2019-01-15
Бэлтгэсэн: Б.Сод-Од, Б.Батзолбоо, Г.Цэнд-Аюуш

Програм хангамж хөгжүүлэлтийн кодын стандарт баримт

БАРИМТЫН ТУУХ

Хувилбар	Засварласан	Огноо	Тайлбар

АГУУЛГА

1.1 Зорилго

Энэхүү баримт бичиг нь програм хангамжийн төслийн хүрээнд код бичихэд шаардагдах кодчиллын стандартыг тодорхойлно.

1.2 Хамрах хүрээ

Компьютерийн ухааны болон бусад салбарын оюутнууд баримтлана.

1.3 Хэрэглээ

Төслийн хөгжүүлэлтийн явцад програм хангамжийн инженер нь бусад програм хангамжийн инженерийн бичсэн кодыг арчлах, үргэлжлүүлэн хөгжүүлдэг. Кодын стандарт баримтлахгүй бол дээрх үйл ажиллагааг гүйцэтгэхэд их цаг зарцуулж, бие биеэсээ хамааралтай болдог. Тиймээс энэхүү кодын стандартын тусламжтай нэг загвараар хөгжүүлж, эцсийн бүтээгдэхүүнийг цаашид хөгжүүлэхэд хялбар байдаг. Мөн илүү инженерийн түвшинд кодчилол хийх боломжийг хөгжүүлэгчдэд олгодог.

1.4 Ерөнхий

1.4.1 Дүрэм – Нэрийн стандарт:

1. Төсөлд холбоотой ямар нэгэн класс, гишүүн өгөгдөл зарлах тохиолдолд дараах нэрийн стандартыг баримтална.

Стандарт	Тайлбар	Жишээ
Pascal	Классын нэр нь хэд хэдэн нэр үгээс бүтсэн нэр үг байна.	**S**alary**C**alculator
Camel	Функц болон хувьсагч, параметрийн нэрийг бичихэд ашиглана. Эхний үсэг жижгээр үлдсэн үсгүүд томоор бичигдэнэ. (Функц болон Method – ууд үйл үг байна)	getEmployeeSalar y($employeeId) – ажилчдын цалинг авах гэсэн
UPPERCASE	Тогтмол хувьсагчийн утгыг зарлахад ашиглана.	const INVENTORY_RE QUEST = ‘300001’

1.4.2 Зөвлөмж:

2. Эхлээд кодын уншигдах байдал байна. Дараа нь ажиллагаа, хурд болон гүйцэтгэлийн асуудал дэлгэрэнгүй яригдана.

3. Классын мөрийн тоог 1000 мөрөөс хэтрүүлэхгүй байхыг зорино. Хэрэв мөрийн тоо стандартаас хэтэрч байвал өөр дэд классуудад хуваах байдлаар асуудлыг шийдвэрлэнэ. Мөн нэг класст хамгийн ихдээ 5 ялгаатай бусад классын объектыг үүсгэж болно. Ингэснээр Bottleneck болон God Class үүсэхээс сэргийлнэ.

4. Функц болон аргын мөрийг тоог 50 мөрөөс хэтрүүлэхгүй байхыг зорино. Хэрэв мөрийн тоо стандартаас хэтэрч байвал өөр дэд функц болон аргуудад хуваах замаар асуудлыг шийдвэрлэнэ.

1.5 Кодын хэв загвар

1.5.1 Дүрэм:

1. **Нэг түвшинд** зай авна гэдэг нь 4 ширхэг space дарсантай адил байна. **4 ширхэг space нь нэг Tab даралтаар орлуулагдаж болно.**

```
    public function createInventoryRequest(Request $request)
```

2. Мөрийн тоо 120-н тэмдэгтээс хэтрэх ёсгүй.

3. Функц болон класс зэргийн угалзан хаалт бичихдээ дараах стандартыг баримтална.

