org.springframework.web.filter

Class DelegatingFilterProxy

java.lang.Object
    org.springframework.web.filter.GenericFilterBean
        org.springframework.web.filter.DelegatingFilterProxy

All Implemented Interfaces:

`Filter, Aware, BeanNameAware, DisposableBean, InitializingBean, EnvironmentAware, EnvironmentCapable, ServletContextAware`

```
public class DelegatingFilterProxy
extends GenericFilterBean
```

Proxy for a standard Servlet Filter, delegating to a Spring-managed bean that implements the Filter interface. Supports a "targetBeanName" filter init-param in `web.xml`, specifying the name of the target bean in the Spring application context.

`web.xml` will usually contain a `DelegatingFilterProxy` definition, with the specified `filter-name` corresponding to a bean name in Spring's root application context. All calls to the filter proxy will then be delegated to that bean in the Spring context, which is required to implement the standard Servlet Filter interface.

This approach is particularly useful for Filter implementation with complex setup needs, allowing to apply the full Spring bean definition machinery to Filter instances. Alternatively, consider standard Filter setup in combination with looking up service beans from the Spring root application context.

NOTE: The lifecycle methods defined by the Servlet Filter interface will by default *not* be delegated to the target bean, relying on the Spring application context to manage the lifecycle of that bean. Specifying the "targetFilterLifecycle" filter init-param as "true" will enforce invocation of the `Filter.init` and `Filter.destroy` lifecycle methods on the target bean, letting the servlet container manage the filter lifecycle.

As of Spring 3.1, `DelegatingFilterProxy` has been updated to optionally accept constructor parameters when using Servlet 3.0's instance-based filter registration methods, usually in conjunction with Spring 3.1's `WebApplicationInitializer` SPI. These constructors allow for providing the delegate Filter bean directly, or providing the application context and bean name to fetch, avoiding the need to look up the application context from the ServletContext.

This class was originally inspired by Spring Security's `FilterToBeanProxy` class, written by Ben Alex.

Since:

1.2

Author:

Juergen Hoeller, Sam Brannen, Chris Beams

See Also:

`setTargetBeanName(java.lang.String), setTargetFilterLifecycle(boolean), Filter.doFilter(javax.servlet.ServletRequest, javax.servlet.ServletResponse, javax.servlet.FilterChain), Filter.init(javax.servlet.FilterConfig), Filter.destroy(), DelegatingFilterProxy(Filter), DelegatingFilterProxy(String), DelegatingFilterProxy(String, WebApplicationContext), ServletContext.addFilter(String, Filter), WebApplicationInitializer`

Field Summary

Fields inherited from class org.springframework.web.filter.GenericFilterBean

`logger`

Constructor Summary

Constructors

Constructor and Description
`DelegatingFilterProxy()` Create a new `DelegatingFilterProxy`.
`DelegatingFilterProxy(Filter delegate)` Create a new `DelegatingFilterProxy` with the given `Filter` delegate.
`DelegatingFilterProxy(String targetBeanName)` Create a new `DelegatingFilterProxy` that will retrieve the named target bean from the Spring `WebApplicationContext` found in the `ServletContext` (either the 'root' application context or the context named by `setContextAttribute(java.lang.String)`).
`DelegatingFilterProxy(String targetBeanName, WebApplicationContext wac)` Create a new `DelegatingFilterProxy` that will retrieve the named target bean from the given Spring `WebApplicationContext`.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`destroy()` Subclasses may override this to perform custom filter shutdown.
`protected void`	`destroyDelegate(Filter delegate)` Destroy the Filter delegate.
`void`	`doFilter(ServletRequest request, ServletResponse response, FilterChain filterChain)`
`protected WebApplicationContext`	`findWebApplicationContext()` Return the `WebApplicationContext` passed in at construction time, if available.
`String`	`getContextAttribute()` Return the name of the ServletContext attribute which should be used to retrieve the `WebApplicationContext` from which to load the delegate `Filter` bean.
`protected String`	`getTargetBeanName()` Return the name of the target bean in the Spring application context.

