eSolutions

ICT Volume 3 : Application Standards

# ICT 3.2.2-2016 Web Application Development Standards

**Abstract**

This document defines standards applicable to any web application developed at Deakin University.

## Document Control

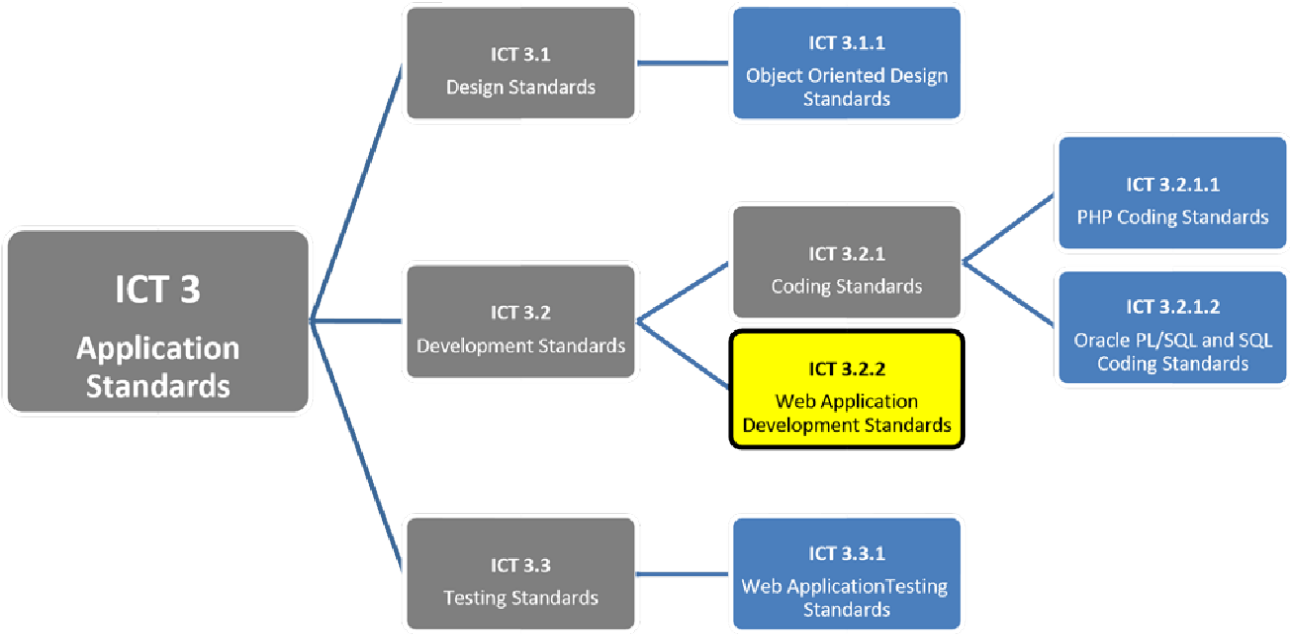
| | |
|---|---|
| **Document Title** | ICT 3.2.2-2016 Web Application Development Standards |
| **Version** | 2016 |
| **Controlled Copy (Electronic Reference)** | |

## Document History

| Ver. | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 2.1 | Michael Heley | Standards Compliance Review | 11-09-2014 |
| 3.0 | Michael Heley | 2016 Review | 15-07-2016 |
| 3.1 | Michael Heley | Accessibility Update | 14-10-2016 |
| 3.2 | Hayden McFadyen | Accessibility Update | 07-11-2018 |

## Table of Contents

# ICT Volume 3 : Application Standards


ICT 3
Application Standards
ICT 3.1
Design Standards
ICT 3.1.1
Object Oriented Design Standards
ICT 3.2
Development Standards
ICT 3.2.1
Coding Standards
ICT 3.2.1.1
PHP Coding Standards
ICT 3.2.1.2
Oracle PL/SQL and SQL Coding Standards
ICT 3.2.2
Web Application Development Standards
ICT 3.3
Testing Standards
ICT 3.3.1
Web ApplicationTesting Standards

## Standards Brief

This document serves to outline standards that shall apply within Deakin University.

## Standard Document Access

All Deakin University staff and authorised/approved contracted personnel are provided access to this document.

## Policy

These standards must be used in conjunction with all other referenced standards, and when considered in isolation from the referenced standards may not constitute adequate conformance.

## Conflict of Information or Clarification

Whenever a conflict of information occurs or clarification of instruction is required all queries shall be made to Deakin University eSolutions (DeS).

# 1 Design Methodology

## 1.1 The Object Oriented design standards shall apply (ICT 3.1.1)

All server-side coding within web applications shall conform to the Object Oriented Design Standards as documented in ICT 3.1.1 where practical. Any standards set out in this document are supplementary to those standards.