   а) Эхлэх болон дуусах угалзан хаалт нь **нэг түвшинд** байршина.

```
if ( $byteBuf == null )
{
    $byteBuf = new ByteBuffer( PurchaseReportController::BUFFER_SIZE );
}
```

b) Төгсгөлийн угалзан хаалт кодын блокийн мөрийн эхлэлтэй нэг түвшинд бичигдэнэ.

```
if ( $byteBuf == null ){
    $byteBuf = new ByteBuffer( PurchaseReportController::BUFFER_SIZE );
}
```

Онцгой тохиолдол: Хэлний түлхүүр үгс хооронд хаалттай бол хаалтууд нэг түвшинд байх албагүй.

```
do {
   //Блок тооцоолол 2 –оос дээш мөр байвал +1 Break авна
   $charNumber++;
   $ch = $clientName->getChar( $charNumber );
   //Блок тооцоолол 2 –оос дээш мөр байвал +1 Break авна
} while ( $ch != '\0' );
```

```
try {
   $this->doSomeThing()
} catch ($exception ex) {
       $this->handleException();
}
```

Дээрх 2 хаалт бичих хэв загвар нь нэг файлд хослуулан ашиглагдаж болно. Хэрэв файлыг болон багцыг засварлах, өөр классаас хуулах бол стандартаа алдахгүй байх ёстойг анхаарна.

1.5.2 Зөвлөмж:

1. Функц болон арга бичиж байгаа бол эхлэлийн дугуй хаалт үргэлж ямар нэгэн зайгүйгээр функц аргын нэрийн ард бичигдэж байх ёстой. **ConfirmRequest()**

2. Операторын 2 талд үргэлж нэг ширхэг Space дагалдана..

Буруу бичиглэл:

```
$newXPos+=$recClient->Width()+$recUpdate->Width()+$margin*$columns;
```

Дээрх байдлаар бичих нь кодын уншигдах байдлыг илүү төвөгтэй болгодог. Тиймээс доорх жишээ код шиг бичиглэл хийхийг зорино.

```
$newXPos += $recClient->Width() + $recUpdate->Width() + $margin * $columns;
```

3. Кодын бичиглэл олон мөрөөр харагдах шаардлага гарвал дараах стандарт бичиглэлээр дараагийн мөр рүү шилжүүлнэ. Дараагийн мөр рүү үргэлжилж орж байгаа гэдэг нь нүдэнд харагдахуйц мөн бичиглэлийн алдаагүй байх ёстой. Мөр үргэлжилж байгаа үед нэг түвшинд бичигдэх шаардлагагүй гэдгийг анхааараарай.

```
 $addressLine1 = $this->getApartmentNumber()
                 + $this->getStreetAddress()
                 + $this->getStreetName();
```

1.6 Тайлбар бичих:

1.6.1 Дүрэм:

1. Бүх класс болон аргууд заавал тайлбартай байх шаардлагатай.
2. Классын тайлбар заавал утгатай тайлбарыг агуулна. Мөн зохиогчийн нэр болон үүсгэсэн огноо, засвар оруулсан бол засварласан огноо багтсан байх шаардлагатай. Мөн классын үүргийг тайлбарлахдаа бүх үсгийг UPPERCASE байдлаар бичнэ.

```php
<?php
/**
 * Created by Edmon.
 * User: M
 * Date: 7/8/2018
 * Time: 12:42 PM
 */
namespace App\Http\Controllers\Backend\Inventory;
use ...
```

```php
/**
 * BACKEND - ШААРДАХТАЙ ХОЛБООТОЙ СЕРВИСҮҮД
 */
class InventoryRequestController
```

3. Функц бичиж байгаа тохиолдолд толгой хэсэгт нэг түвшинд тайлбар бичигдсэн байх шаардлагатай. Функцийн үүргийг тайлбарласан байх шаардлагатай. Мөн дараах стандарт бичиглэлүүд заавал багтсан байх шаардлагатай.

   @**param** – буюу авч байгаа параметрийн төрөл болон үүрэг,

   @**return** – утга буцаадаг функц бол буцаах төрөл болон буцааж буй утга,

   @**exception** – онцгой тохиолдол буюу алдаа боловсруулж байгаа бол tag орсон байх.