protected Filter	initDelegate(WebApplicationContext wac) Initialize the Filter delegate, defined as bean the given Spring application context.
protected void	initFilterBean() Subclasses may override this to perform custom initialization.
protected void	invokeDelegate(Filter delegate, ServletRequest request, ServletResponse response, FilterChain filterChain) Actually invoke the delegate Filter with the given request and response.
protected boolean	isTargetFilterLifecycle() Return whether to invoke the `Filter.init` and `Filter.destroy` lifecycle methods on the target bean.
void	setContextAttribute(String contextAttribute) Set the name of the ServletContext attribute which should be used to retrieve the **WebApplicationContext** from which to load the delegate **Filter** bean.
void	setTargetBeanName(String targetBeanName) Set the name of the target bean in the Spring application context.
void	setTargetFilterLifecycle(boolean targetFilterLifecycle) Set whether to invoke the `Filter.init` and `Filter.destroy` lifecycle methods on the target bean.

Methods inherited from class org.springframework.web.filter.GenericFilterBean

`addRequiredProperty, afterPropertiesSet, createEnvironment, getEnvironment, getFilterConfig, getFilterName, getServletContext, init, initBeanWrapper, setBeanName, setEnvironment, setServletContext`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DelegatingFilterProxy

```
public DelegatingFilterProxy()
```

Create a new `DelegatingFilterProxy`. For traditional (pre-Servlet 3.0) use in `web.xml`.

See Also:
`setTargetBeanName(String)`

DelegatingFilterProxy

```
public DelegatingFilterProxy(Filter delegate)
```

Create a new `DelegatingFilterProxy` with the given `Filter` delegate. Bypasses entirely the need for interacting with a Spring application context, specifying the target bean name, etc.

For use in Servlet 3.0+ environments where instance-based registration of filters is supported.

Parameters:

`delegate - the Filter instance that this proxy will delegate to and manage the lifecycle for (must not be null).`

See Also:

`doFilter(ServletRequest, ServletResponse, FilterChain)`, `invokeDelegate(Filter, ServletRequest, ServletResponse, FilterChain)`, `destroy()`, `GenericFilterBean.setEnvironment(org.springframework.core.env.Environment)`

DelegatingFilterProxy

```
public DelegatingFilterProxy(String targetBeanName)
```

Create a new `DelegatingFilterProxy` that will retrieve the named target bean from the Spring `WebApplicationContext` found in the `ServletContext` (either the 'root' application context or the context named by `setContextAttribute(java.lang.String)`).

For use in Servlet 3.0+ environments where instance-based registration of filters is supported.

The target bean must implement the standard Servlet Filter.

Parameters:

`targetBeanName - name of the target filter bean to look up in the Spring application context (must not be null).`

See Also:

`findWebApplicationContext()`, `GenericFilterBean.setEnvironment(org.springframework.core.env.Environment)`

DelegatingFilterProxy

```
public DelegatingFilterProxy(String targetBeanName,
                             WebApplicationContext wac)
```

Create a new `DelegatingFilterProxy` that will retrieve the named target bean from the given Spring `WebApplicationContext`.

For use in Servlet 3.0+ environments where instance-based registration of filters is supported.

The target bean must implement the standard Servlet Filter interface.

The given `WebApplicationContext` may or may not be refreshed when passed in. If it has not, and if the context implements `ConfigurableApplicationContext`, a `refresh()` will be attempted before retrieving the named target bean.

This proxy's `Environment` will be inherited from the given `WebApplicationContext`.

Parameters:

`targetBeanName - name of the target filter bean in the Spring application context (must not be null).`

`wac - the application context from which the target filter will be retrieved; if null, an application context will be looked up from ServletContext as a fallback.`

See Also:

`findWebApplicationContext()`, `GenericFilterBean.setEnvironment(org.springframework.core.env.Environment)`

Method Detail

setContextAttribute

```
public void setContextAttribute(String contextAttribute)
```

Set the name of the ServletContext attribute which should be used to retrieve the `WebApplicationContext` from which to load the delegate `Filter` bean.

getContextAttribute

```
public String getContextAttribute()
```

Return the name of the ServletContext attribute which should be used to retrieve the `WebApplicationContext` from which to load the delegate `Filter` bean.

setTargetBeanName

```
public void setTargetBeanName(String targetBeanName)
```

Set the name of the target bean in the Spring application context. The target bean must implement the standard Servlet Filter interface.

