Couchbase NoSQL Developer Workshop

ラボハンドブック

ラボ **1:** ユーザ登録と **K/V get** 処理

手順

このラボの目的は、K/V 取得操作を理解するために、新しいユーザーを登録し、顧客を取得することです。

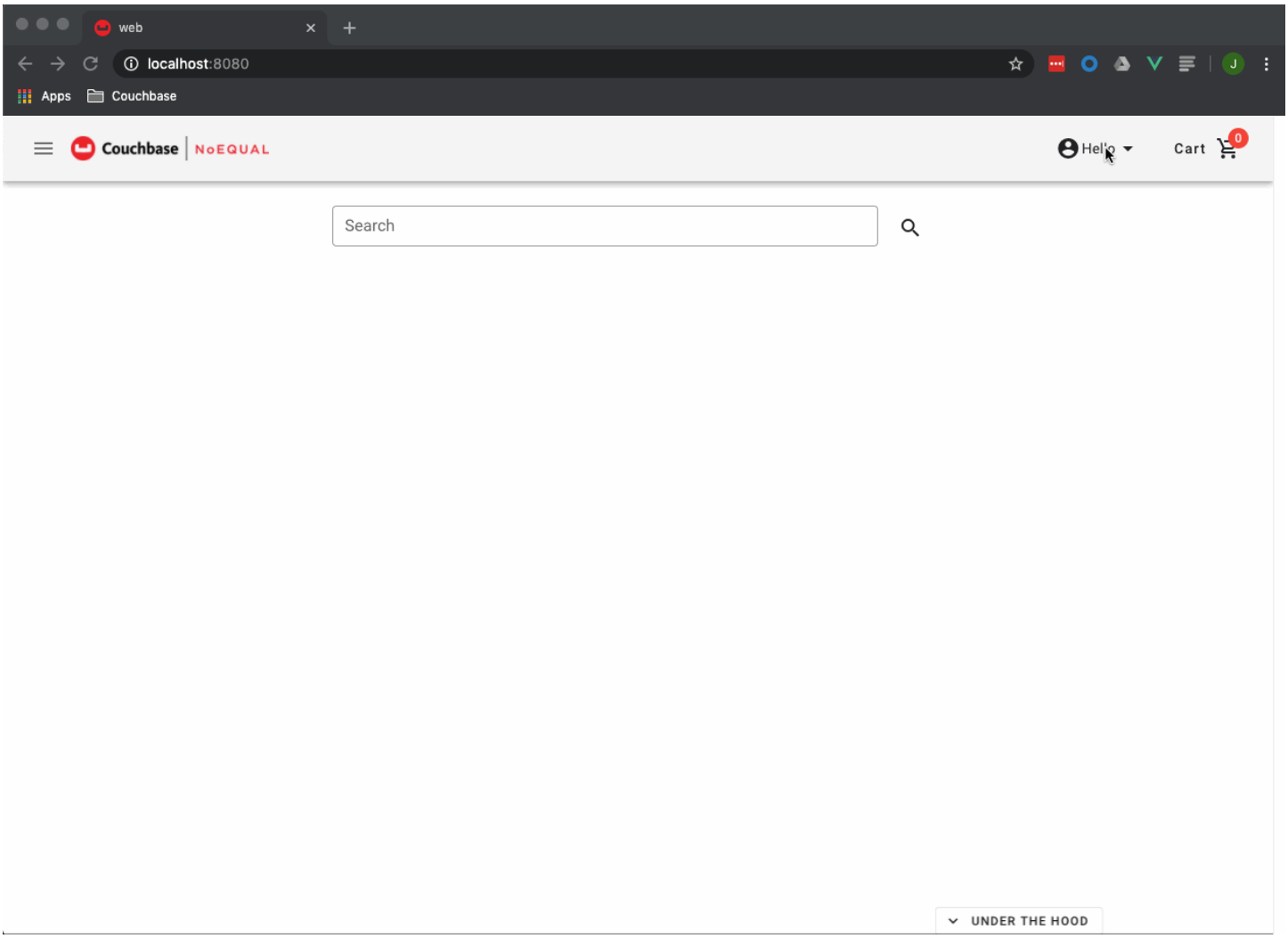
ステップ 1: 登録とログイン

Web UI と Node.js 3.x API コンテナの両方が実行されたら、新しいユーザーを登録する事ができます。 登録が成功すると、新しいユーザーでログインできるようになります。ログインの登録を完了するには、以下の手順に従います(手順のチュートリアルについては、後掲のクリップを参照してください)。

1. http://localhost:8080 に移動します。
2. 右上隅のユーザーアイコンの横にある*[Hello]*をクリックすると、ドロップダウンメニューが表示されます。
3. ドロップダウンメニューで [*Sign In*] をクリックし、Web UI がログイン画面にリダイレクトされます。
4. ログインページで、*ログイン*パネルの左下にある[Register]ボタンをクリックします。
5. すべてのテキストフィールドに値を入力します。
    a. First Name(名) - 任意の文字列が有効です
    b. Last Name(姓) - 任意の文字列が有効です
    c. Username(ユーザー名) - 任意の文字列が有効です
    d. Email(電子メール) -"example@domain.com"という形式の文字列
        i. 電子メールが本物かどうかの検証はありません
    e. Password(パスワード) - 任意の文字列が有効です
6. [REGISTER]ボタンをクリックします。
7. 登録完了後、Web UI は*ログイン*ページに直接リダイレクトされます。 ログインページの上部に「Successfully registered user. Please login」というメッセージが緑色で表示されます。登録した情報を用いて ログインしてください。
8. ステップ#5 で使用するユーザー名とパスワードを入力します。
9. *[LOGIN]*ボタンをクリックします。
10. ログイン後、Web UI はホームページにリダイレクトされます。 右上に "Hello {名}" が表示されます ({名} は、ステップ #5 で入力した値です)。

注 ***

画面上部に、青の四角で囲まれたメッセージとして、*"/user/checkForNewOrder operation not build yet* "と表示されます。これ は、エンドポイントが実装されていない事を示しています。 実装は後の演習で行われます。



ステップ 2: API にロジックを追加する

ドキュメント

- ドキュメントの取得

API リポジトリ ディレクトリで *repository.js* ファイルを開きます(付録の API プロジェクト構造を参照)。 *getCustomer( )*メソッドを検索します。 このラボの目的は、*getCustomer()*メソッドを編集し、特定の cus tomer ドキュメントの キーに基づいて顧客ドキュメントを返すキーバリュー (K/V) 操作を実行することです。

メモ ***

> K/V 取得操作の結果オブジェクトには、ドキュメントとメタデータが含まれています。 このラボおよび他のすべてのラボの要件として、ドキュメントの内容のみを返す必要があります。 したがって、取得操作が成功した後に、*result.value* を返すようにする必要があります。

getCustomer() 入力:

- 顧客 ID : 注文ドキュメント のドキュメント キー
- コールバック

getCustomer() 出力:

- エラー オブジェクト (該当する場合)
- カスタマー ドキュメント:サンプルの顧客ドキュメントの付録を参照してください。

*getCustomer()*メソッドの実装については、以下のコード スニペットを参照してください。
注意: 「NOP」 コード行 (“return "NOP";” など) を新しいコードに置き換えます。

```
0:  async getCustomer(customerId) {
1:    try {
2:      /**
3:       * Lab 1:  K/V operation - Get
4:       *  1.  Get customer:  bucket.get(key)
5:       */
6:      let result = await this.collection.get(customerId);
7:      return { customer: result ? result.value : null, error: null };
8:    } catch (err) {
9:      //Optional - add business logic to handle error types
10:     outputMessage(err, "repository.js:getCustomer() - error:");
11:     return { customer: null, error: err };
12:   }
13: }
```

コードに関する注意事項:

- aysnc/await 構文を使用します。
- 6 行目: K/V 操作は、3.x SDK の Collection レベルで行われます。
    - K/V は操作パラメータを取得します。
        - ドキュメント キー
- 7 行目: エラーがない場合は結果の値のみを返します。 ラボでは、ドキュメントのコンテンツを返すだけです。
- *outputMessage()*: コンソールに情報を簡単に出力するためのヘルパーメソッドは、/library ディレクトリにあります (付録で詳しく説明されている API のプロジェクト構造を参照してください)
- try/catch: err オブジェクトの処理は、意図的に汎用的な方法で行われてます。 ラボ参加者は、エラー処理のさまざまな方法をテストするために、目的に応じてロジックを追加することができます。

完了したら、*repository.js* ファイルが保存されていることを確認します。 API *Docker* コンテナは API の作業ディレクトリにマップされるため、API コードに対して行われたすべての更新はコンテナに反映される必要があります。 コンテナーの状態の詳細については、*docker logs api* コマンドを使用します。
ユーザープロファイル画面を表示するには、以下の手順に従います(手順のウォークスルーについては、後掲のクリップを参照してください)。

1. http://localhost:8080 に移動
2. 右上隅にある、ユーザーアイコンと「*Hello {名}*」をクリックすると、ドロップダウンメニューが表示されます。
    a. ログインしていない場合:
        i. ドロップダウンメニューで [*Sign In*] をクリックします。ログイン画面にリダイレクトされます。

ii. ユーザー名とパスワードの資格情報を入力します。
iii. [*LOGIN*] をクリックします。
iv. ログイン後、ホーム画面にリダイレクトされます。 ステップ#2 の先頭に戻ります。

3. ドロップダウンメニューで [*User Profile*] をクリックし、ユーザー プロファイルにリダイレクトします。