

Training #1

|  |  |
| --- | --- |
| **Document Owner:** | TNation Team |
| **Email:** |  |
| **Telephone:** |  |
| **Department:** |  |

|  |  |
| --- | --- |
| **Document Version:** | 1.2 |
| **Date:** | August 2016 |
| **Status:** |  |
| **Approved by:** |  |

*This document comprises proprietary and confidential information belonging to Hostopia Inc.*

Table of contents

[1 Intro 4](#_Toc431551470)

[2 WSOS 6](#_Toc431551471)

[2.1 WebsiteOS 6](#_Toc431551472)

[2.1.1 About WebsiteOS 6](#_Toc431551473)

[2.1.2 WebsiteOS File System 6](#_Toc431551474)

[2.1.3 WebsiteOS Assignments 7](#_Toc431551475)

[2.1.4 Useful Tips 11](#_Toc431551494)

[2.1.5 WebsiteOS API Cluster 12](#_Toc431551496)

[2.1.6 WebsiteOS API Database 13](#_Toc431551497)

[2.1.7 WebsiteOS Application Version 13](#_Toc431551498)

[2.1.8 WebsiteOS API Error 15](#_Toc431551504)

[2.2 Mobile 16](#_Toc431551505)

[2.2.1 About Mobile 16](#_Toc431551506)

[2.2.2 Mobile File System 16](#_Toc431551507)

[2.2.3 MVC Architecture 17](#_Toc431551508)

[2.2.4 Building Simple App 17](#_Toc431551509)

[2.2.5 Useful Tips 21](#_Toc431551510)

[2.3 RING 23](#_Toc431551511)

[2.3.1 About RING 23](#_Toc431551512)

[2.3.2 RING File System 23](#_Toc431551513)

[2.3.3 Building Simple App 24](#_Toc431551514)

[3 Retails 28](#_Toc431551516)

[3.1 About Retails 28](#_Toc431551517)

[3.2 Retails File System 2](#_Toc431551518)8

[3.3 Building Simple App 29](#_Toc431551519)

[4 RRAD 45](#_Toc431551521)

[4.1 About RRAD 45](#_Toc431551522)

[4.2 RRAD File System 45](#_Toc431551523)

[4.3 Simple rest api request with database access 46](#_Toc431551524)

[5 Repositories 55](#_Toc431551526)

[5.1 GIT 55](#_Toc431551527)

[5.1.1 About GIT 55](#_Toc431551528)

[5.1.2 GIT Basics 55](#_Toc431551529)

[5.1.3 GIT WORKFLOW 59](#_Toc431551530)

[5.2 SVN 62](#_Toc431551531)

[5.2.1 About SVN 62](#_Toc431551532)

[5.2.2 SVN Basics 62](#_Toc431551533)

[6 Debugging 69](#_Toc431551534)

# Intro

With this training, we will go through three platforms on which we work: WSOS, Retails and RRAD. On WSOS platform we are working on three repositories: WebsiteOS, Mobile-CP and RING. All of these repositories, their file systems, code examples and useful tips, will be explained below training. We will show RRAD file system, a simple rest api request with database access, and RRAD useful tips. We will pass the basics of GIT and SVN version control systems with ususal GIT flow that is being used in hostopia. Below is map of of data flow in hostopia:

z:\Hostopia\--Procedure, treninzi, arhitektura--\-- 01 Training#1 - Roman in Serbia--\Map.png

# WSOS

## WebsiteOS

### About WebsiteOS

This is browser based website management applications that allows you to manage many aspects of your website. These applications are applied in the control panel. It will help you become familiar with WebsiteOS applications by creating a basic application.

### WebsiteOS File System

The WebsiteOS starts with /websiteos and contains the following directories:

* cgi-bin
* sessions
* public
* websiteos

cgi-bin is a symlink.

sessions is directory where WebsiteOS stores user sessions.

public is directory where WebsiteOS stores index pages

websiteos is directory where WebsiteOS stores the most of code and contains the following:

* api
* applications
* gui
* misc
* render
* resources
* util
* websiteos.php

api is directory which contains api’s.

applications is directory which contains individual WebsiteOS applications, ‘code’ and ‘interface’ subdirectories. For example code for ‘EasyMail’ application you will find in code/easymail and the interface for same application you will find in interface/easymail. Inside these directories you will find subdirectories of versions that correspond to the version of the easymail application and symlink ‘current’ that must point to the latest version of application. This directory contains the actual php files that are used. Inside ‘interface’ directory you will also find an ‘image’ and ‘lang’ directory.

The /websiteos/websiteos/resources directory contains following subdirectories:

* coreframes
* corepanels
* fonts
* javascript
* lang
* plugins
* public
* themes

### WebsiteOS Assignments

The following is a list of WebsiteOS small assignements.

Assignment 1

Purpose:

Build a WebsiteOS application that displays “Hello world” in the center of the application window.

Hint:

1. ../applications/code/helloworld/1.0/helloworld.php using:

init()

start()

getdisplay()

2. ../applications/interface/helloworld/1.0/helloworld.ddml

Assignment 2

Purpose:

Using Project #1 have your application support Multilingual

Hint:

1. ../applications/interface/helloworld/lang/en.strings

Assignment 3

Purpose:

Using Project #2 have your application ask you to enter & submit your name using form submission. The result displays "Hello <name>" at the top of the application page and asks for your name again.. etc

Assignment 4

Purpose:

Using Project #3 have your application use the Tab feature. The application will have 2 tabs. One tab called 'Name' contains the Enter name function (from project#3). The second tab is called 'Description'. When clicked, it will display two radio buttons and will ask.. Are you? Male or Female . A submit will keep the selected value.

1. Under Name Tab

After Entering name, you can click a 'Apply' button. This keeps value in name field; it does not move you to Description page. You must only click the Description tab to move to the description page, (you will have to send and 'store' the name value on this page)

1. Under Description Tab

After selecting gender value, you click a submit button called 'Apply'. This keeps the gender value (selected) on the appropriate radio button.

1. By moving from one Tab to the other, the selected values are sent to each page. If you change any value and click Apply it is properly sent from page to page.

Hint:

Use…

1. $TABSTRIP
2. <TABPANE> ... </TABPANE>
3. $this->register("GENDER\_RADIO","genderRadio")

Assignment 5

Purpose:

Using Project #4 store the name and gender values in a comma delimited (csv) file in the websiteos login accounts home directory. When you enter the same name, the gender is automatically selected.

Hint:

Use…

1. you will require a policy file, helloworld.policy.php

ie new FilePermission

1. use FileUtils.php to write/read file
2. consider using values in User object

Assignment 6

Purpose:

Using Project #5…

1. use javascript (trapevent) to ensure that a value is entered in the Name field before 'Apply' is clicked, otherwise a popup windowwarning is displayed. In Addition, the Description Tab is 'grayed' (unclickable) until a Name is entered and 'Apply' is clicked.
2. add to the Description Tab a hListbox (<select></select>) that asks the user for their eye colour. The hoption (<option>) values are read from a comma delimited (csv) file location under /services/websiteos/helloworld/ called eyecolours.cfg containing blue,brown,hazel,green
3. add to the Description Tab a hCheckbox (<input type=checkbox ..>) that asks the user to choose their hobbies (interests). The values are read from a comma delimited (csv) file located under /services/websiteos/helloworld/ called hobbies.cfg containing music,sports,movies,theatre,reading. The user will select all that apply.

Hint:

Use…

1. code/helloworld/current/helloworld.js

Assignment 7

Purpose:

Using Project #6, convert using csv files to using a mysql db (ie HELLOWORLD) that:

1. reads/writes the name and description values from/into a table (ie HELLOWORLD.users)
2. reads the configuration values from a table (ie. HELLOWORLD.config)

Hint:

1. create Database abstraction layer for helloworld ie. api/HelloworldDB.php

eg. class HelloworldDB extends SkeletonDB

Assignment 8

This is a \* NEW Helloworld Application \* (does not extend Assignment #7).

Purpose:

This application will have 3 tabs: "Usage", "Contact", "Service". It will be internationalized. It will use the RRAD api to:

1. under tab "Usage" display:

Usage Info eg. disk space used

Product type

1. under tab "Contact" display:

domains Contact Information And will allow updates. eg. "FirstName","LastName", "Company", "Address1" ... "Email", "Phone", etc.

1. under tab "Service" displays:

the service, BGVSCANFPROT which can be turned On or Off.

Hint:

1. use latest rrad version eg. api/rrad

Assignment 9

This project extends project#8.

Purpose:

Display on the Usage tab page the userName & clientIP (Read from session). Add to session a value containing "Your Name" and display on bottom center of each page.

Hint:

1. use the Session api

$this->session->userName

$this->session->addToSession();

$this->session->register();

### Useful Tips

Screen:

‘Screen’ is a secure way to view terminal sessions. It allows you to see websiteos application development. See ‘man screen’ for more information.

The following Screen Rules have been defined.

1. all websiteosteam members will use screen
2. each member will have a min. 1 chat session and 1-3 edit sessions
3. chat session will allow 'chatting' with another websiteosteam member in TOR
4. edit session will allow the TOR member to view websiteos file editing

Usage:

login to screen1 account using ssh (see manager for password)

once logged in, you can see existing sessions

screen –ls

1. When starting the screen session do this:

screen -S < screen name > -h 100000

This gives you a scrollback buffer of one hundred thousand lines.

During the screen session, if you want to scroll back, you must do:

"Ctrl-a" then "Esc"

This puts you into "Copy mode".

You can then use the arrow keys to go up and down, or page up and page down.

Once done, press

"Esc" to leave "Copy mode"

VIM/Vi:

1. In Vi, the "page up" and "page down" is inconsistent and sometimes it is hard to figure out where you are.

In Vi, use the keys:

Ctrl-f (for forward), and Ctrl-b (for back)

This works reliably, every time.

1. Inconsistent command line editing in Vi ("the filename is lost etc.").

This happens if you resize your window to be smaller, Screen doesn't know about it, so the VI status line is lost. This is very easy to fix, simply do:

"Ctrl-a" followed by "F"

REMEMBER: it is case sensitive. so "Ctrl-a", "F" is different than "Ctrl-a", "f".

### WebsiteOS API Cluster

The Cluster API provides a simple way to retrieve the value of cluster-dependent properties. An instance of the Cluster class is available in the global $OS object:

$OS->cluster

The Cluster instance will be pre-populated with values which are relevant to the cluster on which the application code is executing. The constructor for the Cluster object loads the values from the FARMStable.

