Security and Encryption

Overview

The `org.openmrs.util.Security` class provides basic encryption and decryption methods for use in the API.

Single Direction Encryption or Hash Validation

The following public methods provide access to single direction encryption utilities:

```java
public static boolean hashMatches(String hashedPassword, String passwordToHash);

This is mostly helpful with password validation and checks against both SHA1 and SHA-512 + 128 character salt algorithms.
```

```java
public static String encodeString(String strToEncode);

The returned value is the parameter after being encoded using the OpenMRS default encryption (currently hardcoded to SHA-512).
```

```java
public static String getRandomToken();

This simply returns an encoded string using the current time in milliseconds plus a random long value.
```

Two Way Encryption

OpenMRS utilizes the AES/CBC/PKCS5Padding method for block cipher encryption and decryption. The initialization vector is an array of 16 bytes (typically random) and it will only properly encrypt or decrypt if paired with a specific secret key byte array. Following are the OpenMRS Constants involved:

```java
/**
 * Encryption properties; both vector and key are required to utilize a two-way encryption
 */
public static final String ENCRYPTION_CIPHER_CONFIGURATION = "AES/CBC/PKCS5Padding";
public static final String ENCRYPTION_KEY_SPEC = "AES";
public static final String ENCRYPTION_VECTOR_RUNTIME_PROPERTY = "encryption.vector";
public static final String ENCRYPTION_VECTOR_DEFAULT = "9wy8UNh1FCRCV5hMi3Ta3Q==";
public static final String ENCRYPTION_KEY_RUNTIME_PROPERTY = "encryption.key";
public static final String ENCRYPTION_KEY_DEFAULT = "dTyfGRzAICDwzjHBjuh==";
```

The encryption vector and key are necessary to form a reliable two way hash, and can be overridden by runtime properties.
public static String encrypt(String text, byte[] initVector, byte[] secretKey);
public static String encrypt(String text);
public static String decrypt(String text, byte[] initVector, byte[] secretKey);
public static String decrypt(String text);

These methods encrypt and decrypt text using provided or stored initialization vectors and secret keys. The most common API users should not have to provide initVector and secretKey; the methods requiring those values only do so for convenience in testing and special circumstances.

public static byte[] generateNewInitVector();
public static byte[] generateNewSecretKey();

The only time these methods should be used is during the initialization wizard's rendering of runtime properties, although they are available for public use.

⚠️ Warning
Changing the init vector and secret key values in the runtime properties file after data is encrypted will invalidate encrypted data!