**ЎЗБЕКИСТОН РЕСПУБЛИКАСИ АЛОҚА, АХБОРОТЛАШТИРИШ ВА ТЕЛЕКОММУНИКАЦИЯ ТЕХНОЛОГИЯЛАРИ ДАВЛАТ ҚЎМИТАСИ**

**ТОШКЕНТ АХБОРОТТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ ФАРҒОНА ФИЛИАЛИ**

**«АХБОРОТ ТЕХНОЛОГИЯЛАРИ» КАФЕДРАСИ**

ҲИМОЯГА»

Кафедра мудири

_______________

« ____ »__________2014 й

# "Developing of offical Web Sie for Academic Lyceum №3 under Fergana State University"

**МАВЗУСИДА**

**МАЛАКАВИЙ БИТИРУВ ИШИ**

БИТИРУВЧИ: Б.Имомалиев

611-10 гуруҳ талабаси

**Фарғона – 2014 йил**

CONTENTS

## ANNOTATION

This graduate work performed in accordance with the Presidential Decree of May 30, 2002 №UP- 3080 “On further development of computerization and introduction of information and communication technologies” and the cabinet of Ministers of the Republic of Uzbekistan dated June 6, 2002 № 200 “On Measures for further development of computerization and the introduction of information and communication and technologies ” and aims to create an electronic training manual on the subject of “Mathematical and logical foundation digital devices ” for the Tashkent University of Information Technologies, Fergana branch.

This paper discusses issues related to the creation of the web site of the lyceum. This web site gives an opportunity for everybody to gain the information about this lyceum. It contains an e-library, on-line tests and news blogs.

This site is developed with the help of framework CodeInegter and technology MVS.

All students of this lyceum can gain the information and to check their knowledge in on-line format.

This work consists of four parts, introduction, conclusion, bibliography.

# INTRODUCTION

By the turn of the century, information, including access to the Internet, will be the basis for personal, economic, and political advancement. The popular name for the Internet is the information superhighway. Whether you want to find the latest financial news, browse through library catalogs, exchange information with colleagues, or join in a lively political debate, the Internet is the tool that will take you beyond telephones, faxes, and isolated computers to a burgeoning networked informationfrontier.

The Internet supplements the traditional tools you use to gather information, Data Graphics, News and correspond with other people. Used skillfully, the Internet shrinks the world and brings information, expertise, and knowledge on nearly every subject imaginable straight to your computer.

The Internet links are computer networks all over the world so that users can share resources and communicate with each other. Some computers, have direct access to all the facilities on the Internet such as the universities. And other computers, eg privately-owned ones, have indirect links through a commercial service provider, who offers some or all of the Internet facilities. In order to be connected to Internet, you must go through service suppliers. Many options are offered with monthly rates. Depending on the option chosen, access time may vary.

The Internet is what we call a Meta network, that is, a network of networks that spans the globe. It's impossible to give an exact count of the number of networks or users that comprise the Internet, but it is easily in the thousands and millions respectively. The Internet employs a set of standardized protocols which allow for the sharing of resources among different kinds of computers that communicate with each other on the network. These standards, sometimes referred to as the Internet Protocol Suite, are the rules that developers adhere to when creating new functions for the Internet.

The Internet is also, what we call a distributed system; there is no central archives.

Technically, no one runs the Internet. Rather, the Internet is made up of thousands of smaller networks. The Internet thrives and develops as its many users find new ways to create, display and retrieve the information that constitutes the Internet.

## History and Development of the Internet:

In its infancy, the Internet was originally conceived by the Department of Defense as a way to protect government communications systems in the event of a military strike. The original network, dubbed ARPANet (for the Advanced Research Projects Agency that developed it) evolved into a communications channel among contractors, military personnel, and university researchers who were contributing to ARPA projects.
The network employed a set of standard protocols to create an effective way for these people to communicate and share data with each other.
ARPAnet's popularity continued to spread among researchers, and in the 1980's the National Science Foundation, whose NSFNet, linked several high speed computers, and took charge of what had come to be known as the Internet.
By the late 1980's, thousands of cooperating networks were participating in the Internet.
In 1991, the U.S. High Performance Computing Act established the NREN (National Research & Education Network). NREN's goal was to develop and maintain high-speed networks for research and education, and to investigate commercial uses for the Internet.
The rest, as they say, is history in the making. The Internet has been improved through the developments of such services as Gopher and the World Wide Web.
Even though the Internet is predominantly thought of as a research oriented network, it continues to grow as an informational, creative, and commercial resource every day and all over the world.