## 1.2 The PHP 5 language shall be used

Web development will be implemented using compliant PHP 5 language and shall be aligned with server supported infrastructure.

## 1.3 An MVC architecture approach should be used

A Model View Controller approach to application architecture should be used where practical.

# 2 Content

The following standards relate to web applications, classes or components which return or generate content which will be sent to the client.

## 2.1 Content shall conform to digital and accessibility standards

Both content and the platform must conform to Deakin's Digital standards and Accessibility policy and guidelines.

## 2.2 Content shall conform to all relevant legislative requirements

Content shall conform to all relevant legislative requirements including laws surrounding copyright, censorship and all other applicable legislation.

## 2.3 HTML standards shall apply

HTML content sent to the client shall be valid (HTML 4.01, XHTML 1.0 and 1.1) as set out by the W3C (http://validator.w3.org/ or internally by web-validator.its.deakin.edu.au)

## 2.4 Client script should be compressed

Client-side scripting (JavaScript) should be compressed in order to reduce size and improve performance.

## 2.5 HTML forms

### 2.5.1 Form naming conventions

The following may be prefixed to the value of the “name” attribute within a form element:

- "txt_" for text elements
- "chk_" for checkbox elements
- "rdb_" for radio button elements
- "sel_" for select elements
- "btn_" for button elements
- "hid_" for hidden elements

### 2.5.2 Form element length

The length of a form element should be representative of any related database column length. For example, a database column of VARCHAR(50) should be represented by a form element with a maximum data entry length of 50 characters.

### 2.5.3 Form File Uploads

Ensure forms do not allow files of any size to be uploaded. It is recommended to enforce a max file size limit of no more than 20Mb unless there is a requirement to handle more. Using PHP’s MAX_FILE_SIZE is helpful for user experience but is not intended for security.

### 2.5.4 HTTP verbs shall be used appropriately

The HTTP request method employed shall conform to RESTful principals and as a general rule should map to the CREATE, READ, UPDATE, DELETE (CRUD) operations associated with database technologies as follows:

| **HTTP** | **CRUD** |
|---|---|
| POST | Create, Update, Delete |
| GET | Read |
| PUT | Create, Overwrite/Replace |
| DELETE | Delete |

## 2.6 Naming

### 2.6.1 Class names shall apply to Zend naming conventions

Class names shall comply with the standards set out by Zend http://framework.zend.com/manual/2.3/en/ref/coding.standard.html#namingconventions. An exception is granted for the “DU” namespace, but not child namespaces of DU.

### 2.6.2 Class names shall be defined by the file system path which they reside in

A class will contain all namespaces / directories that are derived from the class root. These shall be formatted according to ICT standards.

### 2.6.3 File system

#### 2.6.3.1 Namespaces shall be represented by directories

Namespaces will be implemented through the creation of directories on the file system. A directory represents a namespace in any file structure that is deemed or constitutes a “library”.

#### 2.6.3.2 Each file shall contain a maximum of one class

#### 2.6.3.3 Class filenames shall replicate the class name, minus packages/namespace

A class filename shall be the class name, without leading namespace and/or package name, followed by the “.php” extension. For example, the class “MyPackage_MyClass” will reside within a file named “MyClass.php”

## 2.7 Email

### 2.7.1 Information Security Standards shall apply

The “Information Security - Standard for mass emails and emails generated by information system” shall apply for all system generated emails.

### 2.7.2 Application-generated email of a system-support nature shall be sent to an email account dedicated to the receipt of such correspondence

Where an application generates email for the purposes of systems support (status notification, error notification, etc), this email will be only sent to an email account that is dedicated to receiving that type of email. This shall also apply for vendor support correspondence. It is not acceptable for these emails to be sent to individuals’ email accounts or to mailing lists that individuals are subscribed to.

# 3 Documentation

## 3.1 Class diagram

The class diagram shall conform to the requirements of a class diagram stated in ICT 3.1.1 with the following additions.

### 3.1.1 Type hinting for Abstract Data Types shall be depicted

Where abstract data types are passed as parameters to function, a type hint shall be depicted on the class diagram.

## 3.2 Code shall be documented using phpDocumentor conventions

phpDocumentor style comment tags shall be used to document PHP code. Documentation shall comply with the standards and conventions set out by http://manual.phpdoc.org/HTMLframesConverter/default/

## 3.3 Code shall not be commented-out

“Commented-out” code shall not be stored within the application source and must be removed.

# 4 Error handling

## 4.1 PHP Exception architecture

### 4.1.1 The PHP Exception architecture shall be used to handle errors

Errors shall be represented by Exceptions and caught by the program code. Other forms of error handling are not acceptable.