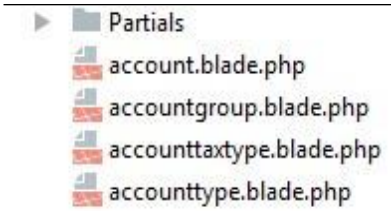
   Функцийн үүрэг, тайлбарыг бичихдээ бүх үсгийг томоор бичнэ.

```php
/**
 * БАРАА МАТЕРИАЛ ШААРДАХ ҮҮСГЭХ
 * @param $request - InvRef - шаардахын дугаар|AccountPkId - дансны дугаар
 * @return \Illuminate\Http\JsonResponse
 * @exception Exception $ex - өгөгдлийн сантай холбоотой алдаа гарвал боловсруулах
 */
public function createInventoryRequest(Request $request)
```

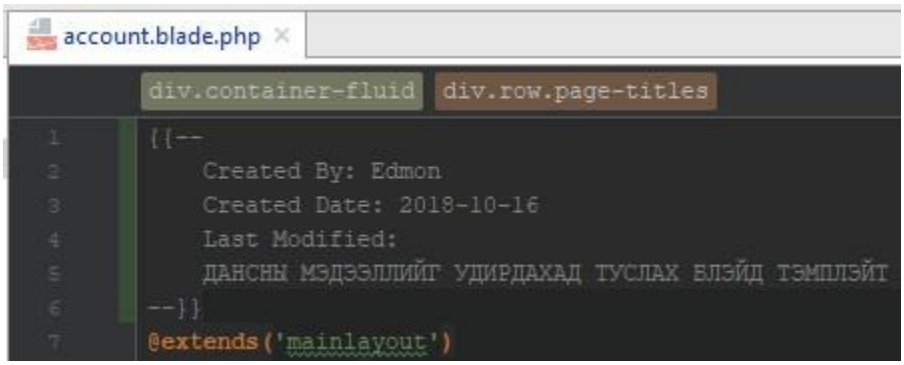
1.7 Файлууд хадгалах:

1.7.1 Дүрэм:

Үүрэг зориулалтаар нь багцлаад нэг хавтсанд хийнэ. Кодчиллоор үүсээгүй файлууд snake (доогуур зураасаар холбох) стандартын дагуу бичигдэнэ. Blade – ээс бусад класс болон өөр нэрс Pascal стандартын дагуу бичигдэнэ.



Нэгж тэмплэйтийн толгой дээр дараах стандарт тайлбар бичигдсэн байх шаардлагатай. Үүсгэсэн хүн, огноо, тухайн тэмплэйтийн үүрэг бичигдсэн байна.



1.7.2 Зөвлөмж 01:

Кодын хэмжээг үргэлж бага байлгахыг зорино. Ахин хянаж шалгах замаар кодын мөрийн тоог хамгийн бага байлгахыг зорино.

1.7.4 Зөвлөмж 02:

1. Хэтэрхий олон буюу 5- түүнээс дээш параметр ашиглахаас зайлсхийх. Олон параметр дамжуулах шаардлагатай бол өгөгдлийн бүтэц эсвэл массив ашиглана.

2. Хэтэрхий урт олон үйлдэлтэй функц бичихээс зайлсхийнэ.

1.8 Гишүүн өгөгдлүүд болон төрөл:

1.8.1 Дүрэм:

1. Тоо болон тэмдэгт өгөгдлийг өөрөөр нь ашиглахаас зайлсхийх. Үүний оронд хувьсагч үүсгэж ашиглана.

Буруу бичигдсэн кодын жишээ:

```
$colWidth = $strWidth + 10 + ( $level * 3 );
...
$cellWidth = $cellStrWidth + 10;
```