Everything has to begin somewhere, so let’s start the Web StandardsCurriculum with a focused history lesson. Below we'll give you a brief overview of the creation of the Internet, the World Wide Web, and the web standards that this entire series focuses upon.

The Internet’s origins

On the fourth of October in 1957 an event occurred that would change the world. The Soviet Union successfully launched the first satellite into Earth’s orbit. Called Sputnik 1, it shocked the world — especially the United States of America, who had their own programme of satellite launches underway, but had yet to launch.

This event lead directly to the creation of the US Department of Defence, ARPA, due to a recognised need for an organisation that could research and develop advanced ideas and technology. Perhaps their most famous project (certainly the most widely used) was the creation of the Internet.

In 1960, psychologist and computer scientist Joseph Licklider published a paper entitled Man-Computer Symbiosis, which articulated the idea of networked computers providing advanced information storage and retrieval. In 1962, whilst working for ARPA as the head of the information processing office, he formed a group to further computer research, but left the group before any actual work was done on the idea.

The plan for this computer network (to be called ARPANET) was presented in October 1967, and in December 1969 the first four-computer network was up and running. The core problem in creating a network was how to connect separate physical networks without tying up network resources for constant links. The technique that solved this problem is known as packet switching and it involves data requests being split into small chunks (packets,) which can be processed quickly without blocking communication from other parties — this principle is still used to run the Internet today.

This concept received wider adoption, with several other networks springing up using the same packet switching technique — for example, X.25 (developed by the International Telecommunication Union) formed the basis of the first UK university network JANET (allowing UK universities to send and receive files and emails) and the American public network CompuServe (a commercial enterprise allowing small companies and individuals access to time-shared computer resources, and then later Internet access.) These networks, despite having many connections, were more like private networks than the Internet of today.

This proliferation of different networking protocols soon became a problem, when trying to get all the separate networks to communicate. There was a solution in sight however—Robert Kahn, whilst working on a satellite packet network project for ARPA, started defining some rules for a more open networking architecture to replace the current protocol used in ARPANET. Later joined by Vinton Cerf from Stanford University, the two created a system that masked the differences between networking protocols using a new standard. In the publication of the draft specification in December 1974, this was called the Internet Transmission Control Program.

This specification reduced the role of the network and moved the responsibility of maintaining transmission integrity to the host computer. The result was that it became possible to easily join almost all networks together. ARPA funded development of the software, and in 1977 a successful demonstration of three different networks communicating was conducted. By 1981, the specification was finalised, published and adopted; and in 1982 the ARPANET connections outside of the US were converted to use the new TCP/IP protocol. The Internet as we know it had arrived.

# I ANALYTICAL PART

## 1.1 The creation of World Wide Web

Gopher was an information retrieval system used in the early 1990s, providing a method of delivering menus of links to files, computer resources and other menus. These menus could cross the boundaries of the current computer and use the Internet to fetch menus from other systems. It was very popular with universities looking to provide campus-wide information and large organisations looking to centralise document storage and management.

Gopher was created by the University of Minnesota. In February, 1993, they announced that it was going to charge licensing fees for the use of their reference implementation of the Gopher server. As a consequence, many organisations started to look for alternatives to Gopher.

The European Council for Nuclear Research (CERN) in Switzerland had such an alternative. Tim Berners-Lee had been working on a information management system, in which text could contain links and references to other works, allowing the reader to quickly jump from document to document. He had created a server for publishing this style of document (called hypertext) as well as a program for reading them, which he had called WorldWideWeb. This software had first been released in 1991, however, it took two events to cause an explosion in popularity and the eventual replacement of Gopher.

On the thirtieth of April 1993 CERN released the source code of WorldWideWeb into the public domain, so anyone could use or build upon the software without charge.

Then, later in the same year, the NCSA released a program that was a combined web browser and Gopher client, called Mosaic. This was originally only available on UNIX machines and in source code form, but in December 1993 Mosaic provided a new version with installers for both Apple Macintosh and Microsoft Windows. Mosaic rapidly increased in popularity, and with it the Web.

