Settings Subdomain

Table of Contents

This module (`isis-module-settings`) provides the ability to persist application- and user- configuration settings.

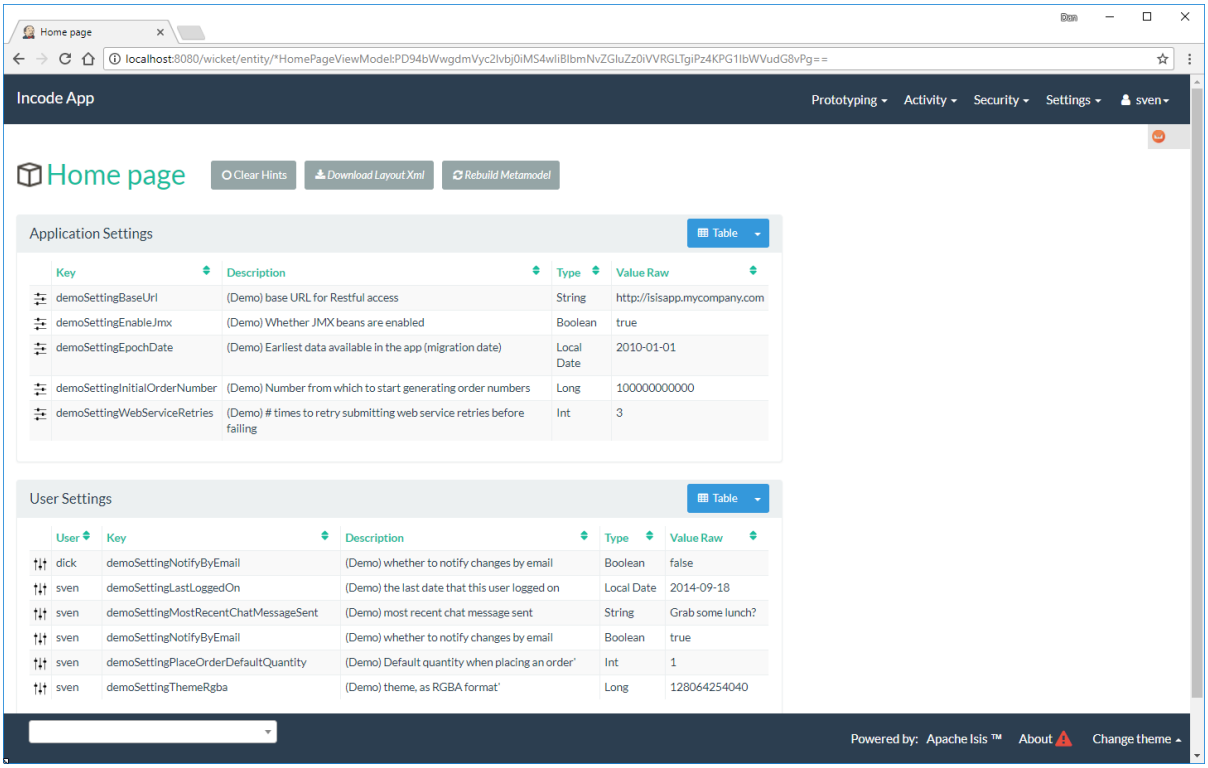
With `ApplicationSettingsService` these settings have global scope; for the `UserSettingsService` the settings are scoped per user.

The settings themselves are keyed by a simple string, and can store any of boolean, String, int, long and `LocalDate`. The implementation persists these values in a single raw format, but the API exposed by the services aims to be type-safe.

Screenshots

The module's functionality can be explored by running the quickstart with example usage using the `org.incode.domainapp.example.app.modules.ExampleDomDomSettingsAppManifest`.

