
Article

[Lorenzo Scalese](#) · Mars 18, 2022 6m de lecture

[Open Exchange](#)

Configuration de l'environnement avec config-api

Chers développeurs,

Écrire un script pour le déploiement d'une application peut être très intéressant pour assurer un déploiement rapide sans rien oublier. config-api est une bibliothèque pour aider les développeurs à écrire des scripts de configuration basés sur un document JSON.

Fonctions implémentées :

- Définir les paramètres du système.
- Définir les paramètres de sécurité.
- Activer les services.
- Configurer les espaces de noms, les bases de données, le mapping.
- Exporter la configuration existante.
- Toutes les fonctionnalités sont exposées avec une API RESTful.

Cette bibliothèque se concentre sur la configuration d'IRIS pour faciliter le déploiement des applications. Ainsi, config-api n'importe/ne compile pas de code, considérant que cela devrait être le rôle du module d'installation de votre application ou du registre du client. config-api pourrait être utilisé avec le client ZPM pour configurer les paramètres IRIS lors du déploiement du module, nous apprendrons comment combiner cette bibliothèque avec ZPM dans un autre article.

Installer

```
zpm "install config-api"
```

Si vous n'êtes pas un utilisateur de ZPM, téléchargez la dernière version au format XML avec les dépendances [sur la page de publication](#), importez et compilez.

Première étape

Écrivons un simple document JSON de configuration pour définir quelques paramètres du système. Dans ce premier document, nous :

- Activons le gel du journal en cas d'erreur.
- Définissons la taille limite du journal à 256 Mo.
- Définissons SystemMode sur développement.
- Augmentons le locksiz.
- Augmentons le LockThreshold.

```
Set config = {  
  "Journal": {  
    "FreezeOnError":1,  
    "FileSizeLimit":256  
  }  
}  
/* Service class Api.Config.Journal */
```

```

    },
    "SQL": {
        "LockThreshold" : 2500
    },
    "config": {
        "locksiz" : 33554432
    },
    "Startup":{
        "SystemMode" : "DEVELOPMENT"
    }
}
Set sc = ##class(Api.Config.Services.Loader).Load(config)

```

structure du document JSON de configuration

Les clés de premier niveau (Journal, SQL ,config, Startup) sont liées aux classes de l'espace de noms %SYS (en utilisant une classe intermédiaire du paquet Api.Config.Services). Cela signifie que Journal prend en charge toutes les propriétés disponibles dans [Config.Journal](#), SQL, toutes les propriétés dans [Config.SQL](#), etc...

Sortie :

```

2021-03-31 18:31:54 Start load configuration
2021-03-31 18:31:54 {
  "Journal":{
    "FreezeOnError":1,
    "FileSizeLimit":256
  },
  "SQL":{
    "LockThreshold":2500
  },
  "config":{
    "locksiz":33554432
  },
  "Startup":{
    "SystemMode":"DEVELOPMENT"
  }
}
2021-03-31 18:31:54 * Journal
2021-03-31 18:31:54 + Update Journal ... OK
2021-03-31 18:31:54 * SQL
2021-03-31 18:31:54 + Update SQL ... OK
2021-03-31 18:31:54 * config
2021-03-31 18:31:54 + Update config ... OK
2021-03-31 18:31:54 * Startup
2021-03-31 18:31:54 + Update Startup ... OK

```

Astuce : la méthode Load est compatible avec un argument de type chaîne, dans ce cas, la chaîne doit être un nom de fichier vers un document de configuration JSON (l'objet stream est également autorisé).

Créer un environnement d'application

Dans cette section, nous écrivons un document de configuration pour créer :

- Un espace de nom « MYAPP ».

- 4 bases de données (MYAPPDATA, MYAPPCODE, MYAPPARCHIVE, MYAPPLLOG)
- 1 application Web CSP (/csp/zwebapp).
- 1 application Web REST (/csp/zrestapp).
- Configuration du mappage des globales.

```

Set config = {
  "Defaults":{
    "DBDIR" : "${MGRDIR}",
    "WEBAPPDIR" : "${CSPDIR}",
    "DBDATA" : "${DBDIR}myappdata/",
    "DBARCHIVE" : "${DBDIR}myapparchive/",
    "DBCOD" : "${DBDIR}myappcode/",
    "DBLOG" : "${DBDIR}myapplog/"
  },
  "SYS.Databases":{
    "${DBDATA}" : {"ExpansionSize":128},
    "${DBARCHIVE}" : {},
    "${DBCOD}" : {},
    "${DBLOG}" : {}
  },
  "Databases":{
    "MYAPPDATA" : {
      "Directory" : "${DBDATA}"
    },
    "MYAPPCODE" : {
      "Directory" : "${DBCOD}"
    },
    "MYAPPARCHIVE" : {
      "Directory" : "${DBARCHIVE}"
    },
    "MYAPPLLOG" : {
      "Directory" : "${DBLOG}"
    }
  },
  "Namespaces":{
    "MYAPP": {
      "Globals":"MYAPPDATA",
      "Routines":"MYAPPCODE"
    }
  },
  "Security.Applications": {
    "/csp/zrestapp": {
      "DispatchClas" : "my.dispatch.class",
      "Namespace" : "MYAPP",
      "Enabled" : "1",
      "AuthEnabled": "64",
      "CookiePath" : "/csp/zrestapp/"
    },
    "/csp/zwebapp": {
      "Path": "${WEBAPPDIR}zwebapp/",
      "Namespace" : "MYAPP",
      "Enabled" : "1",
      "AuthEnabled": "64",
      "CookiePath" : "/csp/zwebapp/"
    }
  },
  "MapGlobals":{
    "MYAPP": [{
      "Name" : "Archive.Data",

