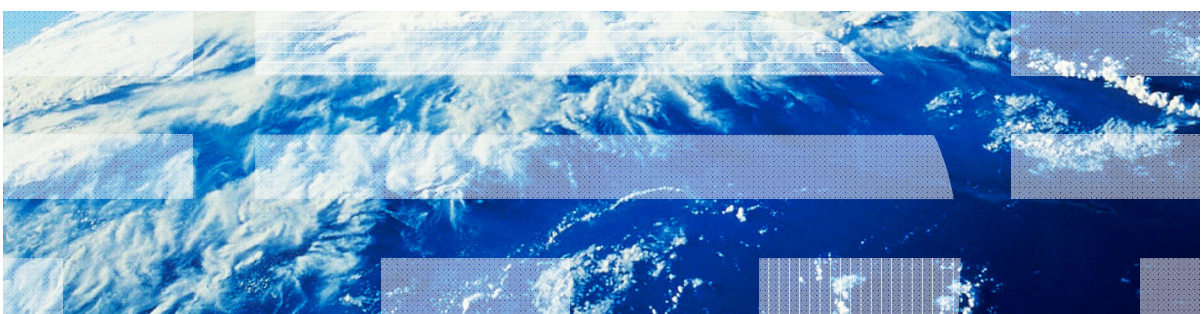
IBM Worklight V5.0.5
Getting Started

Module 7.12 – JSONStore with Security



18 January 2013

Trademarks

- IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Worklight is a trademark or registered trademark of Worklight, an IBM Company. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Other company products or service names may be trademarks or service marks of others.
- This document may not be reproduced in whole or in part without the prior written permission of IBM.

About IBM®

- See http://www.ibm.com/ibm/us/en/

Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

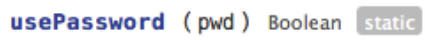
Advanced JSONStore Usage

- This module covers advanced tasks that you can perform on a local JSONStore collection, including encryption.

Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Encryption



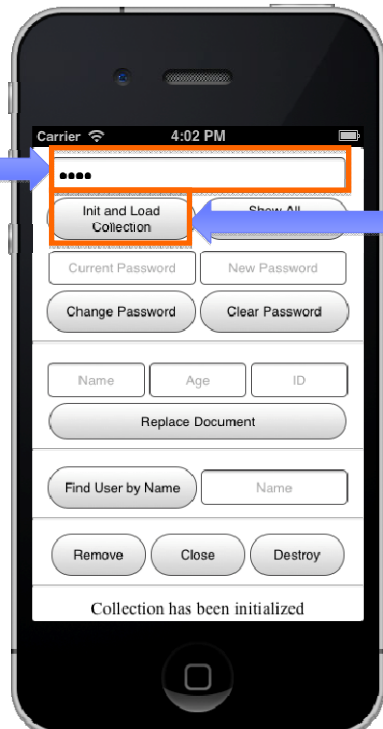
- You can use the JSONStore API to encrypt your entire local data store.
- To encrypt a data store, specify a password in the usePassword method.
- This sets the password that is used to generate keys to encrypt data that is stored locally on the device.
- Note: the store is encrypted at all times.

Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Encryption – Initialization

To enable encryption in the sample, specify a password.



Next, initialize and load the collection.

Encryption – Initialization

```
var pwd = passwordTag.val(),
pwdLength = pwd.length;

if (pwdLength > 0) {

    WL.JSONStore.usePassword(pwd);
    passwordTag.val('');
    pwd = null;

} else {

    WL.Logger.debug('Using NO Password, length: ' + pwdLength);
}

var initCollectionSuccessCallback = function () {
    logMessage('Collection has been initialized');
};

usersCollection = WL.JSONStore.initCollection(
    "users",
    usersSearchFields,
    {adapter: usersAdapterOptions,
    onSuccess: initCollectionSuccessCallback,
    onFailure: genericFailureCallback,
    load:true}),
```

Assigning a password

If a password value was given, pass it into the usePassword() method. This method encrypts the store by using the given key.

Encryption – Initialization

```
var pwd = passwordTag.val(),
pwdLength = pwd.length;

if (pwdLength > 0) {

    WL.JSONStore.usePassword(pwd);
    passwordTag.val('');
    pwd = null;

} else {

    WL.Logger.debug('Using NO Password, length: ' + pwdLength);
}

var initCollectionSuccessCallback = function () {
    logMessage('Collection has been initialized');
};

usersCollection = WL.JSONStore.initCollection(
    "users",
    usersSearchFields,
    {adapter: usersAdapterOptions,
    onSuccess: initCollectionSuccessCallback,
    onFailure: genericFailureCallback,
    load:true}),
```

onSuccess Callback

Define the OnSuccess callback. On success, display a meaningful message.

