org.springframework.context

Interface ConfigurableApplicationContext

All Superinterfaces:

ApplicationContext, ApplicationEventPublisher, AutoCloseable, BeanFactory, Closeable, EnvironmentCapable, HierarchicalBeanFactory, Lifecycle, ListableBeanFactory, MessageSource, ResourceLoader, ResourcePatternResolver

All Known Subinterfaces:

ConfigurablePortletApplicationContext, ConfigurableWebApplicationContext

All Known Implementing Classes:

AbstractApplicationContext, AbstractRefreshableApplicationContext, AbstractRefreshableConfigApplicationContext, AbstractRefreshablePortletApplicationContext, AbstractRefreshableWebApplicationContext, AbstractXmlApplicationContext, AnnotationConfigApplicationContext, AnnotationConfigWebApplicationContext, ClassPathXmlApplicationContext, FileSystemXmlApplicationContext, GenericApplicationContext, GenericGroovyApplicationContext, GenericWebApplicationContext, GenericXmlApplicationContext, GroovyWebApplicationContext, ResourceAdapterApplicationContext, StaticApplicationContext, StaticPortletApplicationContext, StaticWebApplicationContext, XmlPortletApplicationContext, XmlWebApplicationContext

---

```
public interface ConfigurableApplicationContext
extends ApplicationContext, Lifecycle, Closeable
```

SPI interface to be implemented by most if not all application contexts. Provides facilities to configure an application context in addition to the application context client methods in the ApplicationContext interface.

Configuration and lifecycle methods are encapsulated here to avoid making them obvious to ApplicationContext client code. The present methods should only be used by startup and shutdown code.

Since:

03.11.2003

Author:

Juergen Hoeller, Chris Beams

Field Summary

Fields

Modifier and Type	Field and Description
static String	**CONFIG_LOCATION_DELIMITERS** Any number of these characters are considered delimiters between multiple context config paths in a single String value.
static String	**CONVERSION_SERVICE_BEAN_NAME** Name of the ConversionService bean in the factory.

static String	**ENVIRONMENT_BEAN_NAME** Name of the Environment bean in the factory.
static String	**LOAD_TIME_WEAVER_BEAN_NAME** Name of the LoadTimeWeaver bean in the factory.
static String	**SYSTEM_ENVIRONMENT_BEAN_NAME** Name of the System environment bean in the factory.
static String	**SYSTEM_PROPERTIES_BEAN_NAME** Name of the System properties bean in the factory.

Fields inherited from interface org.springframework.beans.factory.BeanFactory

FACTORY_BEAN_PREFIX

Fields inherited from interface org.springframework.core.io.support.ResourcePatternResolver

CLASSPATH_ALL_URL_PREFIX

Fields inherited from interface org.springframework.core.io.ResourceLoader

CLASSPATH_URL_PREFIX

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method and Description
void	**addApplicationListener**(ApplicationListener<?> listener) Add a new ApplicationListener that will be notified on context events such as context refresh and context shutdown.
void	**addBeanFactoryPostProcessor**(BeanFactoryPostProcessor postProcessor) Add a new BeanFactoryPostProcessor that will get applied to the internal bean factory of this application context on refresh, before any of the bean definitions get evaluated.
void	**addProtocolResolver**(ProtocolResolver resolver) Register the given protocol resolver with this application context, allowing for additional resource protocols to be handled.
void	**close**() Close this application context, releasing all resources and locks that the implementation might hold.
ConfigurableListableBeanFactory	**getBeanFactory**() Return the internal bean factory of this application context.
ConfigurableEnvironment	**getEnvironment**() Return the Environment for this application context in configurable form, allowing for further customization.

boolean	**isActive**() Determine whether this application context is active, that is, whether it has been refreshed at least once and has not been closed yet.
void	**refresh**() Load or refresh the persistent representation of the configuration, which might an XML file, properties file, or relational database schema.
void	**registerShutdownHook**() Register a shutdown hook with the JVM runtime, closing this context on JVM shutdown unless it has already been closed at that time.
void	**setEnvironment**(**ConfigurableEnvironment** environment) Set the `Environment` for this application context.
void	**setId**(**String** id) Set the unique id of this application context.
void	**setParent**(**ApplicationContext** parent) Set the parent of this application context.