The number of available web browsers increased dramatically, many created by research projects at universities and corporations, such as Telenor (a Norwegian communications company), which created the first version of the Opera browser in 1994.

The browser wars

The popularization of the web brought commercial interests. Marc Andreessen left NCSA and together with Jim Clark founded Mosaic Communications, later renamed to Netscape Communications Corporation, and started work on what was to become Netscape Navigator. Version 1.0 of the software was released in December 1994.

Spyglass Inc. (the commercial arm of NCSA) licensed their Mosaic technology to Microsoft to form the basis of Internet Explorer. Version 1.0 was released in August 1995.

A rapid escalation soon followed, with Netscape and Microsoft each trying to get a competitive edge in terms of the features they support in order to attract developers. This has since become known as the browser wars. Opera maintained a small but steady presence throughout this period, and tried to innovate and support web standards as well as possible in these times.

The coming of web standards

During the browser wars, Microsoft and Netscape focused on implementing new features rather than on fixing problems with the features they already supported, and adding proprietary features and creating features that were in direct competition with existing features in the other browser, but implemented in an incompatible way.

## The formation of the W3C

In 1994, Tim Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology, with support from CERN, DARPA (as ARPA had been renamed to) and the European Commission. The W3C's vision was to standardize the protocols and technologies used to build the web such that the content would be available to as wide a population of the world as possible.

During the next few years, the W3C published several specifications (called recommendations) including HTML 0.4 the format for PNG images and Cascading Style Sheets versions 1 and 2.

However, the W3C did not (and still do not) enforce their recommendations. Manufacturers only had to conform to the W3C documents if they wished to label their products as W3C-compliant. In practice, this was not a valuable selling point as almost all users of the web did not know, nor probably care, who the W3C were (this is still the case, to a large extent). Consequently, the browser wars of the nineties continued unabated.

## The Web Standards Project

In 1998, the browser market was dominated by Internet Explorer 4 and Netscape Navigator 4. A beta version of Internet Explorer 5 was then released, and it implemented a new and proprietary dynamic HTML, which meant that professional web developers needed to know five *different* ways of writing JavaScript.

As a result, a group of professional web developers and designers banded together. This group called themselves the Web Standards Project (WaSP). The idea was that by calling the W3C documents standards rather than recommendations, they might be able to convince Microsoft and Netscape to support them.

The early method of spreading the call to action was to use a traditional advertising technique called a roadblock, where a company would take out an advert on all broadcast channels at the same time, so no matter how a viewer would flick between channels, they would get exactly the same message. The WaSP published an article simultaneously on various web development focused sites including builder.com, Wired online, and some popular mailing lists.

Another technique the WaSP used was to ridicule the companies involved with the W3C (and other standards bodies) that focused more on creating new, often self-serving, features rather than working to get the basic existing standards supported correctly in their products to start with (this includes some browser companies that shall remain nameless here). This doesn't mean that the WaSP ridiculed the W3C themselves, rather they ridiculed the companies that became W3C members and then misbehaved.

The W3C has a few full time staff, but most of the people who work on the standards are volunteers from member companies, eg Microsoft, Opera, Mozilla, Apple, Google, IBM and Adobe, to name a few of the bigger ones.

This all sounds a bit negative, but the WaSP didn't just sit there criticising people — they also helped. Seven members formed the CSS Samurai, who identified the top ten problems with the CSS support in Opera and other browsers (Opera fixed their problems, others did not).

**The rise of web standards**

In 2000, Microsoft released Internet Explorer 5 Macintosh Edition. This was a very important milestone, it being the default browser installed with the Mac OS at the time, and having a reasonable level of support for the W3C recommendations too. Along with Opera's decent level of support for CSS and HTML, it contributed to a general positive movement, where web developers and designers finally felt comfortable designing sites using web standards, as they knew they would work to a reasonable level across multiple browsers.

The WaSP persuaded Netscape to postpone the release of the 5.0 version of Netscape Navigator until it was much more compliant (this work formed the basis of what is now Firefox, a very popular browser). The WaSP also created a

Dreamweaver Task Force to encourage Macromedia to change their popular web authoring tool to encourage and support the creation of compliant sites.