```

```

        "Database" : "MYAPPARCHIVE"
    }, {
        "Name" : "App.Log",
        "Database" : "MYAPPLLOG"
    }
]
}
}
Set sc = ##class(Api.Config.Services.Loader).Load(config)

```

Sortie :

```

2021-03-31 20:20:07 Start load configuration
2021-03-31 20:20:07 {
  "SYS.Databases":{
    "/usr/irissys/mgr/myappdata/":{
      "ExpansionSize":128
    },
    "/usr/irissys/mgr/myapparchive/":{
    },
    "/usr/irissys/mgr/myappcode/":{
    },
    "/usr/irissys/mgr/myapplog/":{
    }
  },
  "Databases":{
    "MYAPPDATA":{
      "Directory":"/usr/irissys/mgr/myappdata/"
    },
    "MYAPPCODE":{
      "Directory":"/usr/irissys/mgr/myappcode/"
    },
    "MYAPPARCHIVE":{
      "Directory":"/usr/irissys/mgr/myapparchive/"
    },
    "MYAPPLLOG":{
      "Directory":"/usr/irissys/mgr/myapplog/"
    }
  },
  "Namespaces":{
    "MYAPP":{
      "Globals":"MYAPPDATA",
      "Routines":"MYAPPCODE"
    }
  },
  "Security.Applications":{
    "/csp/zrestapp":{
      "DispatchClas":"my.dispatch.class",
      "Namespace":"MYAPP",
      "Enabled":"1",
      "AuthEnabled":"64",
      "CookiePath":"/csp/zrestapp/"
    },
    "/csp/zwebapp":{
      "Path":"/usr/irissys/csp/zwebapp/",
      "Namespace":"MYAPP",
      "Enabled":"1",
      "AuthEnabled":"64",

```

```
    "CookiePath":"/csp/zwebapp/"
  },
  "MapGlobals":{
    "MYAPP":[
      {
        "Name":"Archive.Data",
        "Database":"MYAPPARCHIVE"
      },
      {
        "Name":"App.Log",
        "Database":"MYAPPLLOG"
      }
    ]
  }
}

2021-03-31 20:20:07 * SYS.Databases
2021-03-31 20:20:07 + Create /usr/irissys/mgr/myappdata/ ... OK
2021-03-31 20:20:07 + Create /usr/irissys/mgr/myapparchive/ ... OK
2021-03-31 20:20:07 + Create /usr/irissys/mgr/myappcode/ ... OK
2021-03-31 20:20:07 + Create /usr/irissys/mgr/myapplog/ ... OK
2021-03-31 20:20:07 * Databases
2021-03-31 20:20:07 + Create MYAPPDATA ... OK
2021-03-31 20:20:07 + Create MYAPPCODE ... OK
2021-03-31 20:20:07 + Create MYAPPARCHIVE ... OK
2021-03-31 20:20:07 + Create MYAPPLLOG ... OK
2021-03-31 20:20:07 * Namespaces
2021-03-31 20:20:07 + Create MYAPP ... OK
2021-03-31 20:20:07 * Security.Applications
2021-03-31 20:20:07 + Create /csp/zrestapp ... OK
2021-03-31 20:20:07 + Create /csp/zwebapp ... OK
2021-03-31 20:20:07 * MapGlobals
2021-03-31 20:20:07 + Create MYAPP Archive.Data ... OK
2021-03-31 20:20:07 + Create MYAPP App.Log ... OK
```

Ça marche ! La configuration est correctement chargée.

Dans le prochain article, nous apprendrons comment utiliser config-api avec ZPM pour déployer votre application.

L'application est présentée pour le concours, [votez pour elle](#) si vous l'aimez ;-).

Merci de votre lecture.

[#DevOps](#) [#Déploiement](#) [#InterSystems](#) [IRIS](#)

[Voir l'application sur InterSystems Open Exchange](#)

URL de la source:<https://fr.community.intersystems.com/post/configuration-de-lenvironnement-avec-config-api>