Methods inherited from interface org.springframework.context.ApplicationContext

getApplicationName, getAutowireCapableBeanFactory, getDisplayName, getId, getParent, getStartupDate

Methods inherited from interface org.springframework.beans.factory.ListableBeanFactory

containsBeanDefinition, findAnnotationOnBean, getBeanDefinitionCount, getBeanDefinitionNames, getBeanNamesForAnnotation, getBeanNamesForType, getBeanNamesForType, getBeanNamesForType, getBeansOfType, getBeansOfType, getBeansWithAnnotation

Methods inherited from interface org.springframework.beans.factory.HierarchicalBeanFactory

containsLocalBean, getParentBeanFactory

Methods inherited from interface org.springframework.beans.factory.BeanFactory

containsBean, getAliases, getBean, getBean, getBean, getBean, getBean, getType, isPrototype, isSingleton, isTypeMatch, isTypeMatch

Methods inherited from interface org.springframework.context.MessageSource

getMessage, getMessage, getMessage

Methods inherited from interface org.springframework.context.ApplicationEventPublisher

publishEvent, publishEvent

Methods inherited from interface org.springframework.core.io.support.ResourcePatternResolver

getResources

Methods inherited from interface org.springframework.core.io.ResourceLoader

getClassLoader, getResource

Methods inherited from interface org.springframework.context.Lifecycle

isRunning, start, stop

Field Detail

CONFIG_LOCATION_DELIMITERS

```
static final String CONFIG_LOCATION_DELIMITERS
```

Any number of these characters are considered delimiters between multiple context config paths in a single String value.

See Also:

AbstractRefreshableConfigApplicationContext.setConfigLocation(java.lang.String), ContextLoader.CONFIG_LOCATION_PARAM, FrameworkServlet.setContextConfigLocation(java.lang.String), Constant Field Values

CONVERSION_SERVICE_BEAN_NAME

```
static final String CONVERSION_SERVICE_BEAN_NAME
```

Name of the ConversionService bean in the factory. If none is supplied, default conversion rules apply.

Since:

3.0

See Also:

ConversionService, Constant Field Values

LOAD_TIME_WEAVER_BEAN_NAME

```
static final String LOAD_TIME_WEAVER_BEAN_NAME
```

Name of the LoadTimeWeaver bean in the factory. If such a bean is supplied, the context will use a temporary ClassLoader for type matching, in order to allow the LoadTimeWeaver to process all actual bean classes.

Since:

2.5

See Also:

LoadTimeWeaver, Constant Field Values

ENVIRONMENT_BEAN_NAME

```
static final String ENVIRONMENT_BEAN_NAME
```

Name of the `Environment` bean in the factory.

Since:

`3.1`

See Also:

`Constant Field Values`

SYSTEM_PROPERTIES_BEAN_NAME

```
static final String SYSTEM_PROPERTIES_BEAN_NAME
```

Name of the System properties bean in the factory.

See Also:

`System.getProperties(), Constant Field Values`

SYSTEM_ENVIRONMENT_BEAN_NAME

```
static final String SYSTEM_ENVIRONMENT_BEAN_NAME
```

Name of the System environment bean in the factory.

See Also:

`System.getenv(), Constant Field Values`

Method Detail

setId

```
void setId(String id)
```

Set the unique id of this application context.

Since:

`3.0`

setParent

```
void setParent(ApplicationContext parent)
```

Set the parent of this application context.

Note that the parent shouldn't be changed: It should only be set outside a constructor if it isn't available when an object of this class is created, for example in case of WebApplicationContext setup.