To retrieve the value of a cluster-dependent property, use the get() method. For example:

global $OS;

$dbServer = $OS->cluster->get("CustomerMySQL"); // get the customer SQL server host name

### WebsiteOS API Database

The Database api encapsulates the connection, query and result stages of database work. In addition, it hides the fact that the low level database is MySQL.

The connection stage requires instantiating the appropriate Database object and then utilizing the Low Level Database to run SQL queries.

The DBQuery class provides functionality that allows a user to build an SQL query.

Once you have your query object ($q), you can now execute it.

The SQLResult object is created for operations that return data such as sql\_select. Important methods in this class are the iterators 'next()' and 'prev()'.

### WebsiteOS Application Version

The following describes the DB tables and how they are used by WebsiteOS.

#PANELAPPS

* List of applications, version, code level and hostopianid.
* The default Hostopian minimum code level is ACTIVE, and ACTIVE apps will have precedence over FROZEN apps.
* The HostopianId field in this table is NOT USED.
* If an application is below the Hostopian’s minimum code level but it is enabled for the Hostopian, the version will displays as ‘??’.

Code levels (and priority):

* 'FROZEN' = 4
* 'ACTIVE' = 3 (precedence over FROZEN)
* 'BETA' = 2
* 'ALPHA' = 1
* 'HACK' = 0

#PANELSETUP

* Hostopian level WebsiteOS settings
* Lists the lowest application Code Level available in the AppLevel field.

#PANELAPPVERSIONS

* Hostopian level application access
* The application does not need to meet the Hostopian's minimum code level. If it appears here, it will be displayed to the user.
* If an application for a Hostopian does not appear in this table, the "current" version of the application is used.

The way in which you can check the version:

select \* from PANELAPPVERSIONS where appid=’app\_id’ and hostopianid=’hostopian\_id’

The way in which you can change the version:

update PANELAPPVERSIONS set version=’api\_version’ where hostopianid=’hostopian\_id’

#PANELACCESS

* Domain level application access
* "\*" indicates the current version of the application
* An application version starting with "-" indicates that this application should be revoked for the user
* If multiple entries of the same application for the same user exists, the last entry selected from the DB will be used.

#PANELAPPGROUPS

* List of application groups (NavMenu - left menu in websiteOS)
* An application must belong to a group
* The hostopianid field in the table is NOT USED! The first entry is used.
* The language field in the table is NOT USED for display groups in the NavMenu (top left)! Only the english language is used ("en").
* The language field in the table IS USED to obtain the list of applications to display in the NavDetail (middle left). The first of the user's language or english ("en") is used.
* Group names are obtained from a string file (websiteos.strings). If the application group does not exist in the string file, the name from the database is used.

#OSTEMPLATES

* List of application included in a template (package)
* If the template exists for the user's hostopian, it will be used. Else, the template for hostopianid "null" will be used.
* VMS accounts use the template "RSMASTER"
* The application does not need to meet the Hostopian's minimum code level. If it appears here, it will be displayed to the user.

### WebsiteOS API Error

The Error API exists to provide a simple way of storing and retrieving error/informational messages that can aid in debugging.

The API recognizes 8 different "levels" of message:

* ERR\_WARN
* ERR\_USER\_NOTICE
* ERR\_SYSTEM
* ERR\_APP\_LOGIC
* ERR\_APP\_SECURITY
* ERR\_APP\_FATAL
* ERR\_NOTICE
* ERR\_FINAL

You can set the level of messages that you are interested in capturing by using the setLevel() method.

## Mobile

### About Mobile

Clients control panel used for the management of products using mobile devices. The whole control panel/platform is written in backbone framework and marionette.js, and therefore uses only front end functionality (html, css, js). Communication with database works via Rest API calls on rrad. Each of developers have instance of mobile control panel cloned on thear env.

### Mobile File System

Root directory for the dev environment for mobile commonly found on:

/development/developer\_name/public/mobile-cp

This directory contains following subdirectories:

* php/
  + models/
    - model.php
  + templates/
    - template.html
* js-dev/
  + pages/
    - shop/
      * shop\_control.js
      * shop\_app.js
      * helloWorld/
        + helloWorld\_control.js
        + helloWorld\_view.js
  + models/
    - productsInfo.js
* css/
  + main.css

*Model.php* is a file which contains the path of files that are included for all applications.

*Template.html* is a file which contains Undersore templates. Each template is located in a separate script and has its own id that calls from view.

All request calls that are triggering controllers are stored in *shop\_app.js*

In *productsInfo.js* are stored local data about some products. This file is a model in MVC architecture.

File “*helloworld\_control.js” is basicly collection in backbone that* control data, how you want to show them in view. This file forwards the data to the view.

In *helloworld\_view.js* you can do an additional manipulation of the data, and handle the events.

### MVC Architecture

Each application is usually made up of several parts:

* Model
* Controller
* View
* Template

Model-view-controller is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of informations from the ways that information is presented to or accepted from the user.

A model is an object representing data or even activity. Model stores data that is retrieved according to commands from the controller and displayed in the view.

A view is some form of visualization of the state of the model. View generates an output presentation to the user based on changes in model.

A controller offers facilities to change the state of the model. Controller can send commands to the model to update the model’s state. It can also send commands to its associated view to change the view’s presentation of the model.

### Building Simple App

File name - helloworld\_control.js

mobileApp.module('pages.shop.helloWorld', function(helloWorld, mobileApp, Backbone, Marionette, $, \_)

{

helloWorld.controller = {

show: function(){

/\* We are using productInfo.js because don't have this product in database. \*/

var productInfo = mobileApp.request('get:productInfo', 'HELLOWORLD'),

collection = [],

view;

// Getting domains from database

$.when(mobileApp.request('get:domains')).done(function(response){

for(var domain in response){

collection.push(response[domain]);

}

view = new mobileApp.pages.shop.helloWorld.view.collection({

collection: new Backbone.Collection({domains: collection});

});

mobileApp.on('use:current:domain', function(domain){

var product = {};

//This example is like purchase flow for SSL service

product.additionalServices = productInfo.additionalServices;

product.currentService = productInfo.code;

product.title = productInfo.title + ' <br/> (' + domain + ')';

product.domain = domain;

// Triggering event for adding product in cart

mobileApp.trigger('cart:add', product, 'service');

mobileApp.backTrigger = 'shop:show';

// Change route

mobileApp.startRouter.navigate('cart', {trigger: true});

});

mobileApp.mainRegion.show(view);

// Show empty footer

mobileApp.footerRegion.empty();

// Setting back button path

mobileApp.trigger('setBackMenu', 'Hello World', 'shop:show');

}).fail(function(){

// Showing error message if getting domains is failed

mobileApp.trigger('dialog:show','error', 'You don\'t have any domain with hosting.');

});

}

}

});

File name - helloworld\_view.js

mobileApp.module('pages.shop.helloWorld.view', function(view, mobileApp, Backbone, Marionette, $, \_)

{

var selectDomain = Marionette.ItemView.extend({

template: '#helloWorld',

ui: {

selectDomain:'#js-selectDomain',

activeSpan: '#active-selectDomain',

continueButton: '.js-clicked'

},

events: {

'change #js-selectDomain': 'domainSelect',

'click .ap-button-orange': 'showCart'

},

domainSelect: function() {

var selected = this.ui.selectDomain.find('option:selected');

this.ui.activeSpan.text(selected.text());

if ('' != selected.val()){

this.ui.continueButton.removeClass('ap-button-grey').addClass('ap-button-orange');

} else{

this.ui.continueButton.removeClass('ap-button-orange').addClass('ap-button-grey');

}

},

showCart: function(e) {

e.preventDefault();

mobileApp.trigger('use:current:domain',

this.ui.selectDomain.find('option:selected').attr('value'));

}

});

view.collection = Marionette.CollectionView.extend({

childView: selectDomain

});

});

File name - template.html

<script type="text/template" id="helloWorld">

<div class='ap-content ap-content-pad-box'>

<div class="ap-gray-line"></div>

<h2 class="ap-content-title-text-lowercase">Hello World</h2>

<div class="ap-gray-box ap-gray-text-form">

<div class="ap-resp-pad">

<div class="ap-clear">

<div class="ap-select-custom">

<span class="ap-select-custom-value" id="active-selectDomain">

Select Domain Name

</span>

<select name="domainName" id="js-selectDomain">

<% if ( 'undefined' != typeof domains && \_(domains).isArray() && 0 < domains.length ) { %>

<% \_.each(domains, function(domain, key){ %>

<option value="<%= domain %>"><%= domain %></option>

<% }); %>

<% } %>

</select>

<i></i>

</div>

</div>

</div>

</div>

<div class="ap-footer ap-clear">

<a href="javascript:void(0);" class="js-clicked ap-button ap-button-grey ap-button-big ap-button-right">

Continue

<i class="ap-button-arrow-right js-arrow"></i>

</a>

</div>

</div>

</script>

File name - productsInfo.js

HELLOWORLD: {

code: 'HELLOWORLD',

shortInfo: 'Your Hello World dashboard will show you a list of infected pages and

potential threats, and Hello World’sour Expert Service team will help you

remove the issues!',

fullInfo: ' Your Hello World dashboard will show you a list of infected pages and

potential threats, and Hello World’sour Expert Service team will help you

remove the issues! Plus they are available 24/7 to help with any questions you

may have',

title: 'Hello World',

dollars: 50,

cents: 00,

additionalServices: {

HELLOWORLD: {

Code: "HELLOWORLD",

Currency: "USD",

FixedPrice: "none",

Price: "49.00",

Surcharge: "0.00",

TPeriodDays: "0",

TPeriodPrice: "0.00",

TaxRate: "0.07125",

TaxType: "USTAX",

Termlength: "12"

}

}

}

File name - apiCalls.js

helloWorld: function(){

shopApp.helloWorld.controller.show();

},

mobileApp.on('shop:helloWorld:show', function(){

API.helloWorld();

});

### Useful Tips

We are using two usefull features of Backbone.wreqr plugin:

- Backbone.Wreqr.EventAggregator();

We are using this mostly for triggering dialog messages, setting back buttons paths, triggering controllers and calling other functions without return.

example:

mobileApp.on('shop:hostingServices:sslCertificates:setup', function(data){

API.sslCertificatesSetup(data);

});

mobileApp.trigger('shop:hostingServices:sslCertificates:setup', dataCached);

- Backbone.Wreqr.RequestResponse();

We are using this mostly for triggering API calls and some functions with return.

example:

mobileApp.reqres.setHandler("get:products", function(products){

return API.getProducts(products);

});

mobileApp.request('get:products', sslCodes).done(function(products){

});

## RING

### About RING

Clients control panel used to operate management products using pc devices. The entire control panel/platform is written in native js/jquery, and therefore only uses front-end functionality (html, css, js). Communication with database works via Rest API calls on rrad. Each of developers have instance of ring control panel. Ring control panel is part of whole control panel, ie. consists of:

* My services
* Add services / Shop
* Support
* Web mail
* Account

My services is responsible for displaying and listing websiteos applications. Here you can access and launch all of the products and services associated with your account.

