**Why PHP/MYSQL?**

\*\* NOW UPDATED FOR IONIC 3 \*\*

According to [W3tech's usage statistics PHP accounts for 82.4% of all the websites analysed](https://w3techs.com/technologies/details/pl-php/all/all) which means there's a strong possibility that as developers we will, at some point, need to work with that language in our projects.

This is no surprise as, amongst other features, PHP offers a solid and extensive Database API, supporting popular solutions ranging from MySQL, PostgreSQL, Oracle, SQL Server and SQLite to Sybase, Firebird and DB2. The almost universal hosting support for the language, low running costs and extensive online documentation also help to make this an attractive option for development!

On top of this [MySQL's usage statistics](http://db-engines.com/en/ranking) makes using PHP with MySQL something of a no-brainer for most organisations.

For the purposes of this tutorial here's the server set-up being used:

* Apache/2.4.23
* PHP 5.3+
* MySQL 5.5.53+

**What we'll be building**

So with our server-side technologies in mind the Ionic project that we're going to develop in this tutorial will be used to:

* Display a list of our favourite web/mobile technologies
* Create, amend and remove individual technology listings

All data will be stored remotely in a MySQL database which we'll interact with using [PHP Data Objects (PDO)](http://php.net/manual/en/book.pdo.php).

If you're relatively new to PHP or have never used PDO before it's simply a database abstraction layer that provides a universal API for working with different database vendors.

As the PDO syntax isn't tied specifically to one database API (I.e. such as with the [mysqli php commands](http://php.net/manual/en/book.mysqli.php)) this makes it an ideal solution for future proofing data management should the need arise, for example, to switch from MySQL to another database solution such as PostgreSQL or Oracle.

By the end of the tutorial we should be greeted with an application similar to the following (as seen running in the browser courtesy of the ionic serve utility):

Now that we've established the background to the project and what we expect to develop let's make a start...

**Setting up the database**

The data structure for our app is incredibly simple consisting of a single table with 3 fields.

That's it - nothing complex, no relationships, foreign keys or optimising indexes...thankfully!

Using your tool of choice (I.e. command line or phpMyAdmin for example) create a table named **technologies** with the following configurations:

| technologies | | |
| --- | --- | --- |
| **Field name** | **type** | **additional properties** |
| id | INTEGER(3) | PRIMARY KEY UNSIGNED AUTOINCREMENT |
| name | VARCHAR(50) |  |
| description | TEXT |  |

I told you it was simple didn't I?

And that's it - with our data structure configured we can now we can turn our attention to the PHP scripting for interacting with the database.

**Scripting the server side logic**

In your code editor of choice create the following PHP scripts:

* **manage-data.php**
* **retrieve-data.php**

The **manage-data.php** script will handle adding, updating and removing records from the **technologies** table.

The **retrieve-data.php** script will - surprise, surprise - handle retrieving saved data from the **technologies** table and returning that for use in our Ionic app.

Let's start with the scripting for the **manage-data.php** script:

<?php

header('Access-Control-Allow-Origin: \*');

// Define database connection parameters

$hn = 'localhost';

$un = 'root';

$pwd = '';

$db = 'ckkcheck;

$cs = 'utf8';

// Set up the PDO parameters

$dsn = "mysql:host=" . $hn . ";port=3306;dbname=" . $db . ";charset=" . $cs;

$opt = array(

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_OBJ,

PDO::ATTR\_EMULATE\_PREPARES => false,

);

// Create a PDO instance (connect to the database)

$pdo = new PDO($dsn, $un, $pwd, $opt);

// Retrieve specific parameter from supplied URL

$key = strip\_tags($\_REQUEST['key']);

$data = array();

// Determine which mode is being requested

switch($key)

{

// Add a new record to the technologies table

case "create":

// Sanitise URL supplied values

$name = filter\_var($\_REQUEST['name'], FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_ENCODE\_LOW);