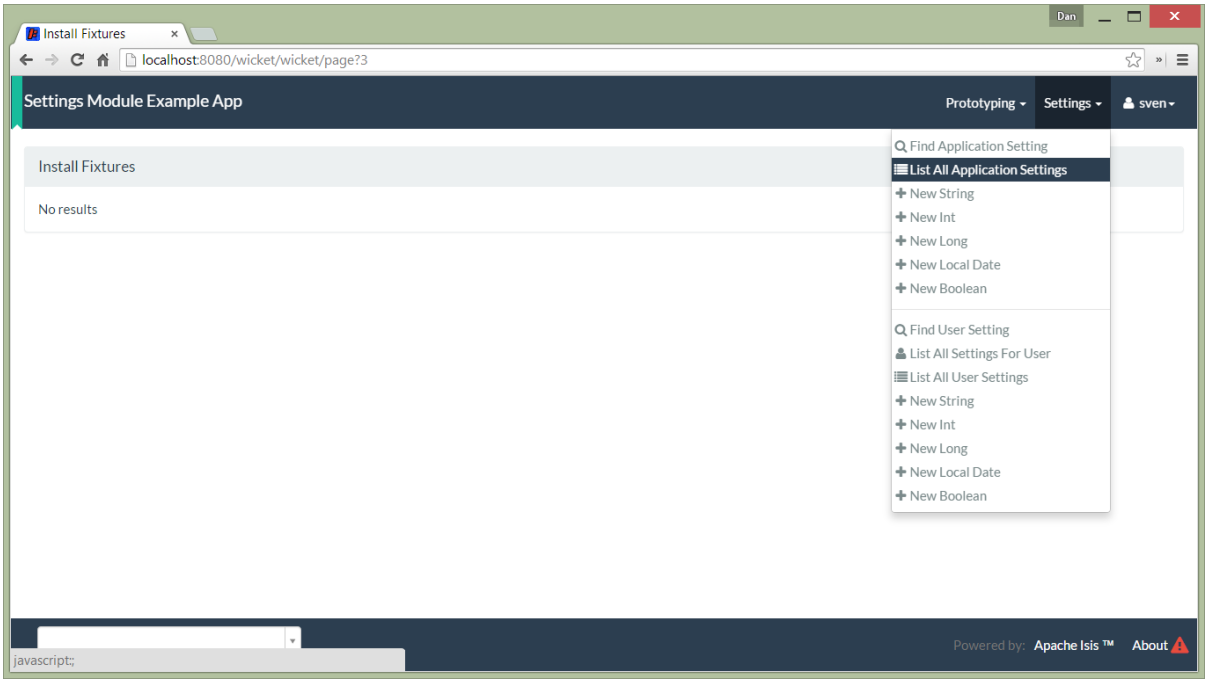
A home page is displayed when the app is run:



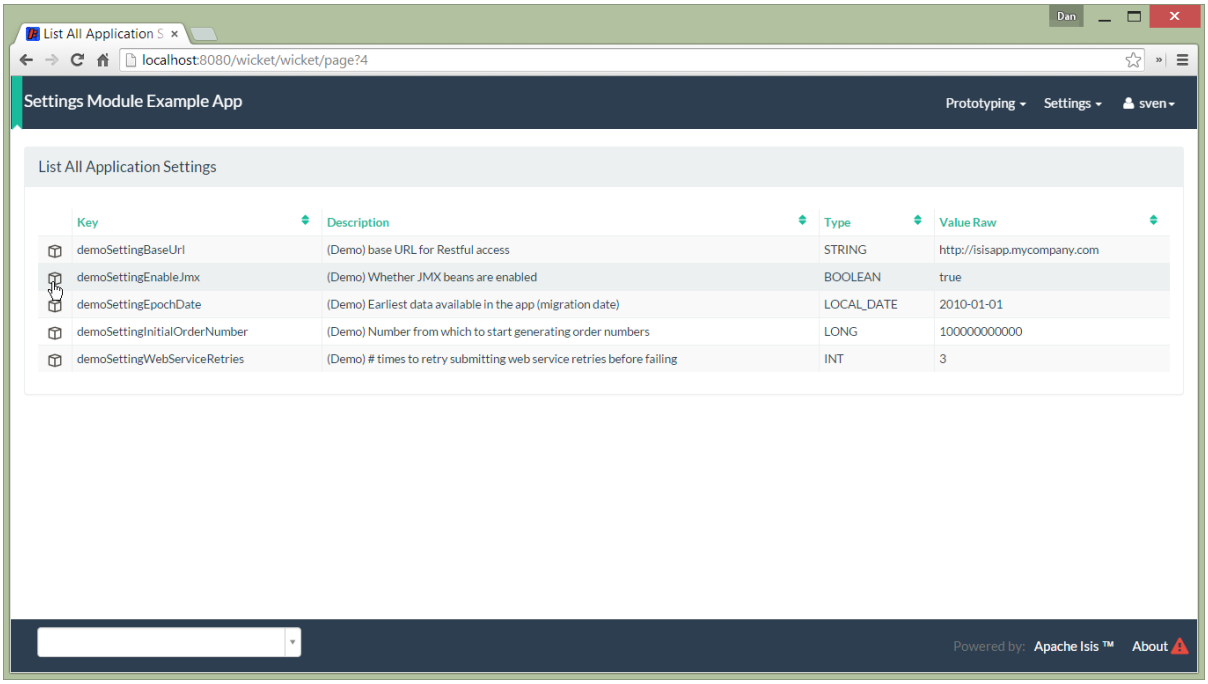
The remaining screenshots below **do** demonstrate the functionality of this module, but are out of date in that they are taken from the original isisaddons/incodehq module (prior to being amalgamated into the incode-platform).

App Settings

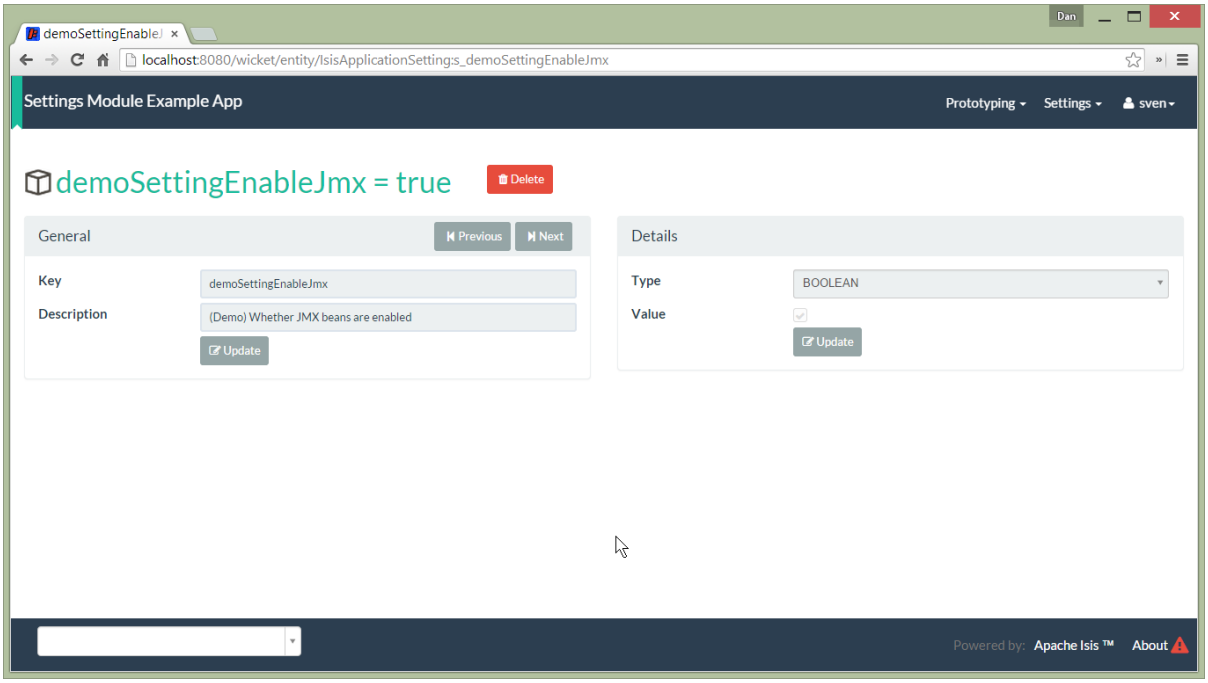
List all (demo) application settings:



listed in a table:

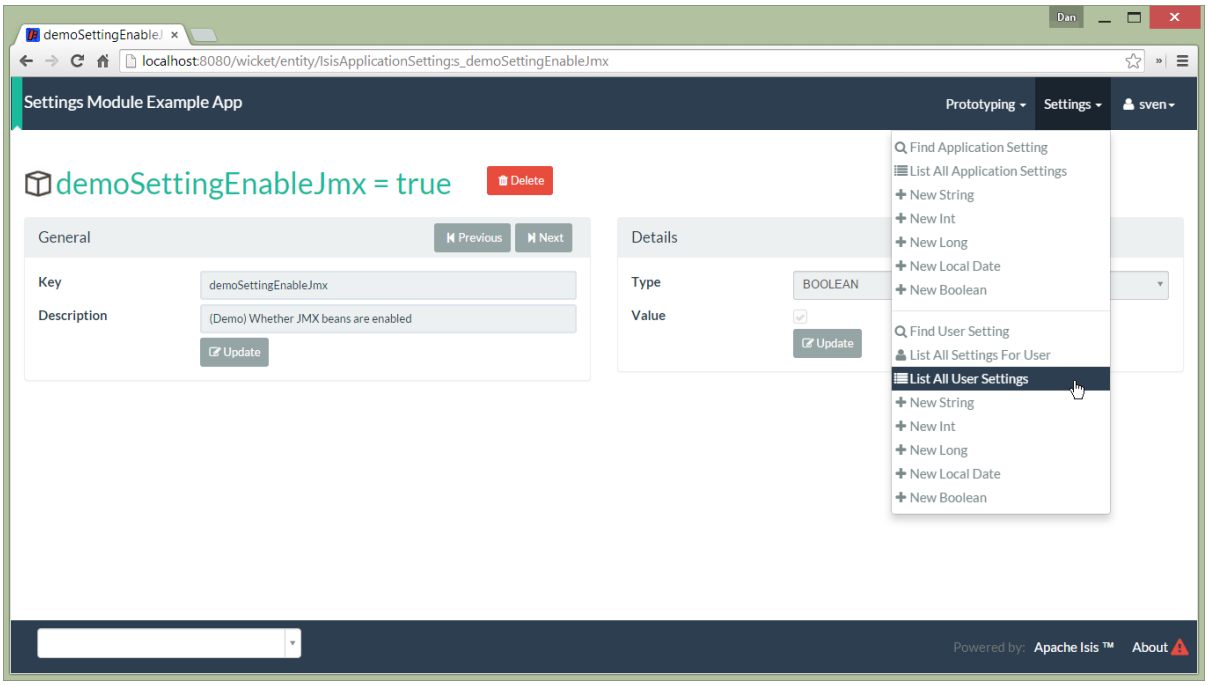


and inspect detail:

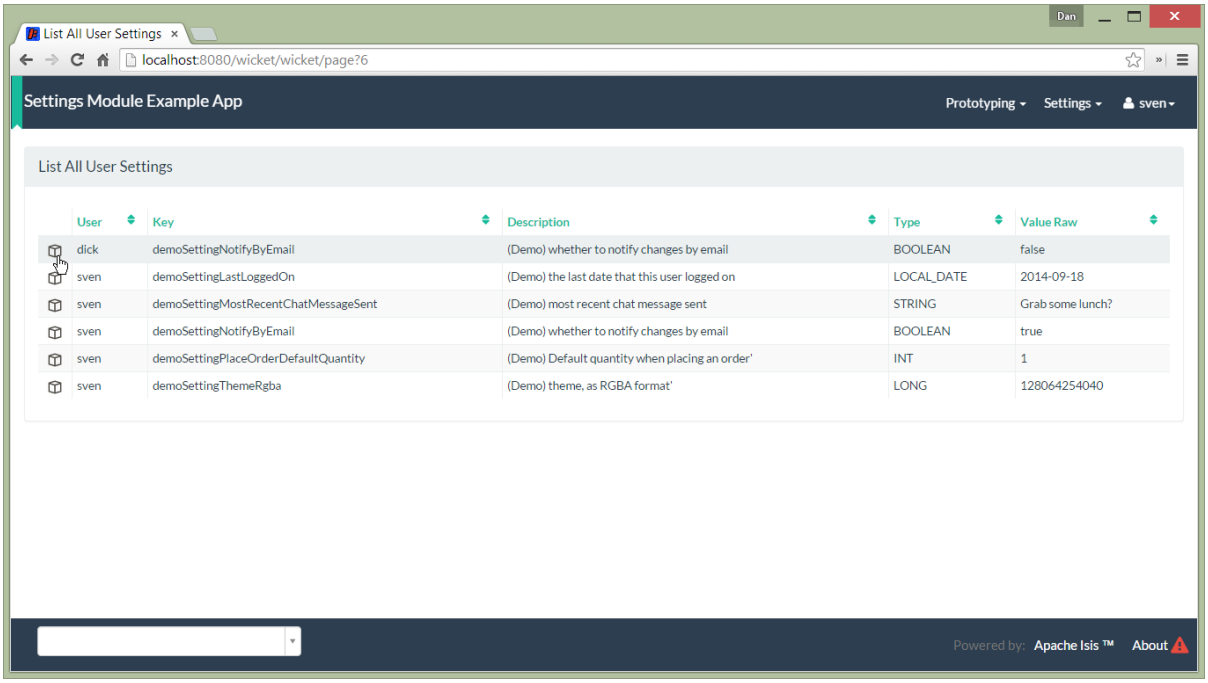


User Settings

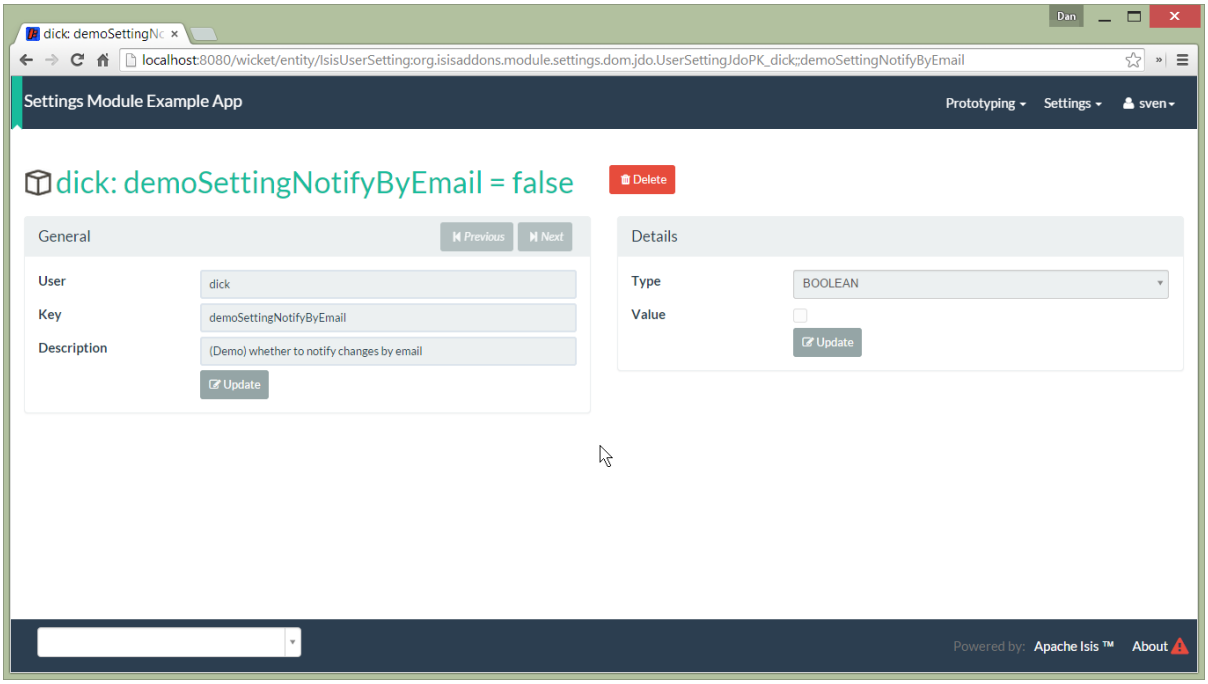
List all (demo) user settings:



listed in a table:



and inspect detail:



How to configure/use

Classpath

Update your classpath by adding this dependency in your dom project's `pom.xml`:

```
<dependency>
    <groupId>org.incode.example.settings</groupId>
    <artifactId>incode-example-settings-dom</artifactId>
</dependency>
```

Check for later releases by searching Maven Central Repo.

Bootstrapping

In the `AppManifest`, update its `getModules()` method, eg:

```
@Override
public List<Class<?>> getModules() {
    return Arrays.asList(
            ...
            org.incode.example.settings.SettingsModule.class,
            ...
    );
}
```

API

ApplicationSettingsService and ApplicationSettingsServiceRW

The module defines two interfaces for application settings. The first, `ApplicationSettingsService`, provides read-only access:

```
public interface ApplicationSettingsService {
    ApplicationSetting find(String key);
    List<ApplicationSetting> listAll();
}
```

The second, `ApplicationSettingsServiceRW`, extends the first and allows settings to be created:

```
public interface ApplicationSettingsServiceRW extends ApplicationSettingsService {
    ApplicationSetting newBoolean(String name, String description, Boolean
defaultValue);
    ApplicationSetting newString(String name, String description, String defaultValue
);
    ApplicationSetting newLocalDate(String name, String description, LocalDate
defaultValue);
    ApplicationSetting newInt(String name, String description, Integer defaultValue);
    ApplicationSetting newLong(String name, String description, Long defaultValue);
}
```

UserSettingsService and UserSettingsServiceRW

The module defines two interfaces for user settings. These are almost identical to the application settings above, the significant difference being each setting is additional identified by the username that 'owns' it.

The first interface, `UserSettingsService`, provides read-only access:

```
public interface UserSettingsService {
    UserSetting find(String user, String key);
    List<UserSetting> listAll();
    List<UserSetting> listAllFor(String user);
}
```

The second, `UserSettingsServiceRW`, extends the first and allows settings to be created:

```
public interface UserSettingsServiceRW extends UserSettingsService {
    UserSetting newBoolean(String user, String name, String description, Boolean
defaultValue);
    UserSetting newString(String user, String name, String description, String
defaultValue);
    UserSetting newLocalDate(String user, String name, String description, LocalDate
defaultValue);
    UserSetting newInt(String user, String name, String description, Integer
defaultValue);
    UserSetting newLong(String user, String name, String description, Long
defaultValue);
}
```

Implementation

The `ApplicationSettingsServiceJdo` implements `ApplicationSettingsServiceRW` (and therefore also `ApplicationSettingsService`).

Similarly, the `UserSettingsServiceJdo` implements `UserSettingsServiceRW` (and therefore also `UserSettingsService`).

Known issues

None known at this time.

Dependencies

Maven can report modules dependencies using:

```
mvn dependency:list -o -pl modules/dom/settings/impl -D excludeTransitive=true
```

which, excluding the Apache Isis modules, returns no direct compile/runtime dependencies.

The module *does* use icons from icons8.