Encryption – Initialization

```
var pwd = passwordTag.val(),
pwdLength = pwd.length;

if (pwdLength > 0) {

    WL.JSONStore.usePassword(pwd);
    passwordTag.val('');
    pwd = null;

} else {

    WL.Logger.debug('Using NO Password, length:
}

var initCollectionSuccessCallback = function () {
    logMessage('Collection has been initialized');
};

usersCollection = WL.JSONStore.initCollection(
    "users",
    usersSearchFields,
    {adapter: usersAdapterOptions,
    onSuccess: initCollectionSuccessCallback,
    onFailure: genericFailureCallback,
    load:true}),
```

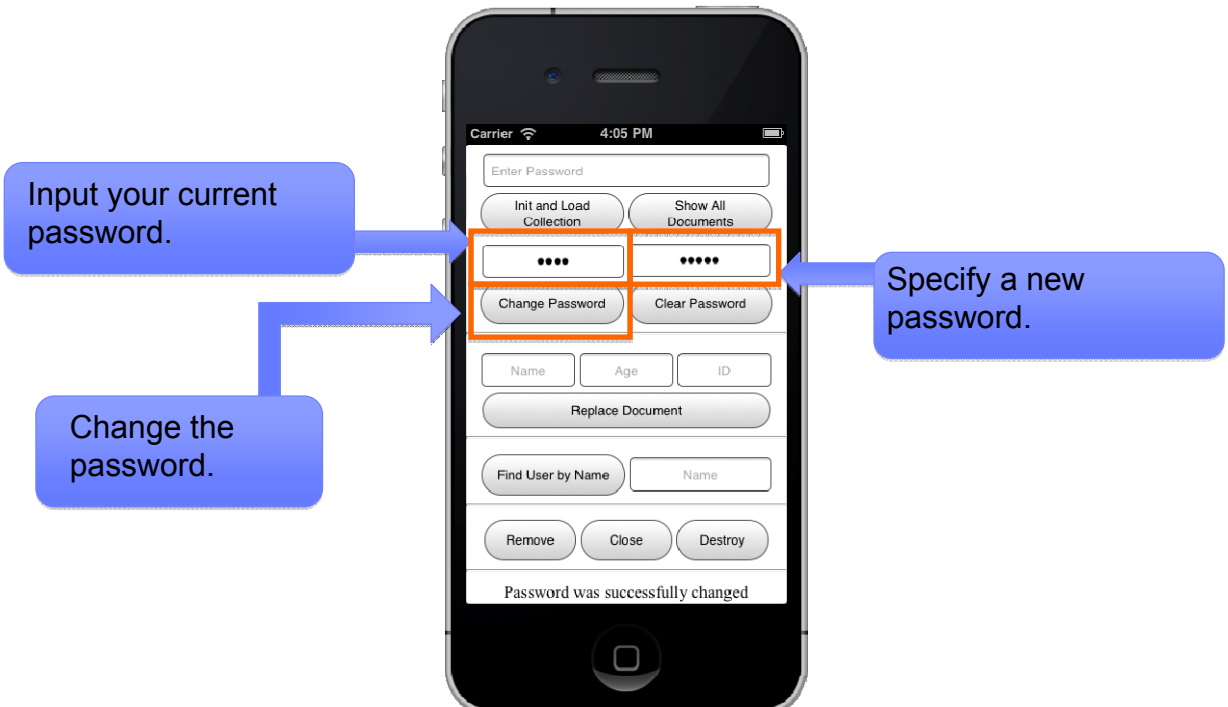
Init and Load

Finally, call the init function with the **load** parameter set to `true`. On success, the collection is created and encrypted using the specified key and the data that is retrieved from our adapter is loaded.

Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Encryption – Changing a Password



Encryption – Changing a Password

Process the input

Obtain the original password and the new one as specified in the inputs and assign them to variables.

```
WLJQ('button#change_password').bind('click', function () {

    WL.Logger.debug('Called button#change_password');

    var password_old = passwordOldTag.val(),
        password_new = passwordNewTag.val();

    if (password_old.length < 1 || password_new.length < 1 ||
        checkUndefOrNull(JSONStoreInstance)) {
        logMessage('You must enter your old password as well as the new one.');
    } else {

        var win =   function (data) {
            logMessage('Password was successfully changed : ' + data);
            passwordOldTag.val('');
            passwordNewTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        WL.JSONStore.changePassword(password_old,password_new,options);
    }
});//END Change Password
```

Encryption – Changing a Password

```
WLJQ('button#change_password').bind('click', function () {

    WL.Logger.debug('Called button#change_password');

    var password_old = passwordOldTag.val(),
        password_new = passwordNewTag.val();

    if (password_old.length < 1 || password_new.length < 1 ||
        checkUndefOrNull(JSONStoreInstance)) {
        logMessage('You must enter your old password as well as the new one.');
    } else {

        var win =   function (data) {
            logMessage('Password was successfully changed : ' + data);
            passwordOldTag.val('');
            passwordNewTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        WL.JSONStore.changePassword(password_old,password_new,options);
    }
});//END Change Password
```

Options

Define the OnSuccess callback. The variable **data** will contain a status code (defined in the JSONStore documentation).

Encryption – Changing a Password

```
WLJQ('button#change_password').bind('click', function () {

    WL.Logger.debug('Called button#change_password');

    var password_old = passwordOldTag.val(),
        password_new = passwordNewTag.val();

    if (password_old.length < 1 || password_new.length < 1 ||
        checkUndefOrNull(JSONStoreInstance)) {
        logMessage('You must enter your old password as well as
    } else {

        var win =   function (data) {
            logMessage('Password was successfully changed : ' +
            passwordOldTag.val('');
            passwordNewTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        WL.JSONStore.changePassword(password_old,password_new,options);
    }
});//END Change Password
```

Change the password

Finally, call the changePassword method with the old and new passwords as parameters. On success, the password is changed.

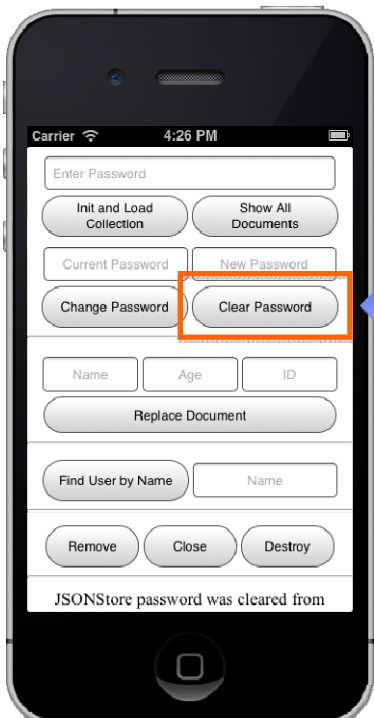
Note

This method is static so call it using the JSONStore instance.

Agenda

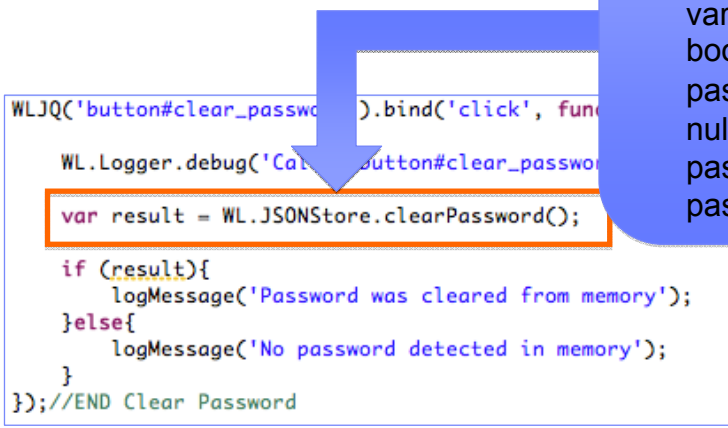
- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Encryption – Clearing a Password



Remove the password from memory if it exists.

Encryption – Clearing a Password



Clear the password

Call the clearPassword static method and assign the result to a variable. This method returns a boolean value. If `true`, the password in memory was set to null. Otherwise, there was no password in memory or the password could not be set to null.

Encryption – Clearing a Password

Check the result

Check the result variable and display a meaningful message.

```
WLJQ('button#clear_password').bind('click', function () {

    WL.Logger.debug('Called button#clear_password');

    var result = WL.JSONStore.clearPassword();

    if (result){
        logMessage('Password was cleared from memory');
    }else{
        logMessage('No password detected in memory');
    }
});//END Clear Password
```

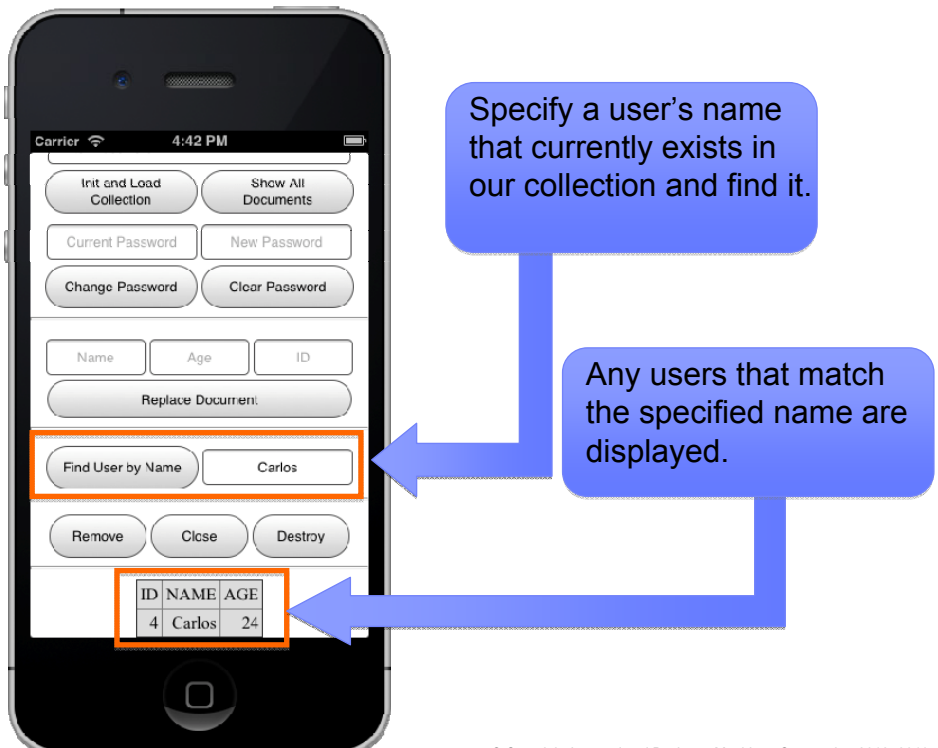
Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Enhance - Intro

- With the enhance method, you can add a function to a collection's prototype.
- You can create custom functions for tasks that might be performed frequently. To access these functions, use your collection instance to call them.
- For example, in the sample app a custom function for finding documents by a user’s name is created.

Enhance – UI



Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

Create a custom function

Define a custom function that creates a query and searches the `users` collection by user name. This function accepts a user name and an onSuccess callback as parameters.

Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

Add the function to collection prototype

Call the `enhance` method and pass in a name for the custom function, and the custom function itself.

The function is added to the collection prototype and is ready for use.

Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

onSucces Callback

Define an onSuccess callback so you can verify that the custom function was executed successfully.

Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

Calling the custom function

Finally, call the new function getUserByName by using the JSONSStore instance. Pass in the name to search for and an onSuccess callback as defined in the custom function. On success, you receive an array of matching documents which are then displayed in a table.

Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Documentify & Replace – UI

Specify a new name and age, together with the ID of the user in the collection that you want to replace.

Any users that match the specified ID in the local store are replaced.

Documentify & Replace – The Code

```
WLJQ('button#replace').bind('click', function () {

    WL.Logger.debug('Called button#replace');
    if (!checkColInit(usersCollection)) {return;}

    var name = nameTag.val(),
        id = idTag.val(),
        age = ageTag.val();

    if (id.length < 1 ) {
        logMessage('You must provide a valid value for ID');
    } else {
        var doc = WL.JSONStore.documentify(id, {name: name, age: age});

        var win =   function (data) {
            logMessage('Replaced ' + data + ' document.');
            //Clear Text Boxes
            nameTag.val('');
            ageTag.val('');
            idTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        usersCollection.replace(doc,options);
    }
});
```

Documentify

The replace method accepts a document object as a parameter so before you can call it you must construct a document object from the user input.

This can be achieved through the documentify method.

Documentify requires an id and JSON data and returns a JSONStore document object.

Documentify & Replace – The Code

```
WLJQ('button#replace').bind('click', function () {

    WL.Logger.debug('Called button#replace');
    if (!checkColInit(usersCollection)) {return;}

    var name = nameTag.val(),
        id = idTag.val(),
        age = ageTag.val();

    if (id.length < 1 ) {
        logMessage('You must provide a valid value for ID')
    } else {

        var doc = WL.JSONStore.documentify(id, {name: name,  age});

        var win =   function (data) {
            logMessage('Replaced ' + data + ' document.');
            //Clear Text Boxes
            nameTag.val('');
            ageTag.val('');
            idTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        usersCollection.replace(doc,options);
    }
});
```

Options

Define the `replace` method's OnSuccess callback. The variable **data** will contain the number of documents replaced.

Documentify & Replace – The Code

```
WLJQ('button#replace').bind('click', function () {

    WL.Logger.debug('Called button#replace');
    if (!checkColInit(usersCollection)) {return;}

    var name = nameTag.val(),
        id = idTag.val(),
        age = ageTag.val();

    if (id.length < 1 ) {
        logMessage('You must provide a valid value for ID');
    } else {

        var doc = WL.JSONStore.documentify(id, {name: name, age: age});

        var win =   function (data) {
            logMessage('Replaced ' + data + ' document.');
            //Clear Text Boxes
            nameTag.val('');
            ageTag.val('');
            idTag.val('');
        };

        var options = {onSuccess: win, on  lure: genericFailureCallback

        usersCollection.replace(doc,options);
    }
});
```

Replace the Document

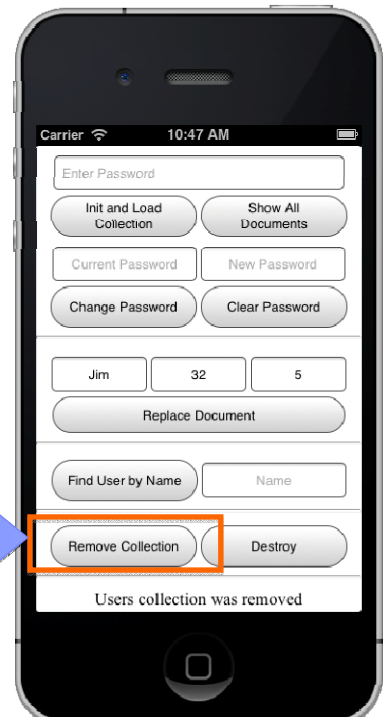
Finally, call the replace method and pass in the newly created document object and the options. You can verify that the document has been replaced by displaying all the documents in the collection by using the **Show all documents** button.

Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Collection Removal – UI

Remove the entire `users` collection. The local store still exists, either empty or with any other collections that have not been removed.



Collection Removal – The Code

Options

Define the onSuccess callback. The variable **data** contains a boolean value that indicates whether the `users` collection was removed.

```
WLJQ('button#remove_collection').bind('click', nction ()

    WL.Logger.debug('Called button#remove_colle ion');

    var win =  function (data) {
        if (data === 0) {
            logMessage('Users collection was removed');
        }else{
            logMessage('Users Collection was not removed');
        }
    };

    var options = {onSuccess: win, onFailure: genericFailureCallback};

    usersCollection.removeCollection(options);
});
```

Collection Removal – The Code

```
WLJQ('button#remove_collection').bind('click', function

    WL.Logger.debug('Called button#remove_collection');

    var win =  function (data) {
        if (data === 0) {
            logMessage('Users collection was removed');
        }else{
            logMessage('Users Collection was not removed');
        }
    };

    var options = {onSuccess: win, onFailure: genericFailureCallback};

    usersCollection.removeCollection(options);
});
```

Remove the Collection

Finally, we call our removeCollection method and pass in our options.

Agenda

- Advanced JSONStore Usage
  - Encryption
    - Initialization
    - Changing a Password
    - Clearing a Password
  - Enhance
  - Documentify & Replace
  - Collection Removal
- Sample

Sample

- The sample for this training module can be found in the Getting Started page of the IBM Worklight documentation website at
  - http://www.ibm.com/mobile-docs

Notices

- Permission for the use of these publications is granted subject to these terms and conditions.
- This information was developed for products and services offered in the U.S.A.
- IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
- IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
  - IBM Director of Licensing
    IBM Corporation
    North Castle Drive
    Armonk, NY 10504-1785
    U.S.A.
- For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
  - Intellectual Property Licensing
    Legal and Intellectual Property Law
    IBM Japan Ltd.
    1623-14, Shimotsuruma, Yamato-shi
    Kanagawa 242-8502 Japan
- **The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law**: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
- This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
- Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
- IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
- Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
- The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.
- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

- This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
- Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

  © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Support and comments

- For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:
  - http://www.ibm.com/mobile-docs
- **Support**
  - Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website at:
    - http://www.ibm.com/software/passportadvantage
  - If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook at:
    - http://www.ibm.com/support/handbook
- **Comments**
  - We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.
  - For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.
  - When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.
  - Thank you for your support.
  - Submit your comments in the IBM Worklight forums at:
    - https://www.ibm.com/developerworks/mobile/mobileforum.html
  - If you would like a response from IBM, please provide the following information:
    - Name
    - Address
    - Company or Organization
    - Phone No.
    - Email address

Thank You