$description = filter\_var($\_REQUEST['description'], FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_ENCODE\_LOW);

// Attempt to run PDO prepared statement

try {

$sql = "INSERT INTO technologies(name, description) VALUES(:name, :description)";

$stmt = $pdo->prepare($sql);

$stmt->bindParam(':name', $name, PDO::PARAM\_STR);

$stmt->bindParam(':description', $description, PDO::PARAM\_STR);

$stmt->execute();

echo json\_encode(array('message' => 'Congratulations the record ' . $name . ' was added to the database'));

}

// Catch any errors in running the prepared statement

catch(PDOException $e)

{

echo $e->getMessage();

}

break;

// Update an existing record in the technologies table

case "update":

// Sanitise URL supplied values

$name = filter\_var($\_REQUEST['name'], FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_ENCODE\_LOW);

$description = filter\_var($\_REQUEST['description'], FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_ENCODE\_LOW);

$recordID = filter\_var($\_REQUEST['recordID'], FILTER\_SANITIZE\_NUMBER\_INT);

// Attempt to run PDO prepared statement

try {

$sql = "UPDATE technologies SET name = :name, description = :description WHERE id = :recordID";

$stmt = $pdo->prepare($sql);

$stmt->bindParam(':name', $name, PDO::PARAM\_STR);

$stmt->bindParam(':description', $description, PDO::PARAM\_STR);

$stmt->bindParam(':recordID', $recordID, PDO::PARAM\_INT);

$stmt->execute();

echo json\_encode('Congratulations the record ' . $name . ' was updated');

}

// Catch any errors in running the prepared statement

catch(PDOException $e)

{

echo $e->getMessage();

}

break;

// Remove an existing record in the technologies table

case "delete":

// Sanitise supplied record ID for matching to table record

$recordID = filter\_var($\_REQUEST['recordID'], FILTER\_SANITIZE\_NUMBER\_INT);

// Attempt to run PDO prepared statement

try {

$pdo = new PDO($dsn, $un, $pwd);

$sql = "DELETE FROM technologies WHERE id = :recordID";

$stmt = $pdo->prepare($sql);

$stmt->bindParam(':recordID', $recordID, PDO::PARAM\_INT);

$stmt->execute();

echo json\_encode('Congratulations the record ' . $name . ' was removed');

}

// Catch any errors in running the prepared statement

catch(PDOException $e)

{

echo $e->getMessage();

}

break;

}

?>

PHP

Copy

The above script should be fairly self-explanatory in terms of what we're looking to accomplish at each stage.

We configure the necessary parameters for interacting with the database using PHP's PDO class.

From there we use a switch statement to determine whether we are adding, editing or removing a record from the database.

For each database operation we implement PHP's filter functions to sanitise the data supplied by the URL that calls the script before using prepared statements to manage interacting with the database.

Now to add the scripting for the **retrieve-data.php** script:

<?php

header('Access-Control-Allow-Origin: \*');

// Define database connection parameters

$hn = 'localhost';

$un = 'username-of-database-here';

$pwd = 'password-for-database-here';

$db = 'name-of-database';

$cs = 'utf8';

// Set up the PDO parameters

$dsn = "mysql:host=" . $hn . ";port=3306;dbname=" . $db . ";charset=" . $cs;

$opt = array(

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_OBJ,

PDO::ATTR\_EMULATE\_PREPARES => false,

);

// Create a PDO instance (connect to the database)

$pdo = new PDO($dsn, $un, $pwd, $opt);

$data = array();

// Attempt to query database table and retrieve data

try {

$stmt = $pdo->query('SELECT id, name, description FROM technologies ORDER BY name ASC');

while($row = $stmt->fetch(PDO::FETCH\_OBJ))

{

// Assign each row of data to associative array

$data[] = $row;

}

// Return data as JSON

echo json\_encode($data);

}

catch(PDOException $e)

{

echo $e->getMessage();

}

?>

PHP

Copy

Similar to our previous PHP script we begin with constructing our database/PDO parameters before attempting to retrieve records saved in the technologies table and return these as key/value pairs in JSON.