Дээрх байдлаар хатуу буюу Hard Coding хийснээр кодод өөрчлөлт ороход илүү хэцүү болгодог. Хэрэв 10 гэсэн утгыг тогтмол зарлаж оноох шаардлагатай. Ингэж бичигдсэн код нь арчилгаа авахдаа илүү хялбар байдаг.

```
const COL_INDENT_WIDTH = 3; // нэгж түвшинд хийгдэх зүүн захаас зайх авалт
const COL_LEFT_MARGIN = 5; // зүүн талын зай авалтын хэмжээ
const COL_RIGHT_MARGIN = 5; // баруун захын зай авалтын хэмжээ
```

```
$COL_MARGINS_WIDTH = PurchaseReportController::COL_LEFT_MARGIN
                     + PurchaseReportController::COL_RIGHT_MARGIN;

$colWidth = $strWidth
            + $COL_MARGINS_WIDTH
            + ( $level * PurchaseReportController::COL_INDENT_WIDTH );
...
$cellWidth = $cellStrWidth + $COL_MARGINS_WIDTH;
```

2. Нэг мөрөнд олон гишүүн өгөгдөл зарлахаас аль болох зайлсхийх.

Стандарт бус жишээ:

```
private $val01 = '', $val02 = '';
```

Стандарт жишээ:

```
private $moveAmount;
private $selectedIndex;
```

1.9 Салаалах оператор болон илэрхийллийн стандарт:

1.9.1 Дүрэм:

1. switch болон case бичихдээ case бүрийн төгсгөлд заавал break түлхүүр үгийг бичиж өгнө. Онцгой тохиолдолд return бичсэн үед break бичихгүй. Ямар ч тохиолдолд default заавал бичигдэнэ.

```
switch ($request)
{
    case "A":
    {
        // А блоктой хамааралтай үйлдэл хийнэ
        break;
    }
    default:
    {
        // Хэрэв дээрх блок кэйс ажиллахгүй бол энэ блок руу орно
    }
}
```

1. Бүх удирдлага, салаалалтын операторууд эхлэх болон дуусах хаалттай байна. (if, else, while, for and do)

   Давталтыг угалзан хаалтгүй бичих нь уншигдахуйц байдлыг бууруулдаг.

```
for($i = 0; $i <= 100; $i++)
    $summary += $i;
```

   1 мөр үйлдэл байсан ч угалзан хаалтыг оруулснаар илүү уншигдахуйц болгодог.

```
for($i = 0; $i <= 100; $i++){
    $summary += $i;
}
```

2. Нэг мөр If болон If.. Else бичихдээ дараах дүрмийг баримтална.

```
$summary = ( $rowNumber == 0 ) ? $this->selectTable()
                               : $this->selectRow( $rowNumber );
```

   Мөн дараах бүтэн бичиглэлтэй байж болно.

```
if ($rowNumber == 0) {
    $this->selectTable();
} else {
    $this->selectRow($rowNumber);
}
```

1.10 HTML, CSS, JS кодчиллын стандарт

Symfony Coding стандарт нь хөгжүүлэгчдийн хамгийн өргөн ашигладаг стандарт юм. Тус кодын стандартыг дагаснаар бүх код эмх цэгцтэй, нэг ижил харагдана. Тус стандарт нь PSR-1, PSR-2 ба PSR-4 стандартууд дээр суурилдаг.
Тус кодын стандартыг ойлгоход дөхөм болох жижиг жишээ авч үзье.

```
/**
 * Энд тухайн классын зорилго, зориулалт товч байна
 */
class Finance
```

Тайлбар 1

```
    /**

     * @var string – Тэмдэгт төрөлтэй хувьсагч
     */
    private $accountNumber;
```

Тайлбар 2

```
    /**

     * @param string $number параметрийн зориулалт илэрхийлж буй утгыг тайлбарлана
     */
    public function     construct($number)
    {

        $this-> accountNumber = $this->transformText($number);
    }
```

Тайлбар 3

```
    /**
     * @return string – Буцах төрөл
     *
     * @deprecated
     */

    public function someDeprecatedMethod()
    {
        @trigger_error(sprintf('The %s() method is deprecated since version 2.8 and will be
removed in 3.0. Use Acme\Baz::someMethod() instead.',     METHOD ),
E_USER_DEPRECATED);
        return Baz::someMethod();

    }
```