The popular web development site A List Apart was redesigned early in 2001 and in an article describing how and why, stated:

In six months, a year, or two years at most, all sites will be designed with these standards. We can watch our skills grow obsolete, or start learning standards-based techniques now.

That was a little optimistic — not all sites, even in 2008, are built with web standards. But many people listened. Older browsers decreased in market share, and two very high profile sites redesigned using web standards: Wired magazine in 2002, and ESPN in 2003 became field leaders in supporting web standards and new techniques.

Also in 2003, Dave Shea launched a site called the CSS Zen Garden. This was to have more impact on web professionals than anything else, by truly illustrating that the entire design can change just by changing the style of the page; the content could remain identical.

Since then, web standards usage have become de rigeur in the professional web development community. And in this series, we will give you an excellent grounding in these techniques so that you can develop clean, semantic, accessible and standards-compliant websites!

## Thenewbreedofwebstandards

After 2003, web standards didn't just sit still. New practices started to really come to the forefront, with many web sites being more like desktop applications than static pages. This new breed of sites is way more complicated than what the web was really intended for, and we still have to concern ourselves with making them semantic, accessible and usable!

When HTML 4 was nearing completion, the W3C decided (in a workshop run in 1998) that in terms of markup languages, the future of the Web was XML and XHTML, not HTML (comparison of XHTML and HTML). So the W3C drew a line under HTML 4.01 and instead concentrated on the XHTML 1.0 spec, finished in early 2000. XHTML 1.0 is just the same as HTML 4.01, except that it uses the stricter markup syntax rules of XML (more on this later). XHTML 2.0 soon followed, which added a whole bunch of new powerful features, and aimed to be the next big thing on the Web.

The trouble with XHTML 2.0 is that it wasn't backwards compatible with the markup already on the Web — the elements worked differently, XHTML did not work properly in Internet Explorer, which still has the majority browser market share as of the time of writing, the developer tools available weren't ready for working with XML, and it didn't reflect what web developers were REALLY doing out there in the wild wild web.

In 2004, a group of like-minded developers and implementers (including representatives from Opera, Mozilla and slightly later, Apple) got together and formed a breakaway spec group called theWHATWG, with the aim of writing a better HTML markup spec that could handle authoring the new breed of web applications, without — crucially — breaking backwards compatibility.

The result was the Web Applications 1.0 specification, which documented existing interoperable browser behaviors and features, as well as new features for the Web stack such as APIs and new DOM parsing rules. After many discussions between W3C Members, on March 7 2007 the work on HTML was restarted with a new HTML Working Group in an open participation process. In the first few days, hundreds of participants joined to continue to work on the next version of HTML. One of the first decisions of the HTML WG was to adopt the Web Applications 1.0 spec and call it HTML5.

HTML5 is a really good thing for web developers and designers, because it:

- Is mostly backwards compatible with what's already there — you don't need to learn completely new languages to use HTML5. The new markup features work in the same way as the old ones (although the semantics of some elements have been changed — we will cover these differences in a future article), and the new APIs are based on mostly the same JavaScript/DOM that developers have been programming in for years.
- Adds powerful new features to HTML that were previously only available on the Web using plugin technologies like Flash, or with complex JavaScript and hacks. Formvalidationandvideoareprimeexamples.
- Has a clearly defined parsing algorithm so that all browsers implementing HTML5 will create the same DOM from the same markup, regardless of validity. This is a massive win for interoperability.

The evolution of CSS is not nearly as long winded and controversial as that of HTML, but it is still very interesting, and worth a mention here. The CSS2 specification was nearing completion is around 1999/2000, and although it was a powerful language with many great features, its creators knew that it had limitations. There were a number of visual/stylistic things that CSS couldn't do, and that developers had to turn to hacks, JavaScript or plugins to achieve. This includes things such as animation, dynamic layouts, and using custom fonts on pages.

To begin to address this, work started on CSS3 as early as 2000. The spec writers decided on a modular structure, with different pieces of distinct functionality being broken down into manageable chunks. This made it easier not only for the writers to write, but also for the browsers to implement, and the web designers/developers to learn. A lot more features have been added since the first spec version in 2000, and we did not start to see browser support for many of the features until about 2006. At the time of writing, CSS3 has over 40 modules in various stages of completion. You can track their progress and find out more at the W3C CSS current work & how to participate page.