Did you notice the following PHP headers at the top of each script?

header('Access-Control-Allow-Origin: \*');

PHP

Copy

These instruct the PHP script to allow a request from ANY domain (through use of the \* wildcard character).

This will help avoid any issues with CORS (Cross Origin Resource Sharing) when requesting the URL for each PHP script from within our Ionic App (basically we want to make sure Angular doesn't spit its dummy out when trying to access a remote URL).

Be sure to upload these PHP scripts to a publicly accessible location on your remote server, ensure that the permissions are set to 755 and make a note of the absolute address to these scripts (I.e. **http://www.website-address.suffix/directory/retrieve-data.php**) as you will need this for the Ionic/Angular side of the project.

At this point we've concluded the necessary scripting for interacting with the database and returning data and now we can focus on developing the Ionic side of our project.

**Building the App**

The structure for the App itself is fairly simple; 2 pages:

* Home
* Add Technology

We won't be using any plugins, just a select number of pre-supplied Angular classes and Ionic UI components.

Navigate to a directory where you want your Ionic app to be created/installed and run the following command in your console of choice (I.e. Mac OS X Terminal or Windows Command Prompt for example):

ionic start favourite-web-technologies blank

Markup

Copy

Once the basic skeleton for the app has been created run the following command to add the second page for the app:

ionic g page addTechnology

Markup

Copy

With the bare bones of the app in place we can now start coding the required functionality and UI which, given Ionic's prebuilt components and available functionality, will be relatively quick and easy.

Begin by opening the **favourite-web-technologies/src/pages/home/home.ts** file and adding the following code:

import { Component } from '@angular/core';

import { NavController } from 'ionic-angular';

import { Http } from '@angular/http';

import 'rxjs/add/operator/map';

@Component({

selector: 'page-home',

templateUrl: 'home.html'

})

export class HomePage {

public items : any = [];

constructor(public navCtrl: NavController,

public http : Http)

{

}

ionViewWillEnter()

{

this.load();

}

// Retrieve the JSON encoded data from the remote server

// Using Angular's Http class and an Observable - then

// assign this to the items array for rendering to the HTML template

load()

{

this.http.get('<http://www.your-remote-url.suffix/retrieve-data.php>')

.map(res => res.json())

.subscribe(data =>

{

this.items = data;

});

}

// Allow navigation to the AddTechnology page for creating a new entry

addEntry()

{

this.navCtrl.push('AddTechnology');

}

// Allow navigation to the AddTechnology page for amending an existing entry

// (We supply the actual record to be amended, as this method's parameter,

// to the AddTechnology page

viewEntry(param)

{

this.navCtrl.push('AddTechnology', param);

}

}

TypeScript

Copy

Nothing too elaborate or complicated here - just calling our **retrieve-data.php** script through the **load** method and handling navigation to the **AddTechnology** page component using the **addEntry** and **viewEntry** methods.

Now we need to add the necessary UI components to the **favourite-web-technologies/src/pages/home/home.html** file:

<[ion-header](http://docs.webplatform.org/w/index.php?fulltext&search=ion-header)>

<[ion-navbar](http://docs.webplatform.org/w/index.php?fulltext&search=ion-navbar)>

<[ion-title](http://docs.webplatform.org/w/index.php?fulltext&search=ion-title)>

My Favourite Technologies

</[ion-title](http://docs.webplatform.org/w/index.php?fulltext&search=ion-title)>

</[ion-navbar](http://docs.webplatform.org/w/index.php?fulltext&search=ion-navbar)>

</[ion-header](http://docs.webplatform.org/w/index.php?fulltext&search=ion-header)>

<[ion-content](http://docs.webplatform.org/w/index.php?fulltext&search=ion-content) [padding](http://docs.webplatform.org/w/index.php?fulltext&search=padding)>