Parameters:

`parent - the parent context`

See Also:

`ConfigurableWebApplicationContext`

setEnvironment

```
void setEnvironment(ConfigurableEnvironment environment)
```

Set the `Environment` for this application context.

Parameters:

`environment - the new environment`

Since:

`3.1`

getEnvironment

```
ConfigurableEnvironment getEnvironment()
```

Return the `Environment` for this application context in configurable form, allowing for further customization.

Specified by:

`getEnvironment in interface EnvironmentCapable`

Since:

`3.1`

addBeanFactoryPostProcessor

```
void addBeanFactoryPostProcessor(BeanFactoryPostProcessor postProcessor)
```

Add a new BeanFactoryPostProcessor that will get applied to the internal bean factory of this application context on refresh, before any of the bean definitions get evaluated. To be invoked during context configuration.

Parameters:

`postProcessor - the factory processor to register`

addApplicationListener

```
void addApplicationListener(ApplicationListener<?> listener)
```

Add a new ApplicationListener that will be notified on context events such as context refresh and context shutdown.

Note that any ApplicationListener registered here will be applied on refresh if the context is not active yet, or on the fly with the current event multicaster in case of a context that is already active.

Parameters:

`listener - the ApplicationListener to register`

See Also:

`ContextRefreshedEvent, ContextClosedEvent`

addProtocolResolver

```
void addProtocolResolver(ProtocolResolver resolver)
```

Register the given protocol resolver with this application context, allowing for additional resource protocols to be handled.

Any such resolver will be invoked ahead of this context's standard resolution rules. It may therefore also override any default rules.

Since:

`4.3`

refresh

```
void refresh()
      throws BeansException,
             IllegalStateException
```

Load or refresh the persistent representation of the configuration, which might an XML file, properties file, or relational database schema.

As this is a startup method, it should destroy already created singletons if it fails, to avoid dangling resources. In other words, after invocation of that method, either all or no singletons at all should be instantiated.

Throws:

`BeansException` - if the bean factory could not be initialized

`IllegalStateException` - if already initialized and multiple refresh attempts are not supported

registerShutdownHook

```
void registerShutdownHook()
```

Register a shutdown hook with the JVM runtime, closing this context on JVM shutdown unless it has already been closed at that time.

This method can be called multiple times. Only one shutdown hook (at max) will be registered for each context instance.

See Also:

`Runtime.addShutdownHook(java.lang.Thread)`, `close()`

close

```
void close()
```

Close this application context, releasing all resources and locks that the implementation might hold. This includes destroying all cached singleton beans.

Note: Does *not* invoke `close` on a parent context; parent contexts have their own, independent lifecycle.

This method can be called multiple times without side effects: Subsequent `close` calls on an already closed context will be ignored.

Specified by:

`close` in interface `AutoCloseable`

Specified by:

close in interface Closeable

isActive

```
boolean isActive()
```

Determine whether this application context is active, that is, whether it has been refreshed at least once and has not been closed yet.

Returns:

whether the context is still active

See Also:

refresh(), close(), getBeanFactory()

getBeanFactory

```
ConfigurableListableBeanFactory getBeanFactory()
                                        throws IllegalStateException
```

Return the internal bean factory of this application context. Can be used to access specific functionality of the underlying factory.

Note: Do not use this to post-process the bean factory; singletons will already have been instantiated before. Use a BeanFactoryPostProcessor to intercept the BeanFactory setup process before beans get touched.

Generally, this internal factory will only be accessible while the context is active, that is, inbetween refresh() and close(). The isActive() flag can be used to check whether the context is in an appropriate state.

Returns:

the underlying bean factory

Throws:

IllegalStateException - if the context does not hold an internal bean factory (usually if refresh() hasn't been called yet or if close() has already been called)

See Also:

isActive(), refresh(), close(), addBeanFactoryPostProcessor(org.springframework.beans.factory.config.BeanFactoryPostProcessor)