Asia Developer Academy

API Authentication with Passport JS

What is Passport.js?

Passport.js is a module acts as a middleware to handle authentication in Node.js.

It can be used as an authentication using different methods including basic authentication using username and password, Twitter, Facebook and many more.

Features of Passport.js

- 140+ authentication strategies
- Single sign-on with OpenID and OAuth
- Easily handle success and failure
- Supports persistent sessions
- Dynamic scope and permissions
- Pick and choose required strategies
- Implement custom strategies
- Does not mount routes in application
- Lightweight code base

1.Revision: Creating User

1) Create a User Scheme with the following fields and validations:
    a) Username: unique, required
    b) Password: required
2) Import the User into the controller.
3) Create an API to register and login

2.Encrypting the password.

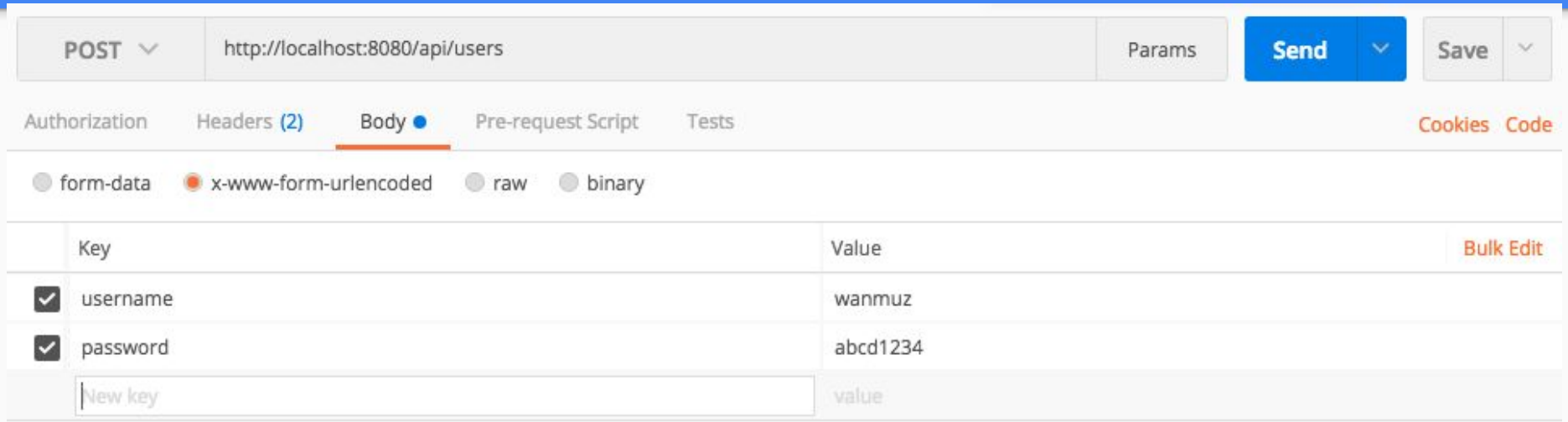
We are using bcrypt-node module to encrypt the password from the user whenever it is created.

Add in **bcrypt-node** into your project.

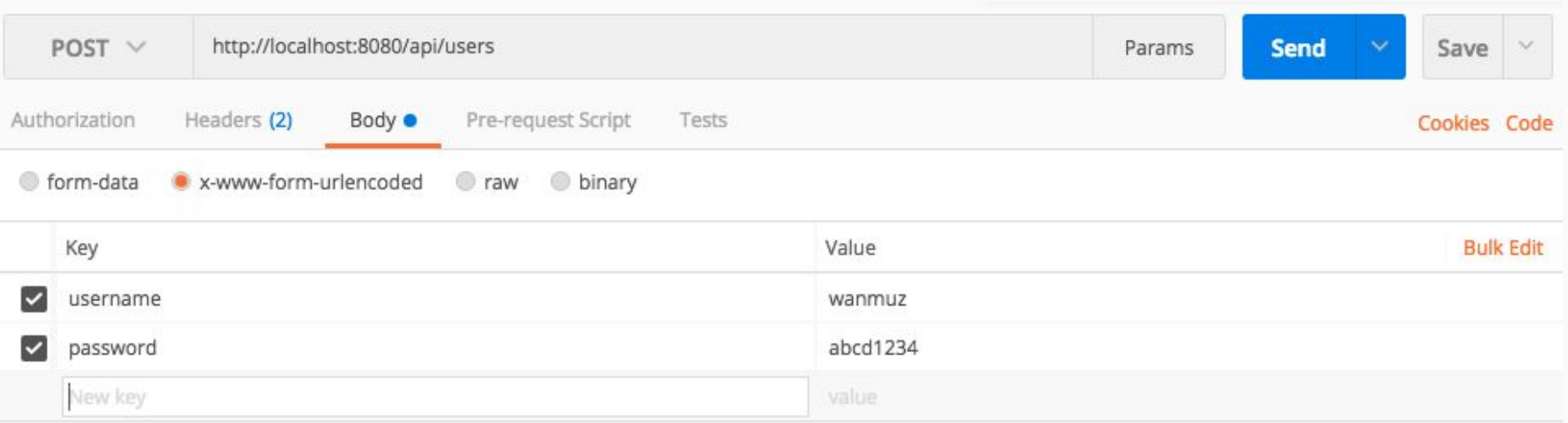
3. Add a pre-save in UserSchema.

```
UserSchema.pre('save', function(callback) {
  var user = this;

  // Break out if the password hasn't changed
  if (!user.isModified('password')) return callback();

  // Password changed so we need to hash it
  bcrypt.genSalt(5, function(err, salt) {
    if (err) return callback(err);

    bcrypt.hash(user.password, salt, null, function(err, hash) {
      if (err) return callback(err);
      user.password = hash;
      callback();
    });
  });
});
```

4. Test on Postman



5. Response From Postman



6. Installing passport and passport-http

Install the following module:

- passport
- passport-http

7. Edit user.js to add verifyPassword method

```
UserSchema.methods.verifyPassword = function(password, cb) {
  bcrypt.compare(password, this.password, function(err, isMatch) {
    if (err) return cb(err);
    cb(null, isMatch);
  });
};
```

8. Create a new Controller: auth.js

```
var passport = require('passport');
var BasicStrategy = require('passport-http').BasicStrategy;
var User = require('../models/user');


exports.isAuthenticated = passport.authenticate('basic', { session :
false });
```

9. Create a new Controller: auth.js

```
var passport = require('passport');
var BasicStrategy = require('passport-http').BasicStrategy;
var User = require('../models/user');


exports.isAuthenticated = passport.authenticate('basic', { session :
false });
```

```
passport.use(new BasicStrategy(
  function(username, password, callback) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return callback(err); }

      // No user found with that username
      if (!user) { return callback(null, false); }

      // Make sure the password is correct
      user.verifyPassword(password, function(err, isMatch) {
        if (err) { return callback(err); }

        // Password did not match
        if (!isMatch) { return callback(null, false); }

        // Success
        return callback(null, user);
      });
    });
  }
));
```

10. Include the file in server.js

```
var passport = require('passport');
var authController = require('./controllers/auth');



app.use(passport.initialize())
```

11. Modify the restricted Controller

```
router.route('/places’)
  .post(authController.isAuthenticated, placeController.postPlaces)
```