<[ion-item](http://docs.webplatform.org/w/index.php?fulltext&search=ion-item)>

<[button](http://docs.webplatform.org/wiki/html/elements/button)

[class](http://docs.webplatform.org/wiki/html/attributes/class)="add"

[ion-button](http://docs.webplatform.org/wiki/html/attributes/ion-button)

[item-right](http://docs.webplatform.org/wiki/html/attributes/item-right)

[icon-right](http://docs.webplatform.org/wiki/html/attributes/icon-right)

[margin-bottom](http://docs.webplatform.org/wiki/html/attributes/margin-bottom)

[color](http://docs.webplatform.org/wiki/html/attributes/color)="primary"

[(click)](http://docs.webplatform.org/wiki/html/attributes/(click))="addEntry()">

Add a technology

<[ion-icon](http://docs.webplatform.org/wiki/html/elements/ion-icon) [name](http://docs.webplatform.org/wiki/html/attributes/name)="add"></[ion-icon](http://docs.webplatform.org/wiki/html/elements/ion-icon)>

</[button](http://docs.webplatform.org/wiki/html/elements/button)>

</[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

<[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item) [\*ngFor](http://docs.webplatform.org/wiki/html/attributes/*ngFor)="let item of items">

<[h2](http://docs.webplatform.org/wiki/html/elements/h2)>{{ item.name }} </[h2](http://docs.webplatform.org/wiki/html/elements/h2)>

<[button](http://docs.webplatform.org/wiki/html/elements/button)

[ion-button](http://docs.webplatform.org/wiki/html/attributes/ion-button)

[color](http://docs.webplatform.org/wiki/html/attributes/color)="primary"

[item-right](http://docs.webplatform.org/wiki/html/attributes/item-right)

[(click)](http://docs.webplatform.org/wiki/html/attributes/(click))="viewEntry({ record: item })">View</[button](http://docs.webplatform.org/wiki/html/elements/button)>

</[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

</[ion-content](http://docs.webplatform.org/wiki/html/elements/ion-content)>

Markup

Copy

Once again, a fairly simple set-up here.

First we start with a standard button at the top right of the screen which calls the **addEntry** method from our class (allowing us to navigate to the **Add Technology** page in order to add a new record to be inserted into the remote MySQL database).

Secondly, we loop through and render to the page each technology record (that was retrieved through the **load** method of the class), assigning a button to accompany each listing. Each button calls the **viewEntry** method from our class, passing in the technlogy record as a parameter so this can be edited on the form (that we're going to build shortly) in the **Add Technology** page.

The more eagle-eyed amongst you may have noticed a glaring omission/problem with the above code.

If there are no records to retrieve - which will more than likely be the case when the app is first launched (unless you directly added some records to the database before getting to this stage of the tutorial) - the user isn't informed of this fact.

Instead all they are greeted with is a blank space underneath the Add Entry button.

Hmm...not good from a user experience perspective.

I'll leave this to you, as an additional exercise, to add in some functionality to handle this scenario :)

Now let's add the necessary logic and HTML for the **Add Technology** page.

**Adding and amending technology records**

As the **AddTechnology** page component will handle both adding new records and displaying existing records for potential amendment this simplifies the amount of code we need to write.

Starting with the **favourite-web-technologies/src/pages/add-technology/add-technology.ts** file implement the following code:

import { Component } from '@angular/core';

import { IonicPage, NavController, NavParams, ToastController } from 'ionic-angular';

import { FormGroup, Validators, FormBuilder } from '@angular/forms';

import { Http, Headers, RequestOptions } from '@angular/http';

import 'rxjs/add/operator/map';

@IonicPage()

@Component({

selector: 'page-add-technology',

templateUrl: 'add-technology.html'

})

export class AddTechnology {

// Define FormBuilder /model properties

public form : FormGroup;

public technologyName : any;

public technologyDescription : any;