Add services / Shop is a part where purchase any products or services you may need for your account.

On support you can find support for your services, as well as contact information and FAQ’s. here you have ‘Messages’, ‘Tickets’, ‘View FAQ’, ‘Send feedback’ and ‘News and Updates’.

### RING File System

RING file system contains the following directories and files:

* /HelloWorld/
  + main.php
  + script.js
* head\_all.php
* rev\_cp.js
* rev\_cp.php
* style.css

In *head\_all.php* are linked/loaded all js and css files.

File *rev\_cp.js* contains main js functionality for RING applications.

Html skeleton for RING applications where all other section are included is stored in *rev\_cp.php* file.

File *style.css* contains shared css for deluxe and aplus applications.

Directory *HelloWorld* contains all files for helloworld section.

Basic html structure of helloworld section is in *main.php* file.

JavaScript functionality for helloworld section ( *init(), getDomains(), displayData(), reloadData()* ) are in *script.js* file.

### Building Simple App

File name - main.php

//section content is defined in thise file

<div class="helloWorld">

<table id="js-domainName">

<tr>

<td>Domains</td>

<td>Account ID</td>

<td>Customer Number</td>

</tr>

<tbody id='tbody'>

</tbody>

</table>

</div>

<a onclick="DFB\_CP.HELLO\_WORLD.reloadData()">Reload</a>

File name - script.js

// functionality of section

/\*\*

\* Object for Hello World section

\*

\*/

DFB\_CP.HELLO\_WORLD = (function(W, $) {

var all\_domains = [];

function init() {

getDomains();

}

/\*

\* getDomains()

\* Geting data from base

\* @returns {void}

\*/

function getDomains() {

all\_domains = [];

$.ajax({

type: 'GET',

url: '/api/db/web/domains.json',

dataType: 'json',

async: false,

// if ajax call is ok

success : function (web\_domains) {

for (i = 0; i < web\_domains.length; i++) {

if ('R' == web\_domains[i].Atype) {

all\_domains.push({

'Domain' : web\_domains[i].Domain,

'AccountID' : web\_domains[i].AccountID,

'CustomerNumber' : web\_domains[i].CustomerNumber

});

}

}

displayData();

},

// if ajax call have error

error: function() {

$(".helloWorld").append("<h2>Service is currently not available.</h2>");

}

});

}

/\*

\* displayData(param)

\* Render data from ajax call

\* @param {String | Required} all\_domains

\* @returns {void}

\*/

function displayData() {

var html = '';

if (0 == all\_domains.length) {

// TODO hide

$(".helloWorld").append("<h2>No domains.</h2>");

} else {

for (i = 0; i < all\_domains.length; i++) {

html += ['<tr>',

'<td>' + all\_domains[i].Domain + '</td>',

'<td>' + all\_domains[i].AccountID + '</td>',

'<td>' + all\_domains[i].CustomerNumber + '</td>',

'</tr>'].join('');

}

$('#tbody').append(html);

$("#js-domainName").show();

}

}

/\*

\* reloadData()

\* reload all data

\* @returns {void}

\*/

function reloadData(){

//empty table for

$('#tbody').empty();

// mesage before reload data

$(".helloWorld").append("<h2 class='wait'>Please wait...</h2>");

// time out is set

setTimeout(function () {

getDomains();

}, 2000)

}

/\*

\* Public methods

\*/

return {

init: init,

reloadData: reloadData

}

}(window, jQuery));

File name - head\_all.php

<!-- include js file from Hello World section -->

<script src="HelloWorld/script.js?\_=<?=$last\_change?>" type="text/javascript"></script>

File name - rev\_cp.js

//in top of file where all vars are declared

mainSectionName = {

// name of the section

helloworld: 'Hello World'

}

// in function switchSection()

// define function to call when the section is changed to hello world

if('helloworld' == section){

DFB\_CP.HELLO\_WORLD.init();

}

File name - rev\_cp.php

// in <ul class="ap-header-menu-list ap-header-menu-left">

<li>

<!-- define a path in the navigation menu to sections -->

<a href="#" onclick="DFB\_CP.switchSection('helloworld'); return false;">Hello World</a>

</li>

// in <div class="ap-wrap-center ap-content-table-mode">

<!-- content of a new section -->

<div id="helloworld\_box\_content" style="display: none;">

<?php include\_once("HelloWorld/main.php"); ?>

</div>

File name - style.css

#js-domainName{

display: none;

width: 100%;

}

# Retails

## About Retails

A content management system (CMS) is a system used to manage the content of a Web site. Typically, a CMS consists of two elements: the content management application (CMA) and the content delivery application (CDA). The CMA element allows the content manager or author, who may not know Hypertext Markup Language (HTML), to manage the creation, modification, and removal of content from a Web site without needing the expertise of a Webmaster. The CDA element uses and compiles that information to update the Web site. The features of a CMS system vary, but most include Web-based publishing, format management, revision control, and indexing, search, and retrieval. Drupal is one of content management systems. It is a free software package that allows you to easily organize, manage and publish your content, with an endless variety of customization. We can say that Drupal 7.0 is content management platform for building nearly any kind of website. Thousands of add-on modules and designs let you build any site you can imagine. Hostipia is using Drupal for front-end work on sites,in this case Drupal Retails allows you to build online store what Hostopia used to build its cart funcionalitty among others.

## Retails File System

Retails root directory contains following directories and files:

* /signup/v1/
  + config/
    - config.js
    - signup\_conf\_en.js
  + css/
    - su-styles.css
  + js/
    - signup.js

All blocks, pages and data that are needed for modules that are included in drupal are stored in the database.

Directory *v1* contains all files for desktop/shop.

In *config* directory you can find config files which are created automatically.

Some functions and descriptions for some products created with data taken from database are stored in *config.js* file.

All informations about all products are stored in *signup\_conf\_en.js* file. Some functions from config.js are using variables from this file.

Directory *css* hold all files for css.

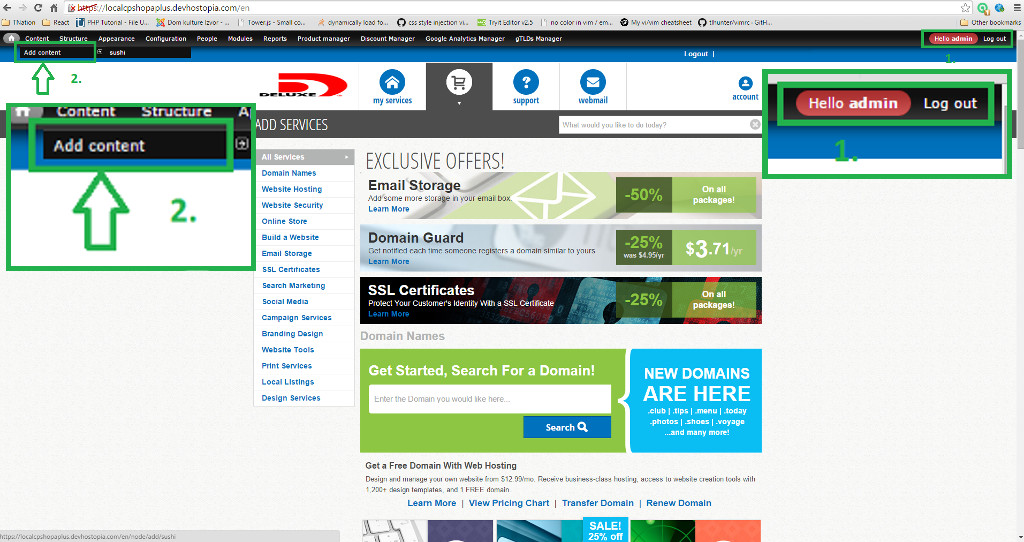
All additional css work is written in *su-styles.css* file.

All javascript files that programmers write goes in *js* directory.

All things that are done by javascript, you can find in *signup.js* file.

## Building Simple App

Picture 1

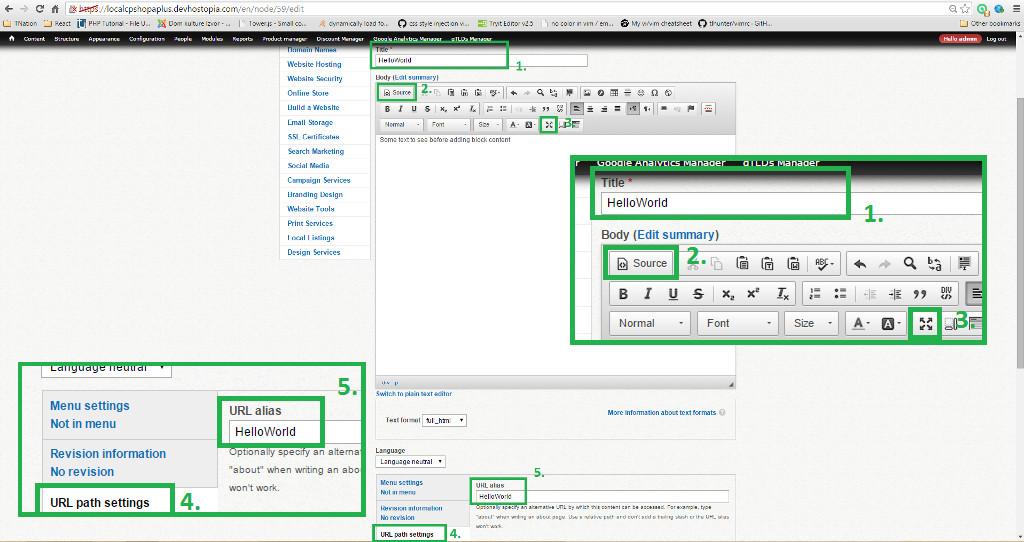


1. For work in drupal we need to login in drupal admin control panel. ( login link is: site\_link/sushilogin )

When you successfully login you will see black admin menu in top of page.

2. We wish to make new page on our site. We need to do next steps to accomplish that.

We need to chose "Add content". "Add content" is section in "Content" from admin menu.

Picture 2

Now we are on page where we can make other pages for our site.

1. "Title" is required field. In list of all pages we will recognise our page by the text we write in Title filed. In this example it will be HelloWorld.

