Legacy configuration

As eZ Publish legacy is part of eZ Publish 5, you might want to access its settings (i.e. not migrated extension). Regarding this, a **LegacyConfigResolver** has been implemented, allowing the access to the legacy settings.

LegacyConfigResolver

The `LegacyConfigResolver` works in the same way the main `ConfigResolver` presented in the configuration documentation. However, it has its own specification.

- **Parameter name** must be the combination of the INI block and parameter name (**`<iniGroupName>.<paramName>`**, e.g. `DebugSettings.DebugOutput`).
- **Namespace** is the **INI file name**, without .ini suffix (e.g. **image** for *image.ini*).
- **Scope** can still be a specific siteaccess to look into (still defaults to current siteaccess)

Example

```
// Inside a controller

$legacyResolver = $this->get( 'ezpublish_legacy.config.resolver' );
// Get [DebugSettings].DebugOutput from site.ini
$debugOutput = $legacyResolver->getParameter( 'DebugSettings.DebugOutput' );
// Get [ImageMagick].ExecutablePath from image.ini
$imageMagickPath = $legacyResolver->getParameter( 'ImageMagick.ExecutablePath',
'image' );
// Get [DatabaseSettings].Database from site.ini, for ezdemo_site_admin siteaccess
$databaseName = $legacyResolver->getParameter( 'DatabaseSettings.Database', 'site',
'ezdemo_site_admin' );

// Note that the examples above are also applicable for hasParameter().
```

Best practices for migration

It might happen that your eZ Publish application still relies on legacy extensions (yours or from the community) that have not been migrated yet to eZ Publish 5. In this case the **LegacyConfigResolver** can be useful, but the problem is that once the extension is migrated you will then need to change your code to use the main config resolver, which can be a bit tedious on large applications.

Actually, the *main* config resolver is set in such a way that **it chains the different available resolvers** (i.e. the main one and the legacy one). The **ChainConfigResolver** always checks the main `ConfigResolver` first, and if no parameter is found, it fallbacks to the `LegacyConfigResolver`.

So the example described above could be done in the following way:

```
// Inside a controller

// Get the (chain) ConfigResolver
$configResolver = $this->getConfigResolver();
// Get [DebugSettings].DebugOutput from site.ini
$debugOutput = $configResolver->getParameter( 'DebugSettings.DebugOutput' );
// Get [ImageMagick].ExecutablePath from image.ini
$imageMagickPath = $configResolver->getParameter( 'ImageMagick.ExecutablePath',
'image' );
// Get [DatabaseSettings].Database from site.ini, for ezdemo_site_admin siteaccess
$databaseName = $configResolver->getParameter( 'DatabaseSettings.Database', 'site',
'ezdemo_site_admin' );
```

The main difference is that **we don't explicitly call the LegacyConfigResolver**. The benefit is that once the settings are migrated to work with eZ Publish 5 (and assuming that these settings are named in the same way), there will be absolutely no change to do in your code! The **ChainConfigResolver** will seamlessly use the new configuration files.

Note for extension developers

Considering what was explained above, the best practice for extension migration regarding configuration is to keep a compatible format.

Example for SQLIImport

Legacy INI settings:

sqliimport.ini for ezdemo_site siteaccess

```
[ImportSettings]
# User ID of the XML Import Robot
# If none provided, site.ini [UserSettings]/AnonymousUserID will be used
RobotUserID=14

# Max time to wait for each Stream (in seconds)
StreamTimeout=50
```

Would become:

```
parameters:
    # Namespace is sqliimport (formerly sqliimport.ini), scope is defined to
ezdemo_site siteaccess
    sqliimport.ezdemo_site.ImportSettings.RobotUserID: 14
    sqliimport.ezdemo_site.ImportSettings.StreamTimeout: 50
```

Usage in a controller:

```
// Assuming that current siteaccess is ezdemo_site
// The following code will work regardless using the legacy extension or the migrated
one.
$resolver = $this->getConfigResolver();
$robotUserId = $resolver->getParameter( 'ImportSettings.RobotUserID', 'sqliimport' );
$streamTimeout = $resolver->getParameter( 'ImportSettings.StreamTimeout', 'sqliimport'
);
```