**Summary**

In this article we have looked at how the modern Internet was created as a result of the space race, how Tim Berners-Lee defined hypertext for a generation and how the commercial interests of two companies caused one of the most notable developer backlashes ever seen. The term web standards is now more widely used by web professionals that any other term applied by the W3C (in fact the W3C have started to use the term on their own pages), so that is what we are going to teach you — the standards way to build web sites

## 1.1 Development of HTML,CSS and PHP history

### HTML

HTML or Hypertext Markup Language is the standard markup language used to create web pages.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the look and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicit presentational HTML.

In 1980, physicist Tim Berners-Lee, who was a contractor at CERN, proposed and prototyped ENQUIRE, a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system. Berners-Lee specified HTML and wrote the browser and server software in late 1990. That year, Berners-Lee and CERN data systems engineer Robert Cailliau collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes from 1990 he listed "some of the many areas in which hypertext is used" and put an encyclopedia first.

1.1.2Tim Berners-Lee

The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Berners-Lee in late 1991. It describes 18 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by SGMLguid, an in-house SGML-based documentation format at CERN. Eleven of these elements still exist in HTML

Hypertext Markup Language is a markup language that web browsers use to interpret and compose text, images and other material into visual or audible web pages. Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of CSS. Many of the text elements are found in the 1988 ISO technical report TR 9537Techniques for using SGML, which in turn covers the features of early text formatting languages such as that used by the RUNOFF command developed in the early 1960s for the CTSS (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and markup; HTML has been progressively moved in this direction with CSS.

HTML markup consists of several key components, including tags (and their attributes), character-based data types, character references and entity references. Another important component is the document type declaration, which triggers standards mode rendering.

The following is an example of the classic Hello world program, a common test employed for comparing programming languages, scripting languages and markup languages. This example is made using 9 lines of code:

```
<! DOCTYPE html>

<Html>
```

```
<Head>

<Title>This is a title</title>

</head>

<body>

<p>Hello world! </p>

</body>

</html>
```

(The text between <html> and </html> describes the web page, and the text between <body> and </body> is the visible page content. The markup text '<title>this is a title</title>' defines the browser page title.)

This Document Type Declaration is for HTML5. If the <! DOCTYPE html> declaration is not included, various browsers will revert to "quirks mode" for rendering.

Elements

Main article: HTML element

HTML documents imply a structure of nested HTML elements. These are indicated in the document by HTML tags, enclosed in angle brackets thus: <p>

In the simple, general case, the extent of an element is indicated by a pair of tags: a 'start tag' <p> and 'end tag’. The text content of the element, if any, is placed between these tags.

Tags may also enclose further tag markup between the start and end, including a mixture of tags and text. This indicates further, nested, elements, as children of the parent element.

The start tag may also include attributes within the tag. These indicate other information, such as identifiers for sections within the document, identifiers used to bind style information to the presentation of the document, and for some tags such as the <img> used to embed images, the reference to the image resource.

Some elements, such as the line break <br>, do not permit any embedded content, either text or further tags. These require only a single empty tag (akin to a start tag) and do not use an end tag.

Many tags, particularly the closing end tag for the very commonly-used paragraph element <p>, are optional. An HTML browser or other agent can infer the closure for the end of an element from the context and the structural rules defined by the HTML standard. These rules are complex and not widely understood by most HTML coders.

The general form of an HTML element is therefore: <tag attribute1="value1" attribute2="value2">content</tag>. Some HTML elements are defined as empty elements and take the form <tag attribute1="value1" attribute2="value2" >. Empty elements may enclose no content, for instance, the BR tag or the inline IMG tag. The name of an HTML element is the name used in the tags. Note that the end tag's name is preceded by a slash character, "/", and that in empty elements the end tag is neither required nor allowed. If attributes are not mentioned, default values are used in each case.

Element examples

Header of the HTML document :< head>...</head>. The title is included in the head, for example:

<Head>

<Title>The Title</title>

</head>

Headings: HTML headings are defined with the <h1> to <h6> tags:

<h1>Heading level 1</h1>

<h2>Heading level 2</h2>

<h3>Heading level 3</h3>

<h4>Heading level 4</h4>

<h5>Heading level 5</h5>

<h6>Heading level 6</h6>

Line breaks :<br>. The difference between <br> and <p> is that 'br' breaks a line without altering the semantic structure of the page, whereas 'p' sections the page intoparagraphs. Note also that 'br' is an empty element in that, while it may have attributes, it can take no content and it may not have an end tag.