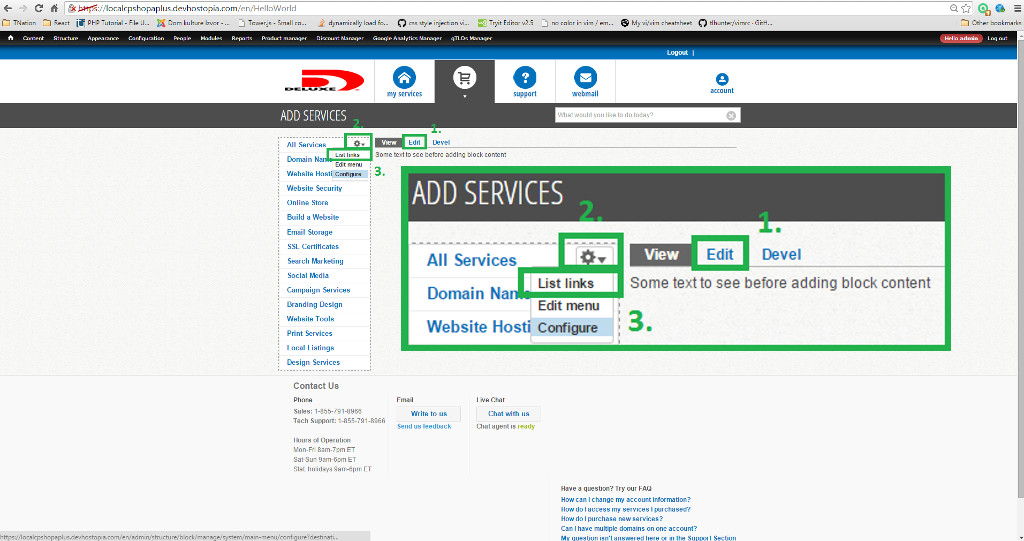
2. Click on "Source" to write code.

3. We can view code in bigger area if we click on this button.

4. We can make out URL path.

5. For "URL alias" we will put HelloWorld. Default URL for pages in drupal is node/page\_number (example: node/59 ).

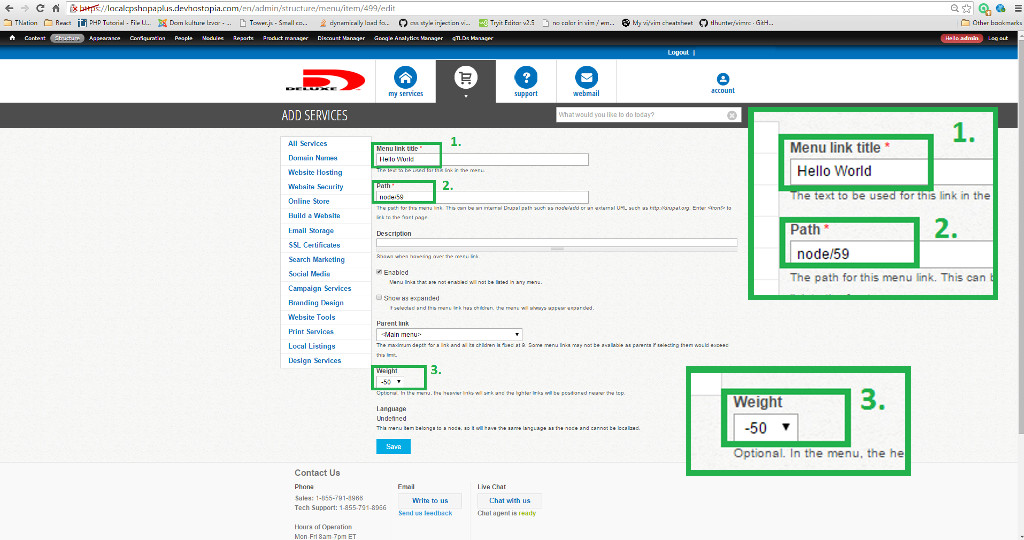
Save all changes, save button is in bottom of the page. If we want to see how our page look we can enter next url *site\_link/HelloWorld or site\_link/node/59*

Picture 3

1. If we want to change look of our page we can click on "Edit" otion. Later we will delete next text "Some text to see before adding block content" and put some blocks in our page.

2. Lets make our page wisible in side menu. Move your mouse point in top right corner of side menu, you will see "gear wheel".

1. Click on "gear wheel" and chose "List links", after redirection click on "Add link".

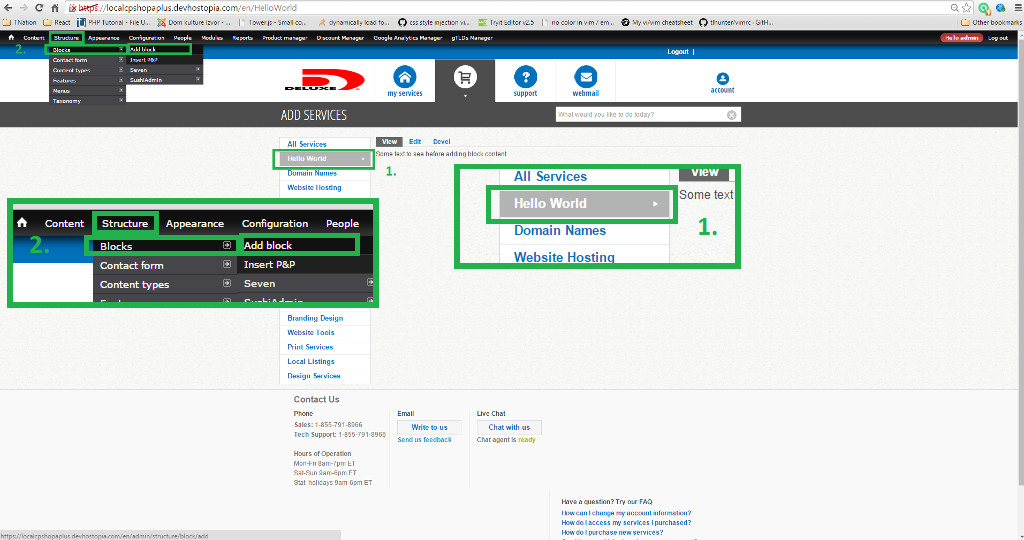
Picture 4

1. In "Menu link title" write down text that will be shown in side menu, in our example it will be "Hello World"

2. For field "Path" we will write "node/59" to show you that we can use default URL the same way as alias URL

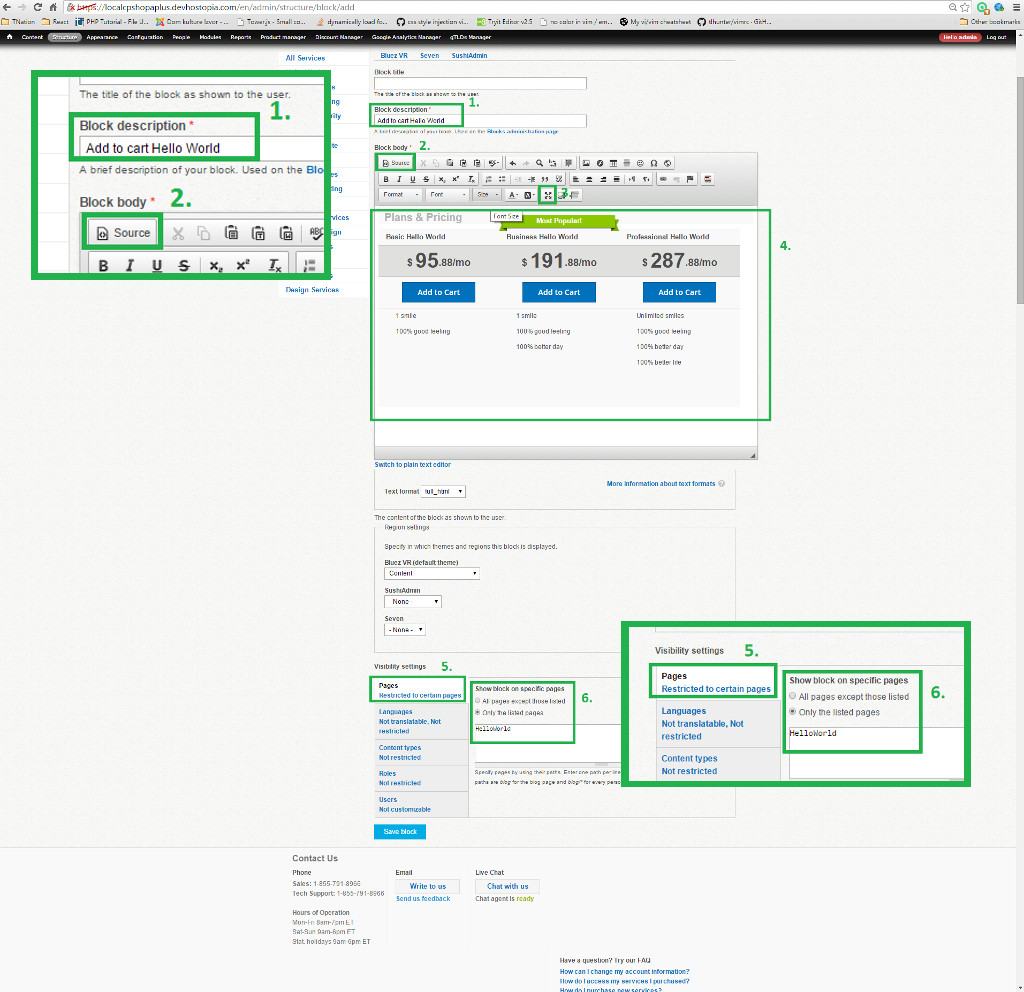
3. We wont to show "Hello world" after "All Services" so we will chose "Weight" = "-50"

Save all changes, save button is in bottom of the page.

Picture 5

1. As we can see we have "Hello World" in our side menu.

2. Now we want to make couple of blocks in our HelloWorld page. We need to chose "Structure" from admin menu than "Block", and click on "Add block".

Picture 6

1. "Block description" is field for name of this block, we can find our block by this name in list of all blocks.

2. and 3. are same steps from Picture 2

4. Write your code here.

5. In "Pages" settings we can specify on which pages our block will be shown

6. We want to display our block only on page HelloWorld so we are chose "Only the listed pages" and in text area we can write URL HelloWorld or node/59

Save all changes, save button is in bottom of the page.