By default, the `filter-name` as specified for the DelegatingFilterProxy in `web.xml` will be used.

getTargetBeanName

```
protected String getTargetBeanName()
```

Return the name of the target bean in the Spring application context.

setTargetFilterLifecycle

```
public void setTargetFilterLifecycle(boolean targetFilterLifecycle)
```

Set whether to invoke the `Filter.init` and `Filter.destroy` lifecycle methods on the target bean.

Default is "false"; target beans usually rely on the Spring application context for managing their lifecycle. Setting this flag to "true" means that the servlet container will control the lifecycle of the target Filter, with this proxy delegating the corresponding calls.

isTargetFilterLifecycle

```
protected boolean isTargetFilterLifecycle()
```

Return whether to invoke the `Filter.init` and `Filter.destroy` lifecycle methods on the target bean.

initFilterBean

```
protected void initFilterBean()
                       throws ServletException
```

Description copied from class: `GenericFilterBean`
Subclasses may override this to perform custom initialization. All bean properties of this filter will have been set before this method is invoked.

Note: This method will be called from standard filter initialization as well as filter bean initialization in a Spring application context. Filter name and ServletContext will be available in both cases.

This default implementation is empty.

Overrides:

`initFilterBean` in class `GenericFilterBean`

Throws:

`ServletException - if subclass initialization fails`

See Also:

`GenericFilterBean.getFilterName()`, `GenericFilterBean.getServletContext()`

doFilter

```
public void doFilter(ServletRequest request,
                     ServletResponse response,
                     FilterChain filterChain)
              throws ServletException,
                     IOException
```

Throws:

`ServletException`

`IOException`

destroy

```
public void destroy()
```

Description copied from class: `GenericFilterBean`
Subclasses may override this to perform custom filter shutdown.

Note: This method will be called from standard filter destruction as well as filter bean destruction in a Spring application context.

This default implementation is empty.

Specified by:

`destroy` in interface `Filter`

Specified by:

`destroy` in interface `DisposableBean`

Overrides:

`destroy` in class `GenericFilterBean`

findWebApplicationContext

```
protected WebApplicationContext findWebApplicationContext()
```

Return the `WebApplicationContext` passed in at construction time, if available. Otherwise, attempt to retrieve a `WebApplicationContext` from the `ServletContext` attribute with the configured name if set. Otherwise look up a `WebApplicationContext` under the well-known "root" application context attribute. The `WebApplicationContext` must have already been loaded and stored in the `ServletContext` before this filter gets initialized (or invoked).

Subclasses may override this method to provide a different `WebApplicationContext` retrieval strategy.

Returns:

`the WebApplicationContext for this proxy, or null if not found`

See Also:

`DelegatingFilterProxy(String, WebApplicationContext)`, `getContextAttribute()`, `WebApplicationContextUtils.getWebApplicationContext(javax.servlet.ServletContext)`, `WebApplicationContext.ROOT_WEB_APPLICATION_CONTEXT_ATTRIBUTE`

initDelegate

```
protected Filter initDelegate(WebApplicationContext wac)
                       throws ServletException
```

Initialize the Filter delegate, defined as bean the given Spring application context.

The default implementation fetches the bean from the application context and calls the standard `Filter.init` method on it, passing in the FilterConfig of this Filter proxy.

Parameters:

`wac - the root application context`

Returns:

`the initialized delegate Filter`

Throws:

`ServletException - if thrown by the Filter`

See Also:

`getTargetBeanName()`, `isTargetFilterLifecycle()`, `GenericFilterBean.getFilterConfig()`, `Filter.init(javax.servlet.FilterConfig)`

invokeDelegate

```
protected void invokeDelegate(Filter delegate,
                              ServletRequest request,
                              ServletResponse response,
                              FilterChain filterChain)
                       throws ServletException,
                              IOException
```

Actually invoke the delegate Filter with the given request and response.

Parameters:

`delegate - the delegate Filter`

`request - the current HTTP request`

`response - the current HTTP response`

`filterChain - the current FilterChain`

Throws:

`ServletException` - if thrown by the Filter

`IOException` - if thrown by the Filter

destroyDelegate

```
protected void destroyDelegate(Filter delegate)
```

Destroy the Filter delegate. Default implementation simply calls `Filter.destroy` on it.

Parameters:

`delegate` - the Filter delegate (never null)

See Also:

`isTargetFilterLifecycle()`, `Filter.destroy()`