<p>This<br> is a paragraph <br> with <br> line breaks</p>

This is a link in HTML. To make a link you use the <a> tag. The href= attribute holds the URL address of the link.

<a href="http://www.google.com/">A Link to Google! </a>

Comments:

<! -- This is a comment -->

Comments can help in the understanding of the markup and do not display in the webpage.

There are several types of markup elements used in HTML:

Structural markup describes the purpose of text

For example, <h2>Golf</h2> establishes "Golf" as a second-level heading. Structural markup does not denote any specific rendering, but most web browsers have default styles for element formatting. Content may be further styled using Cascading Style Sheets (CSS).

Presentational markup describes the appearance of the text, regardless of its purpose

For example <b>boldface</b> indicates that visual output devices should render "boldface" in bold text, but gives little indication what devices that are unable to do this (such as aural devices that read the text aloud) should do. In the case of both <b>bold</b> and <i>italic</i>, there are other elements that may have equivalent visual renderings but which are more semantic in nature, such as <strong>strong text</strong> and <em>emphasised text</em> respectively. It is easier to see how an aural user agent should interpret the latter two elements. However, they are not equivalent to their presentational counterparts: it would be undesirable for a screen-reader to emphasize the name of a book, for instance, but on a screen such a name would be italicized. Most presentational markup elements have become deprecated under the HTML 4.0 specification in favor of using CSS for styling.

**CSS**

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to Simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the Text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantagesof CSS:

- CSS saves time - You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web.
- Pages load faster - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply to all the occurrences of that tag. So less code means faster download times.
- Easy maintenance - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- Superior styles to HTML - CSS has a much wider array of attributes than HTML so
- You can give far better look to your HTML page in comparison of HTML attributes.
- Multiple Device Compatibility - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- Global web standards- Now HTML attributes are being deprecated and it is being recommended to use CSS. So it’s a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

**Who Creates and Maintains CSS?**

CSS is created and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called specifications. When a specification has been discussed and officially ratified by W3C members, it becomes a recommendation. These ratified specifications are called recommendations because the W3C has no control over

the actual implementation of the language. Independent companies and organizations create that software.

NOTE: The World Wide Web Consortium, or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

**CSS Versions:**

Cascading Style Sheets, level 1 (CSS1) was came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 was became a W3C recommendation in May 1998 and builds on CSS1. This version adds support for media-specific style sheets e.g. Printers and aural devices, download able fonts, element positioning and tables.

**CSS Syntax – Selectors**

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A styleruleismadeofthreeparts:

- Selector: A selector is an HTML tag at which style will be applied. This could be any tag like <h1> or <table> etc.
- Property: A property is a type of attribute of HTML tag. Put simply, a the HTML attributes are converted into CSS properties. They could be color or border etc.
- Value: Values are assigned to properties. For example color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule Syntax as follows:

selector{ property: value }

Example: You can define a table border as follows:

table{ border :1px solid #C00; }

Here table is a selector and border is a property and given value 1px solid #C00 is the value of that property. You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one. **The Type Selectors:**

This is the same selector we have seen above. Again one more example to give a color to all level 1 headings: h1 {

color: #36CFFF; }

**TheUniversalSelectors:**

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type:* {

Color: #000000;

}

This rule renders the content of every element in our document in black.

**The Descendant Selectors:**

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to <em> element only when it lies inside <ul> tag.

ulem{

Color: #000000; }

**PHP** is a server-side scripting language designed for web development but also used as a general-purpose programming language. As of January 2013, PHP was installed on more than 240 million websites (39% of those sampled) and 2.1 million web servers. Originally created by RasmusLerdorf in 1995, the reference implementation of PHP is now produced by The PHP Group. While PHP originally stood for *Personal Home Page*, it now stands for *PHP: Hypertext Preprocessor*, a recursive backronym.

PHP code is interpreted by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone graphical applications.