// Flag to be used for checking whether we are adding/editing an entry

public isEdited : boolean = false;

// Flag to hide the form upon successful completion of remote operation

public hideForm : boolean = false;

// Property to help ste the page title

public pageTitle : string;

// Property to store the recordID for when an existing entry is being edited

public recordID : any = null;

private baseURI : string = "<http://localhost/>";

// Initialise module classes

constructor(public navCtrl : NavController,

public http : Http,

public NP : NavParams,

public fb : FormBuilder,

public toastCtrl : ToastController)

{

// Create form builder validation rules

this.form = fb.group({

"name" : ["", Validators.required],

"description" : ["", Validators.required]

});

}

// Determine whether we adding or editing a record

// based on any supplied navigation parameters

ionViewWillEnter()

{

this.resetFields();

if(this.NP.get("record"))

{

this.isEdited = true;

this.selectEntry(this.NP.get("record"));

this.pageTitle = 'Amend entry';

}

else

{

this.isEdited = false;

this.pageTitle = 'Create entry';

}

}

// Assign the navigation retrieved data to properties

// used as models on the page's HTML form

selectEntry(item)

{

this.technologyName = item.name;

this.technologyDescription = item.description;

this.recordID = item.id;

}

// Save a new record that has been added to the page's HTML form

// Use angular's http post method to submit the record data

// to our remote PHP script (note the body variable we have created which

// supplies a variable of key with a value of create followed by the key/value pairs

// for the record data

createEntry(name, description)

{

let body : string = "key=create&name=" + name + "&description=" + description,

type : string = "application/x-www-form-urlencoded; charset=UTF-8",

headers : any = new Headers({ 'Content-Type': type}),

options : any = new RequestOptions({ headers: headers }),

url : any = this.baseURI + "manage-data.php";

this.http.post(url, body, options)

.subscribe((data) =>

{

// If the request was successful notify the user

if(data.status === 200)

{

this.hideForm = true;

this.sendNotification(`Congratulations the technology: ${name} was successfully added`);

}

// Otherwise let 'em know anyway

else

{

this.sendNotification('Something went wrong!');

}

});

}

// Update an existing record that has been edited in the page's HTML form

// Use angular's http post method to submit the record data

// to our remote PHP script (note the body variable we have created which

// supplies a variable of key with a value of update followed by the key/value pairs

// for the record data

updateEntry(name, description)

{

let body : string = "key=update&name=" + name + "&description=" + description + "&recordID=" + this.recordID,

type : string = "application/x-www-form-urlencoded; charset=UTF-8",

headers : any = new Headers({ 'Content-Type': type}),

options : any = new RequestOptions({ headers: headers }),

url : any = this.baseURI + "manage-data.php";

this.http.post(url, body, options)

.subscribe(data =>

{

// If the request was successful notify the user

if(data.status === 200)

{

this.hideForm = true;

this.sendNotification(`Congratulations the technology: ${name} was successfully updated`);

}

// Otherwise let 'em know anyway

else

{

this.sendNotification('Something went wrong!');

}

});

}

// Remove an existing record that has been selected in the page's HTML form

// Use angular's http post method to submit the record data

// to our remote PHP script (note the body variable we have created which

// supplies a variable of key with a value of delete followed by the key/value pairs

// for the record ID we want to remove from the remote database

deleteEntry()

{

let name : string = this.form.controls["name"].value,

body : string = "key=delete&recordID=" + this.recordID,

type : string = "application/x-www-form-urlencoded; charset=UTF-8",

headers : any = new Headers({ 'Content-Type': type}),

options : any = new RequestOptions({ headers: headers }),

url : any = this.baseURI + "manage-data.php";

this.http.post(url, body, options)

.subscribe(data =>

{

// If the request was successful notify the user

if(data.status === 200)

{

this.hideForm = true;

this.sendNotification(`Congratulations the technology: ${name} was successfully deleted`);

}

// Otherwise let 'em know anyway

else

{

this.sendNotification('Something went wrong!');

