org.springframework.web.servlet.view

Class InternalResourceViewResolver

```
java.lang.Object
    org.springframework.context.support.ApplicationObjectSupport
        org.springframework.web.context.support.WebApplicationObjectSupport
            org.springframework.web.servlet.view.AbstractCachingViewResolver
                org.springframework.web.servlet.view.UrlBasedViewResolver
                    org.springframework.web.servlet.view.InternalResourceViewResolver
```

All Implemented Interfaces:

`Aware`, `ApplicationContextAware`, `Ordered`, `ServletContextAware`, `ViewResolver`

```
public class InternalResourceViewResolver
extends UrlBasedViewResolver
```

Convenient subclass of `UrlBasedViewResolver` that supports `InternalResourceView` (i.e. Servlets and JSPs) and subclasses such as `JstlView`.

The view class for all views generated by this resolver can be specified via `UrlBasedViewResolver.setViewClass(java.lang.Class<?>)`. See `UrlBasedViewResolver`'s javadoc for details. The default is `InternalResourceView`, or `JstlView` if the JSTL API is present.

BTW, it's good practice to put JSP files that just serve as views under WEB-INF, to hide them from direct access (e.g. via a manually entered URL). Only controllers will be able to access them then.

Note: When chaining ViewResolvers, an InternalResourceViewResolver always needs to be last, as it will attempt to resolve any view name, no matter whether the underlying resource actually exists.

Since:

17.02.2003

Author:

Juergen Hoeller

See Also:

`UrlBasedViewResolver.setViewClass(java.lang.Class<?>)`, `UrlBasedViewResolver.setPrefix(java.lang.String)`, `UrlBasedViewResolver.setSuffix(java.lang.String)`, `UrlBasedViewResolver.setRequestContextAttribute(java.lang.String)`, `InternalResourceView`, `JstlView`

Field Summary

Fields inherited from class org.springframework.web.servlet.view.UrlBasedViewResolver

`FORWARD_URL_PREFIX`, `REDIRECT_URL_PREFIX`

Fields inherited from class org.springframework.web.servlet.view.AbstractCachingViewResolver

`DEFAULT_CACHE_LIMIT`

Fields inherited from class org.springframework.context.support.ApplicationObjectSupport

`logger`

Fields inherited from interface org.springframework.core.Ordered

`HIGHEST_PRECEDENCE, LOWEST_PRECEDENCE`

Constructor Summary

Constructors

Constructor and Description
`InternalResourceViewResolver()` Sets the default **`view class`** to **`requiredViewClass()`**: by default **`InternalResourceView`**, or **`JstlView`** if the JSTL API is present.
`InternalResourceViewResolver(String prefix, String suffix)` A convenience constructor that allows for specifying **`prefix`** and **`suffix`** as constructor arguments.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`protected AbstractUrlBasedView`	`buildView(String viewName)` Creates a new View instance of the specified view class and configures it.
`protected Class<?>`	`requiredViewClass()` This resolver requires **`InternalResourceView`**.
`void`	`setAlwaysInclude(boolean alwaysInclude)` Specify whether to always include the view rather than forward to it.

Methods inherited from class org.springframework.web.servlet.view.UrlBasedViewResolver

`canHandle, createView, getAttributesMap, getCacheKey, getContentType, getExposeContextBeansAsAttributes, getExposedContextBeanNames, getExposePathVariables, getOrder, getPrefix, getRedirectHosts, getRequestContextAttribute, getSuffix, getViewClass, getViewNames, initApplicationContext, isRedirectContextRelative, isRedirectHttp10Compatible,` loadView, setAttributes, setAttributesMap, setContentType, setExposeContextBeansAsAttributes, setExposedContextBeanNames, setExposePathVariables, setOrder, setPrefix, setRedirectContextRelative, setRedirectHosts, setRedirectHttp10Compatible, setRequestContextAttribute, setSuffix, setViewClass, setViewNames

Methods inherited from class org.springframework.web.servlet.view.AbstractCachingViewResolver

clearCache, getCacheLimit, isCache, isCacheUnresolved, removeFromCache, resolveViewName, setCache, setCacheLimit, setCacheUnresolved

Methods inherited from class org.springframework.web.context.support.WebApplicationObjectSupport

getServletContext, getTempDir, getWebApplicationContext, initApplicationContext, initServletContext, isContextRequired, setServletContext

Methods inherited from class org.springframework.context.support.ApplicationObjectSupport

getApplicationContext, getMessageSourceAccessor, requiredContextClass, setApplicationContext

Methods inherited from class java.lang.Object

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

Constructor Detail

InternalResourceViewResolver

```
public InternalResourceViewResolver()
```

Sets the default `view class` to `requiredViewClass()`: by default `InternalResourceView`, or `JstlView` if the JSTL API is present.

InternalResourceViewResolver

```
public InternalResourceViewResolver(String prefix,
                                    String suffix)
```

A convenience constructor that allows for specifying `prefix` and `suffix` as constructor arguments.

Parameters:

`prefix - the prefix that gets prepended to view names when building a URL`

`suffix - the suffix that gets appended to view names when building a URL`

Since:

4.3

Method Detail

requiredViewClass

```
protected Class<?> requiredViewClass()
```

This resolver requires `InternalResourceView`.

Overrides:

`requiredViewClass` in class `UrlBasedViewResolver`

See Also:

`AbstractUrlBasedView`

setAlwaysInclude

```
public void setAlwaysInclude(boolean alwaysInclude)
```

Specify whether to always include the view rather than forward to it.

Default is "false". Switch this flag on to enforce the use of a Servlet include, even if a forward would be possible.

See Also:

`InternalResourceView.setAlwaysInclude(boolean)`

buildView

```
protected AbstractUrlBasedView buildView(String viewName)
                                  throws Exception
```

Description copied from class: `UrlBasedViewResolver`
Creates a new View instance of the specified view class and configures it. Does *not* perform any lookup for pre-defined View instances.

Spring lifecycle methods as defined by the bean container do not have to be called here; those will be applied by the `loadView` method after this method returns.

Subclasses will typically call `super.buildView(viewName)` first, before setting further properties themselves. `loadView` will then apply Spring lifecycle methods at the end of this process.

Overrides:

`buildView` in class `UrlBasedViewResolver`

Parameters:

`viewName - the name of the view to build`

Returns:

`the View instance`

Throws:

`Exception - if the view couldn't be resolved`

See Also:

`UrlBasedViewResolver.loadView(String, java.util.Locale)`