grepo.core.aop;version="1.5.0",org.codehaus.grepo.core.conf
 ig;version="1.5.0";uses:="org.springframework.beans.factory.annotatio
 n,org.springframework.beans.factory.config,org.springframework.beans.
 factory.support,org.springframework.beans.factory.xml,org.springframe
 work.context.annotation,org.springframework.core.type.classreading,or
 g.springframework.core.type.filter,org.w3c.dom",org.codehaus.grepo.co
 re.converter;version="1.5.0";uses:="org.codehaus.grepo.core.aop,org.c
 odehaus.grepo.core.exception,org.codehaus.grepo.core.registry",org.co
 dehaus.grepo.core.exception;version="1.5.0";uses:="org.springframewor
 k.dao",org.codehaus.grepo.core.executor;version="1.5.0";uses:="org.sp
 ringframework.context",org.codehaus.grepo.core.registry;version="1.5.
 0";uses:="org.springframework.dao",org.codehaus.grepo.core.repository
 ;version="1.5.0";uses:="org.aopalliance.intercept,org.codehaus.grepo.
 core.converter,org.codehaus.grepo.core.exception,org.springframework.
 beans,org.springframework.beans.factory,org.springframework.context,o
 rg.springframework.transaction.support",org.codehaus.grepo.core.util;
 version="1.5.0";uses:="org.codehaus.grepo.core.exception",org.codehau
 s.grepo.core.validator;version="1.5.0";uses:="org.codehaus.grepo.core
 .aop,org.codehaus.grepo.core.exception,org.slf4j"
Bundle-Version: 1.5.0
```

Simple Logging Facade for Java (slf4j)

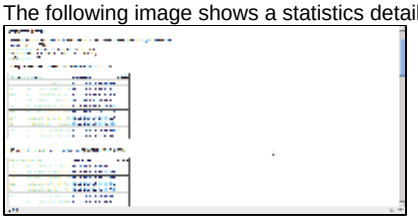
Grepo now uses the slf4j logging api - so the commons-logging api is not required anymore by the framework.

Generic Statistics (gstatistics) component

The *"Generic Statistics"* component allows to collect runtime statistics of software layers. The component is easily extensible in order to fit custom needs. The gquery and gprocedure components provide out of the box support to collect statistics for dao layer(s) using this component. Because of its extensibility collecting statistics is not only limited to class methods but can also be used to track entire business processes and the like.

The following image shows a statistics summary view of an application using gstatistics (via jboss jmx-console):

The following image shows a statistics detail view of an application using gstatistics (via jboss jmx-console):



Upgrading from v1.0

Version 1.5.x is not fully compatible with Version 1.0.x - the following has to be considered when upgrading from grepo version 1.0.x to grepo version 1.5.x:

Location of default config files has changed

The location for the default config files of the grepo framework has changed.

grepo-query-hibernate-default.cfg.xml

```
<import resource="classpath:META-INF/grepo/grepo-query-hibernate-default.cfg.xml"/>
```

grepo-query-jpa-default.cfg.xml

```
<import resource="classpath:META-INF/grepo/grepo-query-jpa-default.cfg.xml"/>
```

grepo-procedure-default.cfg.xml

```
<import resource="classpath:META-INF/grepo/grepo-procedure-default.cfg.xml"/>
```

Procedure repositories must extend from GenericProcedureRepository

Procedure repositories must extend from *GenericProcedureRepository* as follows:

```
public interface DemoProcedure extends GenericProcedureRepository { }
```