}

});

}

// Handle data submitted from the page's HTML form

// Determine whether we are adding a new record or amending an

// existing record

saveEntry()

{

let name : string = this.form.controls["name"].value,

description : string = this.form.controls["description"].value;

if(this.isEdited)

{

this.updateEntry(name, description);

}

else

{

this.createEntry(name, description);

}

}

// Clear values in the page's HTML form fields

resetFields() : void

{

this.technologyName = "";

this.technologyDescription = "";

}

// Manage notifying the user of the outcome

// of remote operations

sendNotification(message) : void

{

let notification = this.toastCtrl.create({

message : message,

duration : 3000

});

notification.present();

}

}

TypeScript

Copy

The significant aspects of the above code are the logic to determine whether we are adding a new record or amending an existing record (contained within the **ionViewWillEnter** method) followed by the methods to create, update and delete records:

* createEntry
* updateEntry
* deleteEntry

With the logic for handling the supply of data to our remote **manage-data.php** script we can now concentrate on crafting the necessary HTML for the **favourite-web-technologies/src/pages/add-technology/add-technology.html** file:

<[ion-header](http://docs.webplatform.org/wiki/html/elements/ion-header)>

<[ion-navbar](http://docs.webplatform.org/wiki/html/elements/ion-navbar)>

<[ion-title](http://docs.webplatform.org/wiki/html/elements/ion-title)>{{ pageTitle }}</[ion-title](http://docs.webplatform.org/wiki/html/elements/ion-title)>

</[ion-navbar](http://docs.webplatform.org/wiki/html/elements/ion-navbar)>

</[ion-header](http://docs.webplatform.org/wiki/html/elements/ion-header)>

<[ion-content](http://docs.webplatform.org/wiki/html/elements/ion-content) [padding](http://docs.webplatform.org/wiki/html/attributes/padding)>

<[div](http://docs.webplatform.org/wiki/html/elements/div)>

<[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item) [\*ngIf](http://docs.webplatform.org/wiki/html/attributes/*ngIf)="isEdited && !hideForm">

<[button](http://docs.webplatform.org/wiki/html/elements/button)

[ion-button](http://docs.webplatform.org/wiki/html/attributes/ion-button)

[item-right](http://docs.webplatform.org/wiki/html/attributes/item-right)

[color](http://docs.webplatform.org/wiki/html/attributes/color)="secondary"

[text-center](http://docs.webplatform.org/wiki/html/attributes/text-center)

[block](http://docs.webplatform.org/wiki/html/attributes/block)

[(click)](http://docs.webplatform.org/wiki/html/attributes/(click))="deleteEntry()">Remove this Entry?</[button](http://docs.webplatform.org/wiki/html/elements/button)>

</[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

<[div](http://docs.webplatform.org/wiki/html/elements/div) [\*ngIf](http://docs.webplatform.org/wiki/html/attributes/*ngIf)="hideForm">

<[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item) [class](http://docs.webplatform.org/wiki/html/attributes/class)="post-entry-message" [text-wrap](http://docs.webplatform.org/wiki/html/attributes/text-wrap)>

<[h2](http://docs.webplatform.org/wiki/html/elements/h2)>Success!</[h2](http://docs.webplatform.org/wiki/html/elements/h2)>

<[p](http://docs.webplatform.org/wiki/html/elements/p)>Maybe you'd like to edit an existing entry or add a new record?</[p](http://docs.webplatform.org/wiki/html/elements/p)>

<[p](http://docs.webplatform.org/wiki/html/elements/p)>Simply go back to the home page and select the option you want to pursue.</[p](http://docs.webplatform.org/wiki/html/elements/p)>

</[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

</[div](http://docs.webplatform.org/wiki/html/elements/div)>

<[div](http://docs.webplatform.org/wiki/html/elements/div) [\*ngIf](http://docs.webplatform.org/wiki/html/attributes/*ngIf)="!hideForm">