PHP is free software released under the PHP License. PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

PHP development began in 1994 when the developer RasmusLerdorf wrote a series of Common Gateway Interface (CGI) Perl scripts, which he used to maintain his personal homepage. The tools performed tasks such as displaying his résumé and recording his web traffic. He rewrote these scripts in C for performance reasons, extending them to add the ability to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

PHP/FI could be used to build simple, dynamic web applications. Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" publicly to accelerate bug location and improve the code, on the Usenet discussion group *comp.infosystems.www.authoring.cgi* on June 8, 1995. This release already had the basic functionality that PHP has as of 2013.

This included Perl-like variables, form handling, and the ability to embed HTML. The syntax resembled that of Perl but was simpler, more limited and less consistent.

Early PHP was not intended to be a new programming language, and grew organically, with Lerdorf noting in retrospect: "I don't know how to stop it, there was never any intent to write a programming language […] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way." A development team began to form and, after months of work and beta testing, officially released PHP/FI 2 in November 1997.

One criticism of PHP is that it was not originally designed, but instead it was developed organically; among other things, this has led to inconsistent naming of functions and inconsistent ordering of their parameters. In some cases, the function names were chosen to match the lower-level libraries which PHP was "wrapping", while in some very early versions of PHP the length of the function names was used internally as a hash function, so names were chosen to improve the distribution of hash values.

ZeevSuraski and AndiGutmans rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive acronym *PHP: Hypertext Preprocessor*. Afterwards, public testing of PHP 3 began, and the official launch came in June 1998. Suraski and Gutmans then started a new rewrite of PHP's core, producing the Zend Engine in 1999. They also founded Zend Technologies in Ramat Gan, Israel.

On May 22, 2000, PHP 4, powered by the Zend Engine 1.0, was released. As of August 2008 this branch reached version 4.4.9. PHP 4 is no longer under development nor will any security updates be released.

On July 13, 2004, PHP 5 was released, powered by the new Zend Engine II. PHP 5 included new features such as improved support for object-oriented programming, the PHP Data Objects (PDO) extension (which defines a lightweight and consistent interface for accessing databases), and numerous performance enhancements. In 2008 PHP 5 became the only stable version under development. Late static binding had been missing from PHP and was added in version 5.3.

PHP interpreters are available on most existing 32-bit and 64-bit operating systems, either by building them from the PHP source code, or by using pre-built binaries.[ For the PHP versions 5.3 and 5.4, the only available Microsoft Windows binary distributions were 32-bit x86 builds, requiring Windows 32-bit compatibility mode while using Internet Information Services (IIS) on a 64-bit Windows platform. PHP version 5.5 made the 64-bit x86-64 builds available for Microsoft Windows.

## 1.2 Using MySQL database for web applications.

**MySQL** is (as of March 2014) the world's second most widely used open-source relational database management system (RDBMS). It is named after co-founder Widleness's daughter, my. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

## Interfaces

MySQL is a relational database management system (RDBMS), and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command linetools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, MySQL Workbench is actively developed by Oracle, and is freely available for use.

## Graphical

The official MySQL Workbench is a free integrated environment developed by MySQL AB that enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL front end, MySQL Workbench lets users manage database design & modeling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).

MySQL Workbench is available in two editions, the regular free and open source *Community Edition* which may be downloaded from the MySQL website, and the proprietary *Standard Edition* which extends and improves the feature set of the Community Edition.

Third-party proprietary and free graphical administration applications (or "front ends") are available that integrate with MySQL and enable users to work with database structure and data visually. Somewell-knownfrontends, inalphabeticalorder, are:

- Adminer – a free MySQL front end written in one PHP script, capable of managing multiple databases, with many CSS skins available.
- DBEdit – a free front end for MySQL and other databases.
- HeidiSQL – a full featured free front end that runs on Windows, and can connect to local or remote MySQL servers to manage databases, tables, column structure, and individual data records. Also supports specialised GUI features for date/time fields and enumerated multiple-value fields.
- LibreOffice Base – LibreOffice Base allows the creation and management of databases, preparation of forms and reports that provide end users easy access to data. Like Microsoft Access, it can be used as a front-end for various database systems, including Access databases (JET), ODBC data sources, and MySQL or PostgreSQL.
- Navicat – a series of proprietary graphical database management applications, developed for Windows, Macintosh and Linux.
- OpenOffice.org – OpenOffice.org Base can manage MySQL databases if the entire suite is installed. Freeandopen-source.
- phpMyAdmin – a free Web-based front end widely installed by web hosts, since it is developed in PHP and is included in the LAMP stack, MAMP, XAMPP and WAMP software bundle installers.
- Webmin – a free Web-based management utility and a MySQL front end, developed in Perl with some parts written in Java.
- SQLBuddy – a free Web-based front end, developed in PHP.
- SQLyog – commercial, but there is also a free 'community' edition available.
- Toad for MySQL – a free development and administration front end for MySQL from Quest Software
- Chive – is a free, open source, web-based database management tool designed as an alternative to phpMyAdmin