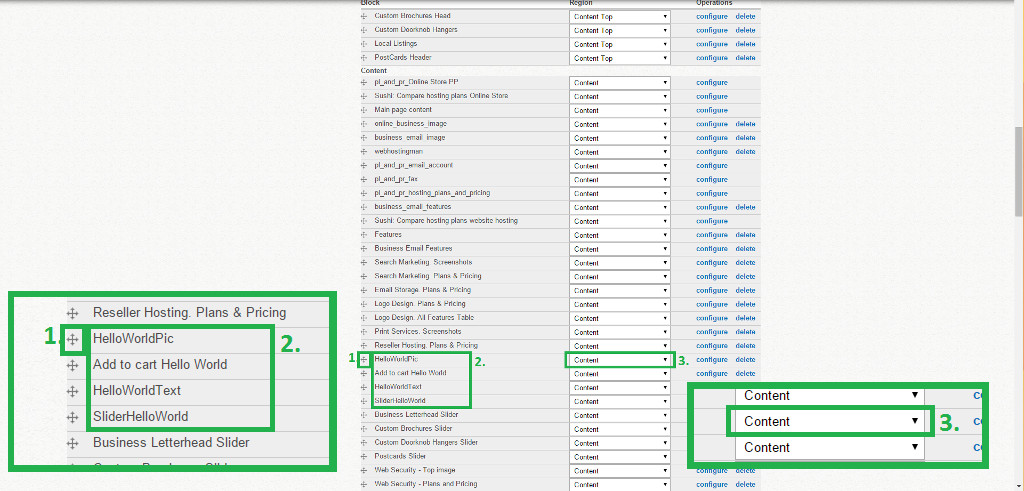
We will make couple of block and include them in HelloWorld page.

Picture 7



1. Lets find next block: "button back block" and make him wisable in our page.

Picture 8



Now let set region amd order of our block in page.

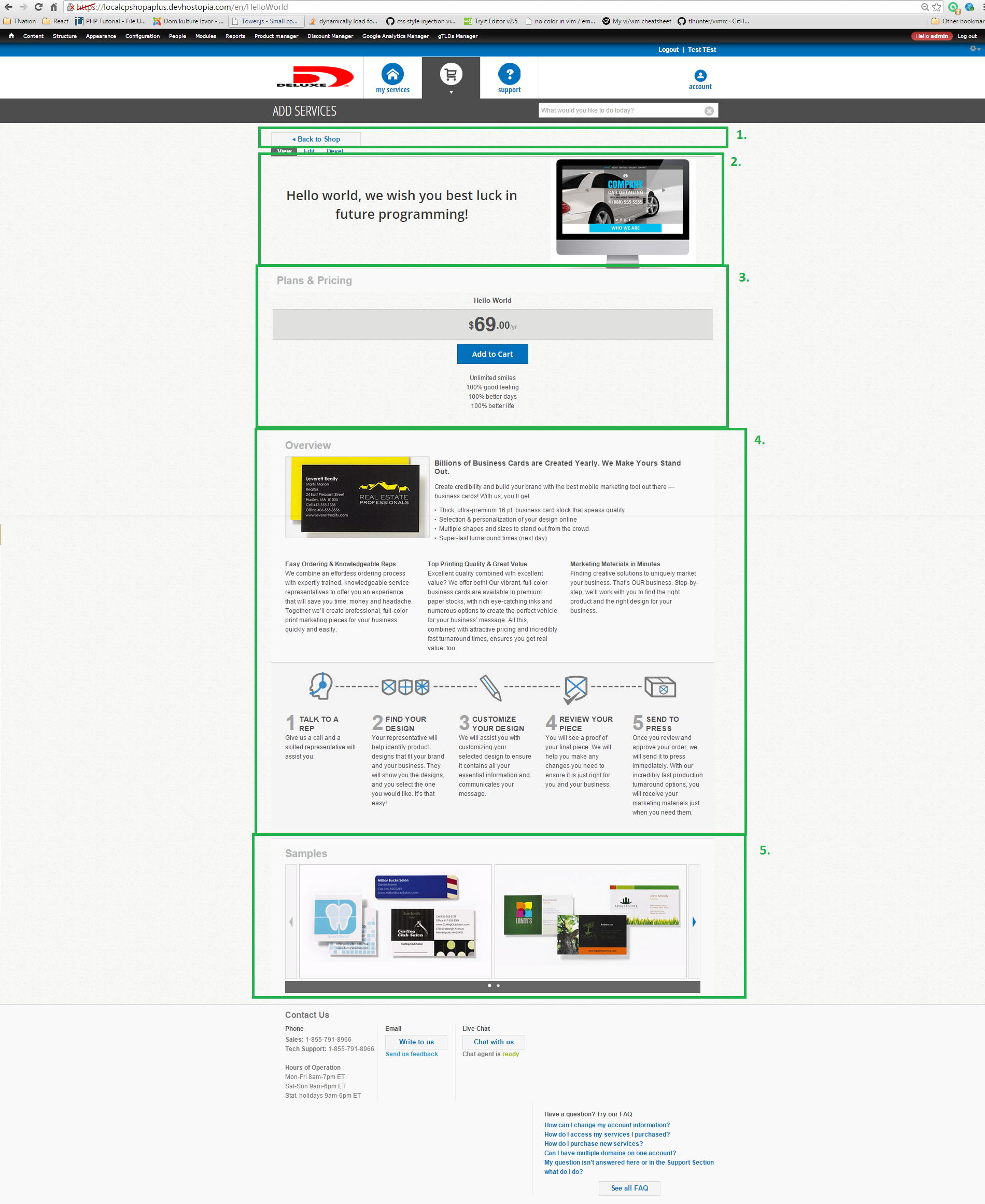
Chose "Structure" from admin menu and click on "Block". Now we will get the list of all blocks.

1. Click and hold left mouse button. Now you can move that block and place it to be shown before other blocks that are behind him, we will make block HelloWorldPic to be displayed first in HelloWorld page.

2. We can see the order of our blocks that will be displayed in page HelloWorld ( or in any other page that include some of our blocks )

3. We will chose "Content" as region where we will show our blocks.

After all changes our page look like this:

Picture 9

1. Blok: button back block

2. Blok: HelloWorldPic

3. Blok: Add to cart Hello World

4. Blok: HelloWorldText

5. Blok: SliderHelloWorld

File name - JS\_Code.js

/\*

We will explain buying flow for product "Domain Guard" as product for example "HelloWorld" since this “Domain Guard” product will have to be in DataBase and “HelloWorld” is not:

1. Go to "HelloWorld" page and click on button "Buy".

Button "Buy" have action on click: onclick = "goToCart('YDOMAINGUARD')"

We can find this function in "singup.js" ( location: "signup/v1/js/signup.js" )

\*/

function goToCart(prod){

popUp('#wait\_animation');

switch(prod){

case 'YDOMAINGUARD':

.

.

.

case 'MPINNACLECART':

//See this.DomainServiceSetup

check\_domains = Signup.DomainServiceSetup;

//Initialisation ( see DomainServiceSetup.init() )

check\_domains.init(prod,function(domains){

//if user don't have domain without "Domain Guard" in this account show him error message

if( domains.trueDomains.length == 0 ){

Signup.showError('To be able to buy this product, please buy a Domain Name first.');

popUpHide('#wait\_animation');

return false;

}

//Redirect to cart ( In our example URL is: "site\_link/cart#addProduct=YDOMAINGUARD;" )

window.location.href = Config.cart\_url + "#addProduct="+prod + ';';

});

break;

default:

//Redirect to cart

window.location.href = Config.cart\_url + "#addProduct="+prod + ';';

break;

}

}

this.DomainServiceSetup = new function(){

this.domains = [];

this.filteredDomains = [];

this.trueDomains = [];

var self = this;

//Get all Services

this.getServices = function(type) {

return $j.ajax({

type: 'GET',

/\* services.json is list of domains that have services ( "Domain Guard" is one of services ) with next informations: Id, Domain, Code, Ptype, Atype, Basecode, Name, Description, OptionString, NextBillingDate.\*/

url: '/api/rrad/db/account/services.json',

dataType: 'json',

async: true,

global: false,

success: function(data) {

self.filteredDomains = [];

//For every service check if it is YDOMAINGUARD

for ( i in data ){

if( data[i].Code.match( new RegExp(type) ) ) {

//Put Domains without YDOMAINGUARD in self.filteredDomains

if( $j.inArray( data[i].Domain,self.filteredDomains ) == -1 ){

self.filteredDomains.push(data[i].Domain);

}

}

}

},

error: function(xhr, desc, obj) {

console.log(desc);

},

complete: function() {

}

});

};

//Get all domains with Atype = R from database

this.getDomains = function(type){

return $j.ajax({

type: 'GET',

/\*domains.json with parameters Domain, ProductCode and atype=R is list of domains that have services with next informations: Domain, ProductCode. \*/

url: '/api/rrad/db/web/domains.json?q=Domain,ProductCode&atype=R',

dataType: 'json',

async: true,

global: false,

success: function(data) {

self.domains = [];

for ( i in data ){

var srvice = data[i].ProductCode.match(new RegExp(type));

//Dont add domains in list that already have YDOMAINGUARD

if( !srvice ) {

self.domains.push(data[i]);

}

}

},

error: function(xhr, desc, obj) {

console.log(desc);

},

complete: function() {

}

});

};

//Build select with domains as options

this.buildDomains = function(type){

if( isCartPage() ) {

$j('#domains\_list').hide();

$j('#email\_list').hide();

$j('#ssl\_domains\_list').show();

$j('select#ssl\_domain').addClass('su-select');

//checkDomains will provide us with domains that we need to show in options

var domains = self.checkDomains(type);

$j('select#ssl\_domain').html('');

for( var i=0;i<domains.length;i++ ){

$j('select#ssl\_domain').append('<option value="' + domains[i].Domain + '">' + domains[i].Domain + '</option>');

}

$j('select#ssl\_domain').select2();

}

};

//checkDomains will provide us with domains for specific service that we need

this.checkDomains = function(type) {

var domains = [];

switch(type) {

case 'YDOMAINGUARD':

case 'YDOMAINMONITOR':

for( var i=0;i<self.domains.length;i++ ){

//add domains only if it have ".com" or ".net" TLD and without "Domain Guard"

if( self.domains[i].Domain.match(/\.(com|net)$/i) && self.filteredDomains.indexOf(self.domains[i].Domain) == -1 ){

domains.push(self.domains[i]);

}

}

break;

default:

//For default (any other service) show all domains that are returned with self.getDomains(type)

for( var i=0;i<self.domains.length;i++ ){

domains.push(self.domains[i]);

}

break;

}

return domains;

};

//Initialisation

this.init = function(type,callback){

$j.when(self.getDomains(type),self.getServices(type)).then(function(data1,data2){

self.buildDomains(type);

self.trueDomains = self.checkDomains(type);

if(callback){

callback(self);

}

},function(data1,data2){

self.buildDomains(type);

self.trueDomains = self.checkDomains(type);

if(callback){

callback(self);

}

});

};

};

…

/\*

After clicking on "Next" button:

we trigger this part of code that is written in drupal:

\*/

$j('#wizard\_domains').steps("next");

//steps() is jQuery plug-in.

//Price and name of products are filled with in this code

//You can find this code in "singup.js"

$j(function(){

$j('#cart\_count').html(Signup.cartCount());

var h;

var last\_step = 0;

var viseted\_steps = [];

var last\_visit = localStorage['last\_visit'] ? localStorage['last\_visit'] : '0000-00-00';

var cur\_date = new Date();

var last\_visit = new Date(last\_visit);

var date\_diff = (cur\_date.getTime() - last\_visit.getTime()) / (1000 \* 60 \* 60 \* 24);

$j('<div/>').addClass('shadowOverlay').hide().

append($j('<div/>').addClass('shadowLoader')).

appendTo('body');

if (!date\_diff || date\_diff > 2){

localStorage.clear();

}

localStorage['last\_visit'] = cur\_date.toString('yyyy-MM-dd');

/\*Here, we will get our product code for "Domain Guard", YDOMAINGUARD, in product variable. \*/

var product = location.hash.match(/#addProduct=([^;]+);/) || location.search.match(/\?addProduct=([^&]+)(&productid=(\d+))(&productname=([^&]+))/);

var id = product && product[3];

var productName = product && product[5];

product = product && product[1];

/\* In this if we will get price and package Name for all products in cart, in our example we have only "Domain Guard" in cart \*/

if (product && isCartPage() && Config.Signup.products[product]) {

var packName=Config.Signup.products[product].Name;

if(packName.indexOf('(') != -1){

packName = packName.substr(0,packName.indexOf('('));

}

$j('#product\_type').html(packName);

var type = Config.productTypes[product];

if (type) {

Signup.CartSummary.items= {};

Signup.clearLocalStorage();

switch (type) {

case 'email-only':

...

/\* This is section that we need for Domain Guard that will take all informations that we need to show in cart \*/

case 'service':

Signup.CartSummary.items.services = {};

Signup.CartSummary.items.services[product] = {};

var service = {

'price': Config.Signup.products[product].USAPrice,

'FreeDomainRegsCount': 0,

'accounts': 0,

'code': product,

'description': Config.Signup.products[product].Description,

'period': 1

};

Signup.CartSummary.items.services[product] = service;

console.log(service);

Signup.CartSummary.setPrice(Signup.CartSummary.items.services[product]);

if(isCartPage()) {

var d\_service = Signup.DomainServiceSetup;

d\_service.init(product);

/\* Signup.Cart will build cart informations that will be send to BulkBuy for purchase \*/

Signup.Cart.init();

return false;

}

break;

default:

break;

}

}

}

.

.

.

});

# RRAD

## About RRAD

RRAD environment is the part that is based on Rest API. It serves as a communication and processing of data between the front end (control panel(wsos,ring,retails)), database and WAIRS system. Rrad is centralized type, so that different applications and different control panels can access it at the same call. Therefore maintenance and data processing is done in one place. DB is database hosted on server and can be accessed using mysql connection and queries.

Rest API is used for sending requests from front end on RRAD and WAIRS.

RRAD is used for data processing.

WAIRS is used for communication between RRAD, banks, opensrs and third party system.

MAD is a system for sending, updating and tracking mails.

WAD is a system for domains.

## RRAD File System

RRAD contains following directories:

* /api/
  + vhosts/rraddev/htdocs/
  + dns/
  + buy/
  + db/
    - stefan/
      * index.php
      * .htaccess
  + rrad/
* /includes/
  + vhosts/rraddev/includes
  + helloworld.php
  + arch\_helloworld.php
  + tables/
    - w3Control.testTable.json
* /development/
* /home/

All requests to rrad are in the *api* directory. Each rest api is located in this directory.

*Includes* directory contains all the global files that are included in the php script. There are shared functionality files that can be included in all scripts.

*Development* directory contains all files used by developer.

In *home* directory are placed all subdirectories of all users who have an account on rrad.

All api calls for information about domains, such as when they are created, when they point out, etc, are placed in *dns* directory. *Dns* continues to communicate with the database and WAIRS.

All api calls to communicate with the database will be sent to the *db* directory. In this directory are placed all individual application subdirectories that work communication.

*Rrad* directory is used for local communication with the database, and it is not linked via URL.

*Rewrite code* that opens the index page when receives request, is stored in .*htaccess* file.

## Simple rest api request with database access

File name – index.php

<?php

###########################

# Hostopia WebsiteOS API #

###########################

# coupons database access #

###########################

/\* If you want to see the errors, remove the comments from the following two lines of code.\*/

#ini\_set('error\_reporting', E\_ALL);

#ini\_set('display\_errors', true);

define('INCHOME', '../../../includes/');

define('LIMIT', '14');

/\* include all the nececery files in order to work with DB and Authentication \*/

require(INCHOME . 'HTTP.php');

require(INCHOME . 'AuthCard.php');

require(INCHOME . 'RestClient.php');

require(INCHOME . 'RestServer.php');

require(INCHOME . 'DbServer.php');

require(INCHOME . 'DbBase.php');

require(INCHOME . 'w3Control.php');

/\* include helloworld.php that contain all data nececery for DB login \*/

require(INCHOME . 'helloworld.php');

function logit($txt)

{

}

function logdb($txt)

{

}

class HelloWorldApi extends DbServer

{

/\*parsing request url in nececery sections

example request:

https://ddencic-wsos.devhostopia.com/api/db/helloworld/testdata/darko/12345

$db = db

$table= helloworld

$function\_name = "testdata" function will be called based on request call in this case we are showing for POST

$path = /darko/12345 parameters that will be passed to called function

\*/

function parseURL($url)

{

list($a, $d, $db, $table, $function\_name, $path) = explode('/', $url, 6);

$this->path = $path;

switch($this->method) /\*Looking for request method (GET, POST, PUT)\*/

{

case 'GET':

$this->setMethod('get'.$function\_name, explode('/', $path));

/\*calling funcion for GET request\*/

break;

case 'PUT':

$this->setMethod('put'.$function\_name, explode('/', $path));

/\*calling funcion for PUT request\*/

break;

case 'POST':

$this->setMethod('post'.$function\_name, explode('/', $path));

/\*calling funcion for POST request\*/

break;

default:

return false;

}

# unified authentication system

function authenticate($uname, $pass)

{

if ( !$this->validate($uname, $pass) )

return false;

$this->helloworld = new helloworld();

return true;

}

function getAllTestData (){

if ( $this->baselvl != 5 ) /\*will work only for account level 500\*/

{

$this->forbid('No access to specified method');

}

$table[] = $this->select('w3Control.testTable');

/\*define from what table and DB to get data\*/

$from = $this->getFrom($table);

$sl = "SELECT id,name,age,UsePayment " .$from;

/\*build select query\*/

$qu = "$sl";

/\*combine select and from queries\*/

try

{

$res = $this->helloworld->assocQ($qu, true);

/\*run query in sql\*/

}

catch(Exception $e){

$this->svrfail($e->getMessage());

}

if ( !$res ) /\*if false it means no data in table \*/

return array('status'=> false,'message'=>'No values found for this user');

return array('status'=> true,'message'=>'Found values','results'=>$res);

/\*return json with all data collected from sql request \*/

}

/\* request example /api/db/helloworld/testdata/ wil return all data from table for given id\*/

function getTestData ($id){

if(empty($id)) /\*validate if id is passed to function\*/

{

return array('status'=> false,'message'=>'Id no present');

}

if ( $this->baselvl != 5 ) /\*will work only for account level 500\*/

{

$this->forbid('No access to specified method');

}

$table[] = $this->select('w3Control.testTable');

/\*define from what table and DB to get data\*/

$from = $this->getFrom($table);

$wh = "WHERE id= $id";

/\*build where query\*/

$sl = "SELECT id,name,age,UsePayment " .$from;

/\*build select query\*/

$qu = "$sl $wh";

/\*combine select,where and from queries\*/

try

{

$res = $this->helloworld->assocQ($qu, true);/\*run query in sql\*/

}

catch(Exception $e){

$this->svrfail($e->getMessage());

}

if ( !$res ) /\*if false it means no data in table \*/

return array('status'=> false,'message'=>'No values found for this id');

return array('status'=> true,'message'=>'Found values','results'=>$res);

/\*return sucess value with values \*/

}

/\* request type PUT is used for update

request example /api/db/helloworld/testdata/ wil UPDATE field UsePayment for given id inside testTable table\*/

function putTestData( $id, $usePayment){

if ( $this->baselvl != 5 ) /\*will work only for account level 500\*/

{

$this->forbid('No access to specified method');

}

$table[] = $this->select('w3Control.testTable');

/\*define in what table and DB to update data\*/

$condition[] = "id={$id}";

$where = $this->getWhere($condition);

/\*build where query\*/

$updates = array();

$updates[] = "UsePayment='" . $usePayment . "'";

/\*build update fields query\*/

$update = $this->getUpdate($table, $updates);

/\*combine update queries\*/

$updateQ = "$update $where LIMIT 1";

/\*combine update and where queries\*/

try

{

$result = $this->helloworld->updateQ($updateQ, true);

/\*run query in sql\*/

}

catch(Exception $e){

$this->svrfail($e->getMessage());

}

if ( !$result ) /\*if false it means no data were updated in table \*/

return array('status'=> false,'message'=>'No value updated');

return array('status'=> true,'message'=>'Updated value');

/\*return sucess value \*/

}

/\* request type POST is used for update

request example /api/db/helloworld/Name/Age wil INSERT data for given name and age inside testTable table\*/

function postTestData( $name , $age){

if ( $this->baselvl != 5 ) /\*will work only for account level 500\*/

{

$this->forbid('No access to specified method');

}

$table[] = $this->select('w3Control.testTable');

/\*define in what table and DB to insert data\*/

$insert = $this->insert($table);

$this->addInsert($insert, 'id', null);

/\*build insert fields query for autoincrement type field\*/

$this->addInsert($insert, 'name', $name);

/\*build insert fields query\*/

$this->addInsert($insert, 'age', $age);

/\*build insert fields query\*/

$this->addInsert($insert, 'UsePayment', 'NO');

/\*build insert fields with default value\*/

$qu = $this->getInsert($table, $insert);

/\*combine insert and into queries\*/

try

{

$result = $this->helloworld->insertQ($qu, true);

/\*run query in sql\*/

}

catch(Exception $e){

$this->svrfail($e->getMessage());

}

if ( !$result ) /\*if false it means no data were inserted in table \*/

return array('status'=> false,'message'=>'No value created');

return array('status'=> true,'message'=>'Created value');

/\*return sucess value \*/

}

}

$server = new HelloWorldApi();

$server->handle();

?>

File name – helloworld.php

<?php

#2009-1-7 using slave

define("DB\_MASTER", (1 << 0)); #master db connection ok

define("DB\_SLAVE", (1 << 1)); #slave db connection ok, if master ok, slave always ok

define("CPM\_DBNAME", 'w3Control');

require('arch\_helloworld.php');

class helloworld extends DbBase

{

protected $\_\_mysql;

//2009-1-7 using slave

protected $\_\_mysqlS;

protected $db\_mode;

protected $flags;

var $id;

var $ip;

var $username;

var $salesrep;

var $clientid;

function setFlag($flag)

{

$this->flags |= $flag;

}

function isFlagged($flag)

{

return ($this->flags & $flag);

}

function isLocal()

{

return ($this->flags & UF\_FROMWEBSVR)

| ($this->flags & UF\_FROMADMIN);

}

function isSlaveOK()

{

return ($this->db\_mode & DB\_SLAVE);

}

function isMasterOK()

{

return ($this->db\_mode & DB\_MASTER);

}

function \_\_construct()

{

$this->ip = qes($\_SERVER["REMOTE\_ADDR"]);

//2009-1-7 using slave

$this->db\_mode=0;

# changed by victor and derek, we changed it so that it always

# connects to the master. But in the future, we will add a flag

# to specify whether or not we need write access - mar 16 2010

if ( $this->\_\_mysql=mysql\_connect(DBHOST, DBUSER, DBPASS, true) )

{

#logit("Failed to connect to slave. Connected to master");

$this->\_\_mysqlS=&$this->\_\_mysql;

$this->db\_mode |= DB\_MASTER;

mysql\_select\_db(CPM\_DBNAME, $this->\_\_mysql);

}

else

if ($this->\_\_mysqlS=mysql\_connect(DBSHOST, DBSUSER, DBSPASS, true))

{

logit("Failed to connect to master. Connected to slave");

$this->\_\_mysql=&$this->\_\_mysqlS;

$this->db\_mode |= DB\_SLAVE;

mysql\_select\_db(CPM\_DBNAME, $this->\_\_mysql);

}

else

if ( !$this->isSlaveOK() )

{

logit("Database: Failed to connect to slave or master: ".mysql\_error());

throw new SoapFault("Database", "Failed to connect to slave and

master: ".mysql\_error());

}

}

}

?>

File name - arch\_helloworld.php

<?php

#define dev db

define("WN\_MYSQL\_HOST", "admin.devhostopia.com");

define("WN\_MYSQL\_USER", "purple");

define("WN\_MYSQL\_PASS", "fG$3!dH");

define("WN\_MYSQL\_DB", "w3Control");

?>

File name - w3Control.testTable.json

{

"table":"testTable", "as":"P",

"fields":

[

["id", 1,5,""],

["name", 1,5,""],

["age", 1,5,""],

["UsePayment", 1,5,""]

]

}

# Repositories

## GIT

### About GIT

Git is distributed version control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflow. Every Git working directory is full-fladged repository with complete history and full version-tracking capabilities, independent of network access or a central server.

Tree main sections of the git project:

* Git directory
* Working directory
* Staging area

The Git directory is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you clone a repository from another computer.

The working directory is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.

The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit. It’s sometimes referred to as the “index”, but it’s also common to refer to it as the staging area.

The basic Git workflow goes something like this:

1. You modify files in your working directory.
2. You stage the files, adding snapshots of them to your staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

### GIT Basics

Some commands for setting your identity:

$ git config --global user.name ‘name’

$ git config --global user.email ‘email’

$ git config --global user.editor ‘editor’

$ git config --list // List all the settings

If you ever need help while using Git, there are three ways to get the manual page (manpage) help for any of the Git commands:

$ git help *<verb>*

$ git *<verb>* --help

$ man git*-<verb>*

Command that creates a new subdirectory .git with files for git repository:

$ git init

After you initialize a repository must specify the files you want to track:

$ git add *file*

$ git add *file*

and than enter command:

$ git commit –m ‘Description of this commit’

If you want to get copy of an existing Git repository, command that you need is:

$ git clone *url*

Each file in your working directory can be in one of two states: tracked or untracked.

Tracked files are files that were in the last snapshot; they can be unmodified, modified, or staged. Untracked files are everything else – any files in your working directory that were not in your last snapshot and are not in your staging area. When you first clone a repository, all of your files will be tracked and unmodified because you just checked them out and haven’t edited anything.

The main tool you use to determine which files are in which state is the command:

$ git status

The command tells you which branch you’re on and informs you that it has not diverged from the same branch on the server.

If the git status command is too vaque for you – you want to know exactly what you changed, not just which files were changed – you can use command:

$ git diff

Although git status answers those questions very generally by listing the file names, git diff shows you the exact lines added and removed.

After you have created several commits, or if you have cloned a repository with an existing commit history, you’ll probably want to look back to see what has happened. The most basic and powerful tool to do this is the command:

$ git log

At any stage, you may want to undo something. If you want to try that commit again, you can run commit with the --amend option:

$ git commit --amend

If you want to remove a file from the stage you can use command:

$ git reset file

To add a new remote Git repository as a shortname you can reference easily, run command:

$ git remote add *short\_name* *url*

To see which remote servers you have configured, you can run the command:

$ git remote

It lists the shortnames of each remote handle you’ve specified.

You can also specify –v, which shows you the URLs that Git has stored for the shortname to be used when reading and writing to that remote.

To add a new remote Git repository as a shortname you can reference easily, run:

$ git remote add *short\_name url*

To get data from your remote projects, you can run:

$ git fetch *remote\_name*

The command goes out to that remote project and pulls down all the data from that remote project that you don’t have yet. After you do this, you should have references to all the branches from that remote, which you can merge in or inspect at any time.

If you clone a repository, the command automatically adds that remote repository under the name “origin”. So, git fetch origin fetches any new work that has been pushed to that server since you cloned (or last fetched from) it.

Running git pull generally fetches data fjrom the server you originally cloned from and automatically tries to merge it into the code you’re currently working on.

When you have your project at a point that you want to share, you have to push it upstream. The command for this is simple:git push remote\_name branch\_name. If you want to push your master branch to your origin server (agjain, cloning generally sets up both of those names for you automatically), then you can run this to push any commits you’ve done back up to the server:

$git push origin master

If you want to see more information about a particular remote, you can use:

$ git remote show *remote\_name*

If you want to rename a reference you can run git remote rename to change a remote’s shortname:

$ git remote rename *old\_name new\_name*

You can easily set up an alias for each command using git config:

$ git config --global alias.ci commit

Nearly every VCS has some form of branching support. Branching means you diverge from the main line of development and continue to do work without messing with that main line.

What happens if you create a new branch? Well, doing so creates a new pointer for you to move around. Let’s say you create a new branch called testing:

$ git branch testing

This creates a new pointer at the same commit you’re currently on.

To switch to an existing branch, you run the git checkout  command. Let’s switch to the new testing branch:

$ git checkout testing

The git merge tool is used to merge one or more branches into the branch you have checked out. It will then advance the current branch to the result of the merge.

$ git checkout *branch\_name*

$ git merge master

Thie git cherry-pick command is used to take the change introduced in a single Git commit and try to re-introduce it as a new commit on the branch you’re currently on.

The git rebase command is basically an automated cherry-pick. It determines a series of commits and then cherry-picks them one by one in the same order somewhere else.

### GIT WORKFLOW

Bellow we will show some of the GIT flows that are used in different project (this is just some template of flow,they deffer from project to project)

GIT WORKFLOW FOR NEW STORY ON DRUPAL

(cp\_deluxe\_submodule and cpaplus\_sushi\_submodule )

1. git checkout **master**
2. git pull
3. git checkout –b *branch\_name*
4. do changes
5. git add files
6. git commit –m ‘*commit\_message*’ (write the number of the bug or the number of stories)
7. git fetch origin && git rebase –i origin/master
8. git push –u origin *branch\_name*

SEND TO CODE REVIEW

* After code review failed

1. git checkout *branch\_name*
2. git pull
3. do changes
4. git add files
5. git commit ‘*commit message #2*’
6. git push –u origin *branch\_name*

UPDATE CODE REVIEW

* After code review pass

MERGE WITH DEV

1. git checkout dev
2. git pull
3. git merge *branch\_name*
4. resolve conflict if necessary
5. git push –u origin dev

MERGE WITH QA

To be discussed

GIT WORKFLOW FOR NEW STORY ON MOBILE CP

1. git checkout master
2. git pull
3. git checkout -b *branch\_name*
4. do changes
5. git add files
6. git commit -m ‘*commit message*’ (write the number of the bug or the number of stories)
7. git checkout master
8. git pull
9. git checkout *branch\_name*
10. git rebase -i origin/master
11. git push -u origin *branch\_name*

SEND TO CODE REVIEW

* After code review failed

1. git checkout *branch\_name*
2. git pull
3. do changes
4. git add files
5. git commit -m ‘*commit message #2*’
6. git push -u origin *branch\_name*

UPDATE CODE REVIEW

* After code review pass

MERGE WITH QA

1. git checkout QA-testing
2. git pull
3. git merge *branch\_name*
4. resolve conflict if necessary
5. git push -u origin QA-testing

SEND REQUEST FOR SYNC

MERGE WITH STAGE (BASIC)

1. git checkout master
2. git pull
3. git checkout -b darko-release
4. git push -u origin darko-release
5. git checkout *every\_branch*
6. git checkout darko-release
7. git pull
8. git merge *every\_branch*
9. resolve conflict if necessary
10. *check is everything ok ( commits, code validator and functionality )*
11. git push -u origin darko-release
12. git checkout release
13. git pull
14. git merge darko-release
15. resolve conflict if necessary
16. git push -u origin release
17. git checkout master
18. git pull
19. git merge darko-release
20. resolve conflict if necessary
21. git push -u origin master

GIT WORKFLOW FOR NEW STORY ON RING AND APPELTS

1. git checkout master
2. git pull
3. git checkout -b *branch\_name*
4. do changes
5. git add files
6. git commit -m ‘*commit message*’ (write the number of the bug or the number of stories)
7. git checkout master
8. git pull
9. git checkout *branch\_name*
10. git rebase -i origin/master
11. git push -u origin *branch\_name*

SEND TO CODE REVIEW

* After code review failed

1. git checkout branch\_name
2. git pull
3. do changes
4. git add files
5. git commit -m ‘commit message #2’
6. git push -u origin branch\_name

UPDATE CODE REVIEW

* After code review pass

MERGE WITH QA

To be discussed

## SVN

### About SVN

Subversion is a free/open source version control system (VCS). That is, Subversion manages files and directories, and the changes made to them, over time. This allows you to recover older versions of your data or examine the history of how your data changed. In this regard, many people think of a version control system as a sort of “time machine.”

Subversion can operate across networks, which allows it to be used by people on different computers. At some level, the ability for various people to modify and manage the same set of data from their respective locations fosters collaboration. Progress can occur more quickly without a single conduit through which all modifications must occur. And because the work is versioned, you need not fear that quality is the trade-off for losing that conduit—if some incorrect change is made to the data, just undo that change.

### SVN Basics

Subversion commands can be run from a command shell such as Bash on Linux. The subversion client command is svn followed by optional sub-commands, options, and arguments.

Show the program version and modules:

$ svn --version

**Getting the source code**

From the parent directory of where you want the working copy. In this example the aoo-trunk directory will be created if it does not exist:

$ svn co https://svn.apache.org/repos/asf/openoffice/trunk aoo-trunk

A aoo-trunk/tools

A aoo-trunk/tools/dev

A aoo-trunk/tools/dev/fetch-all-cws.sh

A aoo-trunk/tools/dev/cws-list.txt

A aoo-trunk/tools/dev/fetch-all-web.sh

A aoo-trunk/tools/dev/web-list.txt

A aoo-trunk/tools/dev/single-hg.sh

Checked out revision 1145818.

A" indicates file or directory is "Added" to working copy

**Basic work cycle**

* Update your working copy - For this you use the *svn update* command
* Make changes - For this you may edit files in an editor, or use the *svn add, svn delete, svn copy, svn-move* commands
* Review Changes - For this you use the *svn status* and *svn diff*
* Fix Mistakes - Make additional edits to files or you can use the svn revert to restore files or directories to an unmodified state
* Resolve Conflicts - There is a chance others have committed changes while you have been changing your working copy. You should run the *svn update* command to bring your copy up to date. This may create a local conflict where someone may have added a file with a name that you also want to add, or may have made changes to the same line of a file as you. For this use the *svn resolve* command.
* Publish Changes - For this you use the *svn commit* command

Adding a file

After creating a file ‘test-file.txt’ in the working copy:

$ svn status

? test-file.txt

? indicates test-file.txt is not under version control.

Scheduling a file for adding to repository:

$ svn add test-file.txt

A test-file.txt

$ svn status

A test-file.txt

A indicates file is scheduled for addition.

Running a *diff*

$ svn diff

Index: test-file.txt

===================================================================

--- test-file.txt (revision 0)

+++ test-file.txt (revision 0)

@@ -0,0 +1 @@

+This is a test file for svn-basics.

Property changes on: test-file.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Added: svn:eol-style

+ native

Commiting a file

$ svn commit test-file.txt -m "added test-file.txt"

Adding test-file.txt

Transmitting file data .

Committed revision 2.

Update the working copy

$ svn update

U test-file.txt

Updated to revision 3.

"U" indicates an "Update" to a file or directory

Modify the file (this example uses the vim editor)

$ vim test-file.txt

Check the status

$ svn status

M test-file.txt

"M" indicates the file has been "Modified"

$ svn diff

Index: test-file.txt

===================================================================

--- test-file.txt (revision 3)

+++ test-file.txt (working copy)

@@ -1,2 +1,3 @@

This is a test file for svn-basics.

This is a new line added by someone else.

+This line added by me.

Resolving conflicts

Suppose someone edits the same line as you before you commit

$ svn update

Conflict discovered in 'test-file.txt'.

Select: (p) postpone, (df) diff-full, (e) edit,

(mc) mine-conflict, (tc) theirs-conflict,

(s) show all options:

This is just like if you had ran the svn resolve command

Selecting df displays this:

--- .svn/text-base/test-file.txt.svn-base Sun Jul 17 17:38:52 2011

+++ .svn/tmp/test-file.txt.tmp Sun Jul 17 21:35:09 2011

@@ -1,2 +1,7 @@

This is a test file for svn-basics.

This is a new line added by someone else.

+<<<<<<< .mine

+This line added by me.

+=======

+This line is added by someone else also.

+>>>>>>> .r4

Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,

(mc) mine-conflict, (tc) theirs-conflict,

(s) show all options:

If you choose e, Subversion will launch an editor with both sets of changes included for you to edit. You can save your changes in the editor and then select r (for resolved).

G test-file.txt

Updated to revision 4.

"G" indicates "merGed"

Commiting changes

Only Committers can commit directly to the repository. The following example shows using your Apache ID and password.

$ svn commit test-file.c --username your-name --password your-password \

-m "added new C file"

Sending test-file.txt

Transmitting file data .

Committed revision 5.

In general, you may not have to include always your username or password if you do a proper setup of your ssh key or have subversion store the password.

Always check your changes with "svn diff" and "svn status". Also be careful to specify the files and/or directories you want to change, if you don't specify, SVN will commit all your changes.

Commit message

The examples in the previous sections use a simple commit message with the "-m" option.

This is fine for some quick testing or for large bulk commits of code that you wrote.

Log comments are important. Information like author, where the change start/ends, the date, the bugzilla issue, and the author don't really belong in the code as SVN can keep it much more effectively without altering the coding style. Always try to use a log file for your commits. The previous commit when done by an experienced committer should actually look like this:

$ svn ci -F test-log.txt test-file.c

Sending test-file.c

Transmitting file data .

Committed revision 5.

Merging changes to a branch

The first step is to do a check out of the stable, release, branch. The following examples use the AOO34 release branch, and assume you want to apply changes from trunk for a new release, maybe AOO341.

You can do a complete checkout of the release branch or you can save some space by using the "--depth=empty" option:

$ svn co --depth=empty https://svn.apache.org/repos/asf/openoffice/branches/AOO34 aoo-stable

U aoo-stable

Checked out revision 1347362.

This will put a placeholder branch for the AOO34 in directory "aoo-stable".

In the aoo-stable directory, you can keep saving space (rather convenient) until you reach the directory where you want to make changes:

$ svn up --depth=empty main

Updating 'main':

A main

Updated to revision 1347363.

$ svn up --depth=empty jvmfwk

Updating 'jvmfwk':

A jvmfwk

Updated to revision 1347366.

At this point, there are svn placeholder entries for /main/jvmfwk.

To do a complete checkout from there:

$ svn up --set-depth=infinity

Updating '.':

A source

A source/elements.hxx

A source/fwkbase.cxx

... (and so on)

Now merge the specific revision(s) you want (in this case r1333165):

$ svn merge -c1333165 https://svn.apache.org/repos/asf/openoffice/trunk/main/jvmfwk .

--- Merging r1333165 into '.':

U distributions/OpenOfficeorg/javavendors\_unx.xml

At this point, you have merged r1333165 into r1347366. (For release from trunk to a new release, it's likely you would be merging a higher revision to a lower existing release revision.)

After you finish merging, check your changes with "svn status" and "svn diff" and commit from the aoo-update directory:

$ svn commit -m "Merge r1329539, r1329547, 1333165 - Add Oracle as a Java vendor on unix." distributions/OpenOfficeorg/javavendors\_unx.xml distributions/OpenOfficeorg/javavendors\_freebsd.xml

Sending distributions/OpenOfficeorg/javavendors\_freebsd.xml

Sending distributions/OpenOfficeorg/javavendors\_unx.xml

Transmitting file data ..

Committed revision 1347377.

# Debugging

USUSAL FLOW WHEN DEBUGING (used mobile as platform):

* 1. First check if javascript code is ok on environment that bug was reported.
  2. Compare whether the bug is repeated at all environment, DEV, QA, STAGE (if code is presented there,and if bug is present).
  3. If javascript code is ok, check what data is being sent with ajax (see network tab in browser inspector).
  4. See response for data that was sent with ajax.
  5. If returned data is bad, go to rest api (rrad server) and check whether rest api which is being called is receiveing good informations.
  6. Debug rrad code using *file\_put\_contents()*, never use *print\_r*, *echo*, *die*, *var\_dump*…

Example(file\_put\_content(‘/home/ddencic/test.out’, “test = ”.print\_r($variable, 1).“\n”, FILE\_APPEND | FILE\_TEXT);)

* 1. If everything is as it should be, and data is being sent on WAIRS, see what WAIRS is sending in file /include/WairsClientCmd.php (debug using *file\_put\_contents()*).
  2. Write down WAIRS request and change it in order to turn debug on (command: ‘-d on’).

Ex before:

/services/websoftware/wairs/wairs\_4.0/remote\_request.cgi -h 2332 -pb64 ZG5hbWU9cWF0ZXN0aW5nLWRkLTE3MDkyMDE1MS5uZXQmYWRkaXRpb25hbHNlcnZpY2VzPVlTU0xDRVJUVFJVRUJVU0lEJmFjY291bnRpZD0xMTM1MDImVVNFUl9B[293/1170] BOYW1lPU1vYmlsZStBUExVUytDUCtTaG9wK1NlcnZpY2VzJmNvbW1hbmQ9YnV5X3NlcnZpY2VzJlVTRVJfUkVNT1RFX0FERFI9MjAwMSUzQTE4MTAlM0E1MDUwJTNBMjklM0ElM0ExNTY=

Ex after:

/services/websoftware/wairs/wairs\_4.0/remote\_request.cgi -h 2332 -d on -pb64 ZG5hbWU9cWF0ZXN0aW5nLWRkLTE3MDkyMDE1MS5uZXQmYWRkaXRpb25hbHNlcnZpY2VzPVlTU0xDRVJUVFJVRUJVU0lEJmFjY291bnRpZD0xMTM1MDImVVNFUl9B[293/1170]

BOYW1lPU1vYmlsZStBUExVUytDUCtTaG9wK1NlcnZpY2VzJmNvbW1hbmQ9YnV5X3NlcnZpY2VzJlVTRVJfUkVNT1RFX0FERFI9MjAwMSUzQTE4MTAlM0E1MDUwJTNBMjklM0ElM0ExNTY=

* 1. If the php code is working with database, debug sql query that is beeing send using *file\_put\_contents()*, and then that query try directly in MySQL database. Obsurve the result and ensure it is expected.