<[form](http://docs.webplatform.org/wiki/html/elements/form) [[formGroup]](http://docs.webplatform.org/wiki/html/attributes/%5bformGroup%5d)="form" [(ngSubmit)](http://docs.webplatform.org/wiki/html/attributes/(ngSubmit))="saveEntry()">

<[ion-list](http://docs.webplatform.org/wiki/html/elements/ion-list)>

<[ion-item-group](http://docs.webplatform.org/wiki/html/elements/ion-item-group)>

<[ion-item-divider](http://docs.webplatform.org/wiki/html/elements/ion-item-divider) [color](http://docs.webplatform.org/wiki/html/attributes/color)="light">Technology Name \*</[ion-item-divider](http://docs.webplatform.org/wiki/html/elements/ion-item-divider)>

<[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

<[ion-input](http://docs.webplatform.org/wiki/html/elements/ion-input)

[type](http://docs.webplatform.org/wiki/html/attributes/type)="text"

[placeholder](http://docs.webplatform.org/wiki/html/attributes/placeholder)="Enter a name..."

[formControlName](http://docs.webplatform.org/wiki/html/attributes/formControlName)="name"

[[(ngModel)]](http://docs.webplatform.org/wiki/html/attributes/%5b(ngModel)%5d)="technologyName"></[ion-input](http://docs.webplatform.org/wiki/html/elements/ion-input)>

</[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

</[ion-item-group](http://docs.webplatform.org/wiki/html/elements/ion-item-group)>

<[ion-item-group](http://docs.webplatform.org/wiki/html/elements/ion-item-group)>

<[ion-item-divider](http://docs.webplatform.org/wiki/html/elements/ion-item-divider) [color](http://docs.webplatform.org/wiki/html/attributes/color)="light">Technology Description \*</[ion-item-divider](http://docs.webplatform.org/wiki/html/elements/ion-item-divider)>

<[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

<[ion-textarea](http://docs.webplatform.org/wiki/html/elements/ion-textarea)

[placeholder](http://docs.webplatform.org/wiki/html/attributes/placeholder)="Description..."

[formControlName](http://docs.webplatform.org/wiki/html/attributes/formControlName)="description"

[rows](http://docs.webplatform.org/wiki/html/attributes/rows)="6"

[[(ngModel)]](http://docs.webplatform.org/wiki/html/attributes/%5b(ngModel)%5d)="technologyDescription"></[ion-textarea](http://docs.webplatform.org/wiki/html/elements/ion-textarea)>

</[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

</[ion-item-group](http://docs.webplatform.org/wiki/html/elements/ion-item-group)>

<[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

<[button](http://docs.webplatform.org/wiki/html/elements/button)

[ion-button](http://docs.webplatform.org/wiki/html/attributes/ion-button)

[color](http://docs.webplatform.org/wiki/html/attributes/color)="primary"

[text-center](http://docs.webplatform.org/wiki/html/attributes/text-center)

[block](http://docs.webplatform.org/wiki/html/attributes/block)

[[disabled]](http://docs.webplatform.org/wiki/html/attributes/%5bdisabled%5d)="!form.valid">Save Entry</[button](http://docs.webplatform.org/wiki/html/elements/button)>

</[ion-item](http://docs.webplatform.org/wiki/html/elements/ion-item)>

</[ion-list](http://docs.webplatform.org/wiki/html/elements/ion-list)>

</[form](http://docs.webplatform.org/wiki/html/elements/form)>

</[div](http://docs.webplatform.org/wiki/html/elements/div)>

</[div](http://docs.webplatform.org/wiki/html/elements/div)>

</[ion-content](http://docs.webplatform.org/wiki/html/elements/ion-content)>

Markup

Copy

As you can see this is a very simple form, consisting of a small number of very basic elements.

Initially we have a button located at the top right-hand side of the page which calls the **deleteEntry** method from our class.