Тайлбар 4

```
     /**
     * Эхний параметр утгуудыг хувиргагч байна – функцийн зориулалт
     *
     * @param bool|string $number Энд тус параметрийн тайлбар байна
     * @param array $options Хувиргалтад шаардлагатай сонголтуудын цуглуулга -
     *
     * @return string|null Өөрчлөгдсөн байх утга
     *
     * @throws \RuntimeException Буруу сонголтын үед үүснэ
     */
    private function transformText($number, array $options = array())
    {
```

```
    $defaultOptions = array(
        'some_default' => 'values',
        'another_default' => 'more values',
    );

    foreach ($options as $option) {
        if (!in_array($option, $defaultOptions)) {
            throw new \RuntimeException(sprintf('Unrecognized option "%s"', $option));
        }
    }

    $mergedOptions = array_merge(
        $defaultOptions,
        $options
    );

    if (true === $ number) {
        return null;
    }

    if ('string' === $number) {
        if ('values' === $mergedOptions['some_default']) {
            return substr($number, 0, 5);
        }
        return ucwords($number);
    }
}
```

Тайлбар 5

1.11 Бүтэц

1. Таслал тусгаарлагчийн ард нэг хоосон зай авна.
2. (==, &&, ...) тус үйлдлүүдийн өмнө бас ард нэг хоосон зай авна.
3. Харьцуулах үйлдэл ашиглахдаа дараах зааврыг дагна уу [энд дар](#).
4. Олон мөр болж байгаа жагсаалт массивийг дүрслэхдээ мөр бүрийн ард нэг таслал тавина.
5. Return бичиглэлийн дараа хоосон мөр нэмнэ.
6. Функц нь хоосон утга буцаах бол return null; харин void утга буцаах бол return; гэж бичнэ.
7. Класс бүрийн урт түүний зориулалт, хэрэглээг тайлбарлан бичнэ.

1.12 Нэрлэх

1. **camelCase** загварчлалыг класс болон функц, параметр нэрлэхэд ашиглана. Жишээ нь: $acceptableContentTypes, hasSession());
2. **snake_case** загварчлалыг тохиргооны хувьсагчдад хэрэглэнэ. Жишээ нь: framework.csrf_protection, http_status_code
3. Namespace болон Package -уудыг зарлахдаа **UpperCamelCase** хэлбэр ашиглана. Жишээ нь: ConsoleLogger
4. Файл нэрлэхдээ **camelCase** ашиглана. Жишээ нь: gmScript.js

1.13 HTML бүтэц

1. HTML таг болон атрибутыг нэрлэхдээ том, жижиг үсэг холихгүй.
2. Таг болон атрибутыг нэрлэхдээ дандаа жижиг үсэг ашиглана.
3. Мөн атрибутын нэрийн “дотор бичнэ”. Жишээ нь: Маш муу: <table class=table striped>
   Муу: <table class=striped>
   Зөв: <table class="striped">
4. Зураг буюу img таг ашиглахдаа үргэлж alt атрибут нэмнэ. Жишээ нь: Буруу: <img src="html5.gif">
   Зөв:<img src="html5.gif" alt="HTML5" style="width:128px;">
5. Атрибутад утга оноохдоо хоосон зай авахгүй шууд залгаж бичнэ.
   Буруу: <link rel = "stylesheet" href = "styles.css">
   Зөв: <link rel="stylesheet" href="styles.css">
6. HTML бичихдээ аль болох шаардлагагүй хоосон зай болон мөр авахгүй байна.
7. Таг нээсэн бол заавал хаана.
8. Шаардлагатай үед Viewport ашиглах хэрэгтэй.

