Code Samples

PHP Mysqli Query function

```
/**
    * Executes a Query and returns results
    *
    * @param string $sql The SQL to be executed
    * @param array $params The parameters to bind the query
    * @param bool $return A flag to determine if the results need to be returned
    * @return array An associate array of the query result set
    */

   public function query($sql, $params, $return) {

      //Set private variables
      $this->_sql      = $sql;
      $this->_params   = $params;
      $this->_return   = $return;

      //Instantiate a new reflection to mysqli_stmt
      $Reflect = new ReflectionClass('mysqli_stmt');

      //Call Prepared statement
      $stmt = $this->_con->prepare($this->_sql);

      //If there are parameters passed in
      if($this->_params) {

         //instance variables
         $paramsStack = array();
         $x = 0;

         //Assign ParameterStack by reference to the input parameter so bind_param works
         foreach($this->_params as $value) {
            $paramsStack[$x] = &$this->_params[$x++];
         }

         //Get an instance of the bind_param method
         $method = $Reflect->getMethod('bind_param');

         //Invoke the returned method and pass an array of parameters in
         $method->invokeArgs($stmt, $paramsStack);

      }

      //Execute the prepared statement
      $stmt->execute();

      //If the dataset should be returned
      if($return) {

         //get number of columns returned
         $columnCount = $stmt->field_count;

         //get number of rows returned
         $stmt->store_result();
         $rowCount = $stmt->num_rows;

         //call meta data functions to get column names
         $meta = $stmt->result_metadata();
```

```
        $field = $meta->fetch_fields();

        //Instantiate array for result set
        $resultStack = array();
        $result = array();

        //Build the associative array for the result set
        for($i = 0; $i < $columnCount; $i++) {
           $resultStack[$field[$i]->name] = '';
        }

        //Get an instance of the bind_result method
        $method = $Reflect->getMethod('bind_result');

        //Invoke the result method
        $method->invokeArgs($stmt, $resultStack);

        //instance increment variable
        $i = 0;

        //While there are results to fetch
        while($stmt->fetch()) {

           //instantiate a new element in the result array
           $result[$i] = array();

           //For every column returned from resultStack
           //get the column name and value and set new multidimensional associative array
           foreach($resultStack as $k=>$v){
              $result[$i][$k] = $v;
           }

           //increment result array
           $i++;

        }


     } else {
        $result = 1;
     }

     //Close the query
     $stmt->close();

     return $result;
  }
```

Code Samples

jQuery XML nav Menu

```
/*
 * xml-nav
 *
 * A XML based menu built using jQuery
 *
 * @author Joshua Moore Frankel <joshmfrankel@gmail.com>
 * @copyright Copyright (c) 2011, Joshua Frankel
 * @license Dual licensed under the MIT and GPL licenses.
 * @version 0.5
 *
 */
(function($){

    /*
     * Main Function
     * @acess public
     * @param object literal - Allows for user customization of default options
     */
    $.fn.jTravel = function(options) {

        //Extend the default options with those provided
        //First object empty to prevent overriding of "config" object
        //Always returns new object
        var $this = $(this);
        var config = $.extend({}, $.fn.jTravel.defaults, options);
        var callerID = $this.attr('id');
        var markup = '<ul>';

        //Retain jQuery Chainability
        return this.each(function() {

            //Call the loadXML function to parse the XML as text
            if(config.url.length) {
                var xml = xmlLoader();
            } else {
                alert("Not valid xml file location.");
            }

            //Find each menu that is a child of the root element
            //Call the recursive function to begin building the menu structure
            $(xml).children(config.node).each(recurseXML);

            //Add closing unordered list to the markup string
            markup += '</ul>';


            //Append the completed markup string to the specified div id
            $("#"+callerID).append(markup);


            /*
             * LoadXML Function
             *
             * Loads XML using jQuery ajax
             * Set to Synchronous mode to save xml as a variable
             *
             * @acess private
```

```
         * @param object config     The config object
         * @return string xml
         */
        function xmlLoader() {
           var xml = null;
           $.ajax({
              type:"GET",
              url: config.url,
              dataType: "text",
              async: false,
              success: function(data){
                 xml = data;
              }
           });
           return xml;
        }

         /*
         * recurseXML Function
         *
         * Traverse XML tree recursively and build a markup string
         * to be appended to a div id
         *
         * @acess private
         * @param object    -   The calling object, should be an xml node
         */
        function recurseXML(){

           //Initialize node variables
           var url = $(this).attr("url");
           var text = $(this).attr("text");
           var title = $(this).attr("title");
           var hasChildren = $(this).children(config.node).length > 0;

           //If the current node has a url attribute then it is a link
           //Otherwise it is a category and does not require an anchor tag
           if(url){
              markup += "<li><a href='"+url+"' title='"+title+"'>"+text+"</a>";
           } else {
              markup += "<li><a href='#' title='"+title+"' >"+text+"</a>";
           }

           //If current node has children then use recursion to build an unordered list
           if(hasChildren){

              //add image here
              markup += ' <ul>';

              //Start recursion to move through child nodes of current node
              $(this).children(config.node).each(recurseXML);

              markup += '</ul>';
           }

           markup += '</li>';
        }
     });
  };
```

Code Samples

```
    /*
     * Properties
     *
     * @access public
     * @var object
     */
    $.fn.jTravel.defaults = {
        url: "",
        effect: "slide",
        target: "_self",
        node: "menu"
    };

})(jQuery);
```

Code Samples

PHP User Class

```
/**
 * Private Properties
 */
private $_username      = "";
private $_password      = "";
private $_email         = "";
private $_activationKey  = "";
private $_sessionKey     = "";

/**
 * Activates a User account if the query string matches the activation string
 *
 */
public function activateUser() {

   //Instantiate a new connection
   $Connection = new Connection();

   //Variables to Pass
   $sql = "SELECT userID FROM USERS WHERE activationKey = ? AND active = 0 LIMIT 1";
   $params = array('s', $this->_activationKey);

   //Call the Query method
   $result = $Connection->query($sql, $params, 1);

   if($result) {

      //Variables to Pass
      $sql = "UPDATE Users SET active = 1, activationKey = NULL WHERE userID = ?";
      $params = array('i', $result[0]['userID']);

      //Call the Query method
      $result = $Connection->query($sql, $params, 0);

   } else {
      return false;
   }
}


/**
 * User login Method
 *
 * Check the hashed password vs the saved database password.  Immediately remove all password
 * variables after usage.
 */
public function login() {

   $success = FALSE;

   //Instantiate a new connection
   $Connection = new Connection();

   //bCrypt external hashing library
   $Hasher = new PasswordHash(8, FALSE);

   //get the matching password for the user authentication
```

Code Samples

```
    $loginPassword = $this->_getMatchingPassword();

    //check the input password vs the database password
    $hasherCheck = $Hasher->CheckPassword($this->_password, $loginPassword[0]['password']);

    //Unset password variables
    unset($this->_password);
    unset($loginPassword);

    //if there is a match
    if($hasherCheck) {

        $sessionKey = $Hasher->HashPassword(microtime());

        //Variables to Pass
        $sql = "UPDATE Users SET loginStatus = 1, sessionKey = ? WHERE username = ?";
        $params = array('ss', $sessionKey, $this->_username);

        //Call the Query method
        $result = $Connection->query($sql, $params, 0);

        //Username matches the hashed password
        $success = array($this->_username, $sessionKey);

    } else {

        //Password mismatch
        $success = FALSE;
    }

    return $success;
}


/**
 * Register Method
 */
public function register() {

    //bCrypt external hashing library
    $Hasher = new PasswordHash(8, FALSE);

    //Call method to get matching username
    $result = $this->_getMatchingUsername();

    if($result) {
        return  "User already exists";
    } else {

        //Instantiate a new connection
        $Connection = new Connection();

        //generate random 32 character hash
        $hash = md5(rand(0,1000));
        $access = 2;
        $active = 0;

        //Hash Password
        $this->_password = $Hasher->HashPassword($this->_password);
```

```
            //Variables to Pass
            $sql = "INSERT INTO Users (accessID, username, password, email, active, activationKey,
sessionKey)
                VALUES(?, ?, ?, ?, ?, ?, ?)";

            $params = array('isssiss', $access, $this->_username, $this->_password, $this->_email,
$active, $hash, NULL);

            //Call the Query method
            $result = $Connection->query($sql, $params, 0);


        }

    }

    public function logout() {

        //Instantiate a new connection
        $Connection = new Connection();

        //Call the Query method
        $result = $this->getLoginStatus();

        if($result) {

            //Variables to Pass
            $sql = "UPDATE Users SET loginStatus = 0, sessionKey = NULL WHERE username = ? AND
sessionKey = ?";
            $params = array('ss', $this->_username, $this->_sessionKey);

            //Call the Query method
            $Connection->query($sql, $params, 0);
        }

        return $result;
    }

/**
 * Getters and Setters
 */


    /**
     * Private function to return the user password
     *
     * @return string The current username password
     */
    private function _getMatchingPassword() {

        //Instantiate a new connection
        $Connection = new Connection();

        //Variables to Pass
        $sql = "SELECT password FROM users WHERE username = ? AND active = 1 AND loginStatus =
0 LIMIT 1";
        $params = array('s', $this->_username);
```

```
        //Call the Query method
        $result = $Connection->query($sql, $params, 1);

        //No matching username found or not active account
        if(!$result) {

            return FALSE;
        }

        return $result;
    }


    /**
     * Private method to determine if a username exists
     */
    private function _getMatchingUsername() {

        //Instantiate a new connection
        $Connection = new Connection();

        //Variables to Pass
        $sql = "SELECT username FROM USERS WHERE username = ? LIMIT 1";
        $params = array('s', $this->_username);

        //Call the Query method
        $result = $Connection->query($sql, $params, 1);

        return $result;
    }



    /**
     * Public method to return the login status
     */
    public function getLoginStatus() {

        //Instantiate a new connection
        $Connection = new Connection();

        //Variables to Pass
        $sql = "SELECT userID FROM users WHERE username = ? AND sessionKey = ? AND active = 1
AND loginStatus = 1 LIMIT 1";
        $params = array('ss', $this->_username, $this->_sessionKey);

        //Call the Query method
        $result = $Connection->query($sql, $params, 1);

        return $result;
    }


    public function setUsername($value) {
        $this->_username = $value;
    }

    public function setPassword($value) {
        $this->_password = $value;
```

```
}

public function setEmail($value) {
   $this->_email = $value;
}

public function setActivationKey($value) {
   $this->_activationKey = $value;
}

public function setSessionKey($value) {
   $this->_sessionKey = $value;
}
```

Code Samples

Apache Ant Build file

```
<!--
   Document   : build.xml
   Created on : April 20, 2011, 2:10 PM
   Author     : JFrankel
   Description: An ant build file
-->

<project name="origin" default="init">

   <!-- Import Properties -->
   <property file="build.properties" />
   <property file="${dir.rev}" />


   <!-- Main Method -->
   <target name="init" depends="revision">
      <echo>Starting Ant Build...</echo>
      <antcall target="copy"/>
      <antcall target="minify.html"/>
      <antcall target="optimize.png"/>
      <antcall target="optimize.jpg"/>
      <antcall target="compress.css"/>
      <antcall target="compress.js"/>
      <echo>Done!</echo>
   </target>


   <target name="copy" depends="clean">
      <echo>Starting Build Publish...</echo>
      <mkdir dir="../${dir.publish}"/>
      <copy todir="../${dir.publish}">
         <fileset dir="../">
            <exclude name="**/*.js"/>
            <exclude name="**/*.css"/>
            <exclude name="**/css/libs/**"/>
            <exclude name="**/${dir.build}/**"/>
            <exclude name="**/${dir.publish}/**"/>
         </fileset>
      </copy>
   </target>


   <!-- Clean up previous build directory -->
   <target name="clean">
      <echo>Cleaning previous build R.${build.rev}...</echo>
      <delete dir="../${dir.publish}"/>
   </target>


   <!-- Revision Control -->
   <target name="revision">
      <echo>Creating a new revision...</echo>
      <propertyfile file="${dir.rev}">
         <entry key="build.rev" type="int" default="0000" operation="+" pattern="0000"/>
         <entry key="build.date" type="date" value="now" pattern="dd.MM.yyyy HH:mm:ss" />
      </propertyfile>
      <property file="${dir.rev}"/>
```

```
    </target>


    <!-- Project Upload -->
    <target name="upload">
        <ftp    server   ="${ftp.servername}"
                userid   ="${ftp.userid}"
                password ="${ftp.passwd}"
                port     ="${ftp.port}"
                remotedir="${ftp.dir}"
                passive  ="${ftp.passive}"
                binary   ="${ftp.binary}">

            <!-- fileset node dir defines root directory we want to collect -->
            <fileset dir=".">
                <include name="${dir.publish}/*"/>
            </fileset>

        </ftp>
    </target>


     <!-- Compress HTML -->
    <target name="minify.html">
        <echo>Minify HTML, removing spaces, compress inline js and CSS, removing unnecessary
quotes...</echo>
        <apply executable="java" parallel="false" force="true" dest="../${dir.publish}/">
            <fileset dir="../${dir.publish}/" includes="**/*.html"/>
            <arg value="-jar"/>
            <arg path="tools/htmlcompressor-0.9.3.jar"/>
            <arg line="--type html" />
            <arg line="--remove-intertag-spaces"/>
            <arg line="--compress-js"/>
            <arg line="--remove-quotes"/>
            <arg line="--compress-css"/>
            <srcfile/>
            <arg value="-o"/>
            <mapper type="identity"/>
            <targetfile/>
        </apply>
    </target>


    <!-- Main Compression Method -->
    <target name="compress">
        <java jar="${tools.yui}" fork="true">
            <arg value="${file}" />
            <arg value="-o"/>
            <arg value="${file}" />
        </java>
    </target>


    <!-- JS Compress -->
    <target name="compress.js" depends="concat.js">
        <echo>Compressing JS...</echo>
        <!-- call another method -->
        <antcall target="compress">
            <param name="file" value="../${dir.publish}/${dir.js}/plugin-${build.rev}.js" />
```

```
    </antcall>
</target>


<!-- CSS Compress -->
<target name="compress.css" depends="concat.css">
    <echo>Compressing CSS...</echo>
    <!-- call another method -->
    <antcall target="compress">
        <param name="file" value="../${dir.publish}/${dir.css}/stylesheet-${build.rev}.css" />
    </antcall>
</target>


<!-- Concatentate JS -->
<target name="concat.js">
    <echo>Concatentating JS...</echo>
    <concat destfile="../${dir.publish}/${dir.js}/plugin-${build.rev}.js">
        <fileset dir="../${dir.js}/">
            <exclude name="**/${file.jquery}"/>
            <!--
                        Concatentate all js files with the prefix .min in the
                        JS directory
             -->
            <include name="**/*.js" />
        </fileset>
    </concat>
</target>


<!-- Concatentate CSS -->
<target name="concat.css">
    <echo>Concatentating CSS...</echo>
    <concat destfile="../${dir.publish}/${dir.css}/stylesheet-${build.rev}.css">
        <fileset dir="../${dir.css}/">
            <exclude name="fonts/*.css"/>
            <include name="**/*.css" />
        </fileset>
    </concat>
</target>



<!-- Optimize PNG files -->
<target name="optimize.png">
    <echo>Optimizing PNG files...</echo>
    <apply executable="${tools.png}" osfamily="windows">
        <arg value="-o7"/>
        <fileset dir="../${dir.publish}/${dir.images}/">

            <!--
                This line will include all png files that are in
                child directories of the fileset dir attribute
            -->
            <include name="**/*.png"/>
        </fileset>
    </apply>
</target>
```

Code Samples

```
    <!-- Optimze JPG files (Make sure ImageMagick is installed) -->
    <target name="optimize.jpg">
        <echo>Optimize JPG files...</echo>
        <apply executable="${tools.jpg.exec}" osfamily="windows">

            <!-- Strip Meta Data (Could easily save 6kb per image)-->
            <arg value="-strip"/>
            <arg value="-quality"/>
            <arg value="${tools.jpg.qual}"/>

            <fileset dir="../${dir.publish}/${dir.images}/">

                <!--
                        This line will include all jpg files that are in
                        child directories of the fileset dir attribute
                -->
                <include name="**/*.jpg"/>
            </fileset>
        </apply>
    </target>
</project>
```

Code Samples

Coldfusion and MS SQL Query

```
<cffunction name="getModelParameters" access="private" output="false" returntype="query">
	<cfargument name="argModel" type="String" required="true" displayname="argModel">
	<cfargument name="argAccount" type="String" required="true" displayname="argAccount">
	<cfargument name="argBank" type="String" required="true" displayname="argBank">

	<cfquery name="queryModelParameters" datasource="#DATASOURCE#">
		SELECT DISTINCT
			m.Name,
			m.Name + ' : ' +
			mp.parameter1 + ' : ' +
			(CASE m.Name
			WHEN 'SOV' THEN
				(CASE mp.Mode
				WHEN '1' THEN 'Trend'
				WHEN '0' THEN 'Non-Trend'
				ELSE '_' END)
			ELSE mp.parameter2 + ' : ' +
				(CASE mp.direction
				WHEN '0' THEN 'UD'
				WHEN '1' THEN 'U'
				WHEN '-1' THEN 'D'END)
			END)
			as Model_Full_Name,
			m.Model_ID,
			mp.Parameter1,
			CASE ISNULL(mp.Mode, '')
				WHEN '' THEN '-'
				ELSE mp.Mode
			END as Mode,
			CASE ISNULL(mp.Parameter2, '')
				WHEN '' THEN 0
				ELSE CONVERT(float, mp.Parameter2)
			END as Parameter2,
			mp.parameter3,
			mp.parameter4,
			mp.parameter5,
			mp.Model_Parameter_ID,
			c.Currency,
			mp.direction as Bias,
			rfx.Quantity,
			rfx.Round_Up_Down,
			rfx.Round_To,
			rfx.Weighted_Percent,
			rfx.Strength,
			rfx.Allocation,
			rfx.R_FAMP_ID,
			a.Name AS Account,
			a.Account_ID,
			co.Name AS Bank,
			co.Counterparty_ID,
			c2.Currency AS Major_Currency,
			cp.Currency_Pair,
			cp.Short_Currency_Pair
		FROM
			dbo.Model_Parameters AS mp INNER JOIN
			dbo.Models AS m ON mp.Model_ID = m.Model_ID INNER JOIN
			dbo.R_FX_Account_Model_Parameter AS rfx ON mp.Model_Parameter_ID =
```

```
                rfx.Model_Parameter_ID INNER JOIN
                dbo.Currencies AS c ON rfx.Traded_Currency_ID = c.Currency_ID INNER JOIN
                dbo.Accounts AS a ON rfx.Account_ID = a.Account_ID INNER JOIN
                dbo.Counterparties as co ON rfx.Counterparty_ID = co.Counterparty_ID
                INNER JOIN
                dbo.Currency_Pairs AS cp ON rfx.Currency_Pair_ID = cp.Currency_Pair_ID
                INNER JOIN
                dbo.Currencies AS c2 ON cp.Major_Currency_ID = c2.Currency_ID
        WHERE
                (rfx.Active_Flag = '1')
                <cfif arguments.argModel NEQ 'All'>AND m.Name = <cfqueryparam
                        cfsqltype="cf_sql_varchar" value="#argModel#"></cfif>
                <cfif arguments.argBank NEQ 'All'>AND co.Counterparty_ID = <cfqueryparam
                        cfsqltype="cf_sql_integer" value="#argBank#"></cfif>
                <cfif arguments.argAccount NEQ 'All'>AND a.Account_ID = <cfqueryparam
                        cfsqltype="cf_sql_integer" value="#argAccount#"></cfif>
        ORDER BY
                m.Name,
                c.Currency,
                a.Name,
                co.Name
    </cfquery>

    <cfreturn queryModelParameters>
</cffunction>
```