Following from this we use an **ngIf** directive to conditionally display a message to the user once a form has been submitted and the data successfully saved.

Then another **ngIf** directive is used to handle whether the form is displayed or not based on the value of the **hideForm** property which is managed through the component's TypeScript class.

Our form consists of an input field for the technology name and a textarea for the technology description; both of which have form controls and models assigned to them to manage validation and the data contained within these fields.

When we come to preview the Add a technology view in the browser we are presented with the following screen:

And when we have selected a record from the application home page to view/amend:

**Managing the application modules**

All is good and well with what we've implemented so far but there's one amendment we need to make to the AddTechnology component module - **favourite-web-technologies/src/pages/add-technology/add-technology.module.ts** file.

Change ALL instances of IonicModule to IonicPageModule so that the file resembles the following code:

import { NgModule } from '@angular/core';

import { IonicPageModule } from 'ionic-angular';

import { AddTechnology } from './add-technology';

@NgModule({

declarations: [

AddTechnology,

],

imports: [

IonicPageModule.forChild(AddTechnology),

],

exports: [

AddTechnology

]

})

export class AddTechnologyModule {}

TypeScript

Copy

This amendment bootstraps the AddTechnology component allowing it to be navigated to by its class name (albeit in the form of a string) - as we have done within the **addEntry** and **viewEntry** methods of the **favourite-web-technologies/src/pages/home/home.ts** file.

One final modification remains before we can preview the project in the browser however.

As we are using Angular's Http class we need to import and declare the HttpModule within the application's root module - **favourite-web-technologies/src/app/app.module.ts** like so:

import { BrowserModule } from '@angular/platform-browser';

import { HttpModule } from '@angular/http';

import { ErrorHandler, NgModule } from '@angular/core';

import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';

import { SplashScreen } from '@ionic-native/splash-screen';

import { StatusBar } from '@ionic-native/status-bar';

import { MyApp } from './app.component';

import { HomePage } from '../pages/home/home';

@NgModule({

declarations: [

MyApp,

HomePage

],

imports: [

BrowserModule,

HttpModule,

IonicModule.forRoot(MyApp)

],

bootstrap: [IonicApp],

entryComponents: [

MyApp,

HomePage

],

providers: [

StatusBar,

SplashScreen,

{provide: ErrorHandler, useClass: IonicErrorHandler}

]

})

export class AppModule {}

TypeScript

Copy

With all of our changes in place now you can preview the app in your desktop browser using the ionic serve command from your system command line utility (I.e. the Terminal on Mac OS X or Command Prompt on Windows):

ionic serve

Markup

Copy

All things being well you should be able to add records and see those being displayed as shown in the following screen grabs:

**In summary**

Here we conclude our tutorial on integrating PHP & MySQL with Ionic to create a basic app for listing, adding, editing and removing our favourite web technologies.

We've worked with a range of technologies from PHP Data Objects (PDO) and MySQL to Ionic's UI components and Angular's Http service.

As a result of these we've built a very simple but aesthetically pleasing and functional app which demonstrates just how easy it is to use PHP/MySQL as a data management solution for our Ionic apps.

Pretty good going huh?

There are, however, some modifications that do need to be made with the above codebase:

* Adding network detection functionality to determine whether or not the app can actually connect to a network for retrieving/posting data
* Implementing the ability to manage our data offline
* Handling more advanced validation scenarios such as avoiding duplicate data entries and restricting the length of data that can be added through each field for example

I've deliberately left these out as an exercise for the reader to implement themselves (should you feel so adventurous!)

What did you think of this tutorial? Did it cover the necessary angles or could it be improved in some way?

Let me know what your thoughts and reactions are by leaving your comments in the form below.

If you enjoyed what you've read here then please consider signing up to my mailing list and, if you haven't done so already, take a look at my e-book: [Mastering Ionic](http://masteringionic.com/products/product-detail/s/mastering-ionic-e-book) for information about working with alternative data storage methods in Ionic.