Other available proprietary MySQL front ends include dbForge Studio for MySQL, DBStudio, Epictetus, Microsoft Access, Oracle SQL Developer, SchemaBank, SQLPro SQL Client, Toad Data Modeler andDaDaBIK.

## Command line

MySQL ships with many command line tools, from which the main interface is 'MySQL' client. Third-parties have also developed tools to manage MySQL servers.

- MySQL Utilities – a set of utilities designed to perform common maintenance and administrative tasks. Originally included as part of the MySQL Workbench, the utilities are now a stand-alone download available from Oracle.

- Perc
- One Toolkit – a cross-platform toolkit for MySQL, developed in Perl. Percona Toolkit can be used to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up servers. Percona Toolkit is included with several Linux distributions such as CentOS and Debian, and packages are available for Fedora and Ubuntu as well. Percona Toolkit was originally developed as Maatkit, but as of late 2011, Maatkit is no longer developed.

## Programming

MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp,QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the JDBC driver for Java. In addition, an ODBC interface called MyODBC allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The HTSQL – URL-based query method also ships with a MySQL adapter, allowing direct interaction between a MySQL database and any web client via structured URLs.

## History

MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from mSQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as mSQL. By keeping the API consistent with the mSQL system, many developers were able to use MySQL instead of the (commercially licensed) mSQL antecedent.

**PHP** is a server-side scripting language designed for web development but also used as a general-purpose programming language. As of January 2013, PHP was installed on more than 240 million websites (39% of those sampled) and 2.1 million web servers. Originally created by RasmusLerdorf in 1994, the reference implementation of PHP is now produced by The PHP Group. While PHP originally stood for *Personal Home Page*, it now stands for *PHP: Hypertext Preprocessor*, a recursive backronym.

PHP code can be simply mixed with HTML code, or it can be used in combination with various templating engines and web frameworks. PHP code is usually processed by a PHP interpreter, which is usually implemented as a web server's native module or a Common Gateway Interface (CGI) executable. After the PHP code is interpreted and executed, the web server sends resulting output to its client, usually in form of a part of the generated web page – for example, PHP code can generate a web page's HTML code, an image, or some other data. PHP has also evolved to include a command-line interface (CLI) capability and can be used in standalone graphical applications.

PHP is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

Early PHP was not intended to be a new programming language, and grew organically, with Lerdorf noting in retrospect: "I don't know how to stop it, there was never any intent to write a programming language […] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way." A development team began to form and, after months of work and beta testing, officially released PHP/FI 2 in November 1997.

One criticism of PHP is that it was not originally designed, but instead it was developed organically among other things, this has led to inconsistent naming of functions and inconsistent ordering of their parameters. In some cases, the function names were chosen to match the lower-level libraries which PHP was "wrapping", while in some very early versions of PHP the length of the function names was used internally as a hash function, so names were chosen to improve the distribution of hash values.

## Syntax

The following Hello world program is written in PHP code embedded in an HTML document:

```
<! DOCTYPE html>
<html>
```

```
<head>
<title>PHP Test</title>
</head>
<body>
<?php echo '<p>Hello World</p>'; ?>
</body>
</html>
```

However, as PHP does not need to be embedded in HTML or used with a web server, the simplest version of a Hello World program can be written like this, with the closing tag omitted as preferred in files containing pure PHP code (prior to PHP 5.4.0, this short syntax for echo () only works with the short_open_tag configuration setting enabled, while for PHP 5.4.0 and later it is always available):

```
<? = 'Hello world';
```

The PHP interpreter only executes PHP code within its delimiters. Anything outside its delimiters is not processed by PHP (although non-PHP text is still subject to control structures described in PHP code). The most common delimiters are <? php to open and ?> to close PHP sections. <script language="php