1.14 CSS бүтэц

1. /* CSS стиль бичих үед эхэнд нь тухайн стилийн зориулалтыг тайлбарлана. */
2. #login {} - # чагт нь html элементийн ID – г тодорхойлно. HTML -д нэг элемент нэг давхцахгүй ID -тай байхыг зөвлөдөг.
3. .video {} - **.** цэг нь html элементийн КЛАСС – г тодорхойлно. HTML -д олон элемент нэг КЛАСС -тай байхыг зөвшөөрдөг.
4. Мөр бүрийн ард ; цэгтэй таслал байнга бичнэ. Сүүлийн мөрийн байсан ч цэгтэй таслал бичнэ.
5. CSS -ийг зарлахдаа дундуур зураасаар дэд үгсийг холбодог. Жишээ нь:

```
.foo-help {}
#bar-note {}
```

6. Нэг дор олон төрөл зүйлд стиль бичих гэж байгаа бол дараах хэлбэртэй буюу таслал дараагийн мөр гэх зэргээр бичнэ. Жишээ нь:

```
h1,
h2,
h3 {
        font-weight: normal;
        line-height: 1.2;
}
```

7. Мөн CSS 3 -аас эхлэн олон мөр болдог кодыг цөөхөн мөрт багтаан хийх боломжтой болсон.

Өмнө
border-top-style: none;
font-family: palatino, georgia, serif;
font-size: 100%; line-
height: 1.6; padding-
bottom: 2em; padding-left:
1em; padding-right: 1em;
padding-top: 0;
Дараа
border-top: 0;
font: 100%/1.6 palatino, georgia, serif;
padding: 0 1em 2em;

8. CSS Media стиль нь дараах байдалтай бичигдэнэ.

```
@media screen, projection {
        html {
            background: #fff;
            color: #444;
        }
}
```

1.15 Хавсралт JS.

```
/**
 * 2D - Чарт Line, Column зэргийг зурахад хэрэглэнэ
 * @param container - чартыг агуулах div нэрийг өгнө. жнь: 'container1'
 * @param chartTexts - array байна Title зурна. жнь: var chartInfo = ['Бараа материалын орлого', 'Орлого нийт дүнгээр', 'Орлого,'];
 * @param chartType - чартын төрөл байна Line, Column
 * @param unit - yaxis тэнхлэг дээр гаргах нэгж тэмдэгт, MNT г.мэт
 * @param step - хэдэн алхмын зайтай Label харуулахыг шийднэ
 * */
function drawCustom2DChart(container, chartTexts, chartType, unit, step = 3, stacked = false) {
        var chart = Highcharts.chart(container, {
        chart: {
            //Чартын Default төрлийг тодорхойлох шаардлагатай
            type: chartType
        },
        title: {
            text: chartTexts[0]
        },
        subtitle: {
            text: chartTexts[1]
        },
        xAxis: {
            crosshair: true,
            labels: {
                rotation: 45,
```

```
                step: step,
                style: {
                    fontSize: '9px',
                    fontFamily: 'Arial,sans-serif'
                }
            }
        },
        yAxis: {
            min: 0,
            title: {
                text: chartTexts[2] + " "+unit
            }
        },
        tooltip: {
            headerFormat: '<span style="font-
size:10px">{point.key}</span><table>',
            pointFormat: '<tr><td
style="color:{series.color};padding:0">{series.name}: </td>' +
            '<td style="padding:0"><b>{point.y:.1f}'+unit+'</b></td></tr>',
            footerFormat: '</table>',
            shared: true,
            useHTML: true
        },
        plotOptions: {
            column: {
                pointPadding: 0.2,
                borderWidth: 0,
                stacking: stacked
            }
        }
    });
    return chart;
}

/**
 * 2D - чартан дээр SERIES нэмж өгөх арга
 * @param chart - 2D төрөлтэй чарт
 * @param seriesType - чартын төрөл байна Line, Column
 * @param seriesName - чартын нэгж series нэр
 *
 * @param datax - Лабэлуудийн утга: var datax = ['A','B','C'];
 * @param datay - утгууд : var datay = [49.9,71.5,106.4];
 * @param seriesColor - series өнгө
 * @param style - dashed тасархай - - - гэх мэт
 * */
function addDynamic2DSeries(chart, seriesType = 'column', seriesName, datay,
datax, seriesColor = '#90ed7d', style = '', radius = 3){
    chart.addSeries({
        name: seriesName,
        data: datay,
        color: seriesColor,
        type: seriesType,
        dashStyle: style,
        marker: {
            symbol: "circle",
            radius: radius
        }
    }, false);
    chart.xAxis[0].setCategories(datax);
}
```