### 4.1.2 Custom exception objects shall inherit from a base exception class

Custom exceptions declared and raised by the application shall inherit from a base exception class

### 4.1.3 Functions shall not return customised errors.

Functions shall not return a customised error code, string, object or other. Functions may return a boolean value indicating whether the operation was successful.

### 4.1.4 Try/catch blocks

When calling methods which could trigger an exception, and there is a logical path that code could follow in the event of an exception (other than just propagating it to the user), a try/catch block should be used.

The ‘try’ section shall contain only:

1. The statements that could fail, and
2. Subsequent code that is dependent on the statements that could fail (code that should not be executed in the event of an exception)

The ‘catch’ section shall not re-throw the exception unless it performs other operations as well; otherwise the try/catch block is redundant.

### 4.1.5 PHP warnings

Functions that can trigger warnings may be suppressed with an '@' symbol only if valuable troubleshooting information is not lost by doing so.

# 5 APIs

## 5.1 An object oriented API architecture shall be used

*ICT 3.1.1 Object Oriented Design Standards* shall apply for all APIs.

## 5.2 Acceptable API implementations

Acceptable API implementations are limited to:

1. Standards compliant shared class hierarchy
2. SOAP standards based web service
3. HTTP (RESTful) interface

Excluded API implementations are, but not limited to:

1. Direct database calls
2. Shared files
3. SMTP
4. Telnet
5. Any non-PHP platform-specific implementation

## 5.3 Web Services

### 5.3.1 Web services shall comply with SOAP standards

All web services shall comply with SOAP standards http://www.w3.org/TR/soap/

### 5.3.2 Web services shall employ and comply with WSDL standards

# 6 Database

## 6.1 A database abstraction class shall be used

Any database interaction shall be via a database abstraction class.

## 6.2 A database abstraction shall not contain business logic

Any database abstraction shall be strictly for the purpose of interacting with the database in a generic manner. Business logic shall not be included in any part of a database abstraction.

A database abstraction should be generic to the extent that it could be replaced with minimal effort with another database abstraction of the same interface.

# 7 Testing

## 7.1 The ITSD Web Application Testing Standards shall apply

*ICT 3.3.1 Web Application Testing Standards* shall apply.

# 8 Maintenance

## 8.1 Design documentation shall be kept up to date

Design documentation shall be updated whenever modifications to a system are undertaken which impact the design documentation.

## 8.2 Modifications and / or additions to any application shall not adversely impact the application upgrade path

Any modifications, additions, customisations or configurations applied to any application (either in-house or third party supplied) shall not bear an adverse impact on the application's upgrade path. That is, the application shall continue to be upgradeable and patched in accordance with the application's standard upgrade procedures.

# 9 Security

## 9.1 Session time-out

Inactive sessions shall shut down after period of inactivity that is defined in accordance with the data classification of the application.

## 9.2 Data input shall be validated to ensure data is correct and appropriate.

## 9.3 Internal processing shall be validated to detect and prevent any corruption of information through processing errors

## 9.4 Data output shall be validated to ensure data is correct and appropriate.

### 9.5 Program units shall be technically code reviewed.

### 9.6 A fully compliant code review is required prior to release.

The identification of items requiring attention in a code review shall automatically necessitate additional review until a fully compliant code review is achieved.

### 9.7 Credentials shall be encrypted

Credentials used by the application to log onto other systems (eg: databases) shall be stored in encrypted form.

### 9.8 Application shall log onto systems and databases using suitably restrictive account

Applications shall log on to other systems and databases using an account that is granted with the least amount of privileges required.

## 10 Source Control

### 10.1 Application code shall be revision controlled

In general, web applications should be revision controlled using “Subversion”.

## 11 Monitoring

### 11.1 Production systems shall be monitored

Any application residing within a production environment shall be monitored via a standardised enterprise monitoring service. Any detected change in application state shall be reported and/or escalated in accordance with the standard operating procedure set out by the enterprise monitoring service.

# 12
# Appendix A

## Definitions

| Term/Abbreviation. | Definition |
|---|---|
| API | **Application Programming Interface:** A set of calling conventions that enable one application to utilize the services of another application or shared library. |
| PHP | **PHP: Hypertext Preprocessor** The primary server-side scripting language used for building web applications at Deakin. |
| Class | The basic building block of software in the *object-oriented programming* paradigm. |
| Subversion | Apache Subversion (often abbreviated SVN, after the command name svn) is a software versioning and a revision control system. |
| Web Service | A software system designed to support interoperable machine-to-machine interaction over a network |
| Web application | An application that is accessed via web browser over a network such as the Internet or an intranet |