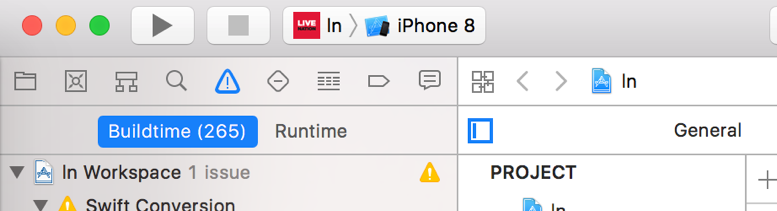
Client Side Instructions

1. Open the file “IBM-EIS.xcodeproj” using Xcode
2. In the top right side of Xcode, make sure a simulator target is chosen (e.g., “iPhone 8”) and click the Run (triangle) button to start the application



Adding employee ‘PUT’ capability into EIS Cluster Service

1. Add PUT route to *server-cluster.js* file

**//add a new record in the DB**

**app.put('/employees', employeesCtrl.updateEmployee);**

1. Add ‘*updateEmployee’* method to module export in *employees\_controller.js* file

**exports. updateEmployee= function updateEmployee (req, res) {**

**// blank space for now**

**}**

1. Check for existing employee id

exports. updateEmployee= function updateEmployee (req, res) {

**var missing\_field\_flag = false;**

**var warning\_msg;**

**var eid=req.params.id || req.query.id;**

**Employee.findOne({'\_id': eid}, function findOneEmployee(err, employee) {**

**// blank space for now**

**}**

}

1. Add error handling for employee lookup

…

Employee.findOne({'\_id': eid}, function findOneEmployee(err, employee) {

**if (err) {**

**console.log('\*\*\* Internal Error while retrieving an employee record:');**

**console.log(err);**

**sendResponse(res, 500, {**

**message: 'findById failed. Internal Server Error' },**

**'employee');**

**return;**

**}**

**if (!employee) {**

**sendResponse(res, 200, {**

**message: 'No records found'**

**}, 'employee');**

**return;**

**}**

**// continue here…**

}

…

1. Store transmitted employee data in a temporary objects

…

if (!employee) {

sendResponse(res, 200, {

message: 'No records found'

}, 'employee');

return;

}

**var employeeObj = req.body.employee;**

**var addressObj = req.body.address;**

**var compensationObj = req.body.compensation;**

**var familyObj = req.body.family;**

**var healthObj = req.body.health;**

**var photoObj = req.body.photo;**

1. Check for missing values in transmitted data

…

var healthObj = req.body.health;

var photoObj = req.body.photo;

**//zipcode and last\_name are required fields**

**if(!employeeObj || !addressObj || !compensationObj || !familyObj || !healthObj || !photoObj){**

**sendJSONResponse(res, 400, {**

**message: 'updateEmployee: Missing employee input'});**

**return;**

**}**

**if(!employeeObj.first\_name || !employeeObj.last\_name){**

**sendJSONResponse(res, 400, {**

**message: 'updateEmployee: Missing employee input'});**

**return;**

**}**

**if(!addressObj.zipcode){**

**sendJSONResponse(res, 400, {**

**message: 'updateEmployee: Missing employee input'});**

**return;**

**}**

1. Assign new values to employee object

…

if(!addressObj.zipcode){

sendJSONResponse(res, 400, {

message: 'updateEmployee: Missing employee input'});

return;

}

**employee.phone = employeeObj.phone;**

**employee.first\_name = employeeObj.first\_name;**

**employee.last\_name = employeeObj.last\_name;**

**employee.email = employeeObj.email;**

**employee.role = employeeObj.role;**

**employee.address = addressObj;**

**employee.compensation = compensationObj;**

**employee.family = familyObj;**

**employee.health = healthObj;**

**employee.photo = photoObj**

1. Save the updated employee data

…

**employee.save(function(err, data) {**

**if (err) {**

**console.log(err);**

**sendJSONResponse(res, 500, {**

**message: 'Saving Employee: Internal error while saving employee record'});**

**return;**

**}**

**var employee\_id = data.\_id;**

**var returned\_obj = {};**

**returned\_obj.employee\_id = employee\_id;**

**if (warning\_msg) {**

**returned\_obj.warning\_msg = warning\_msg;**

**}**

**sendJSONResponse(res, 200, {**

**result: returned\_obj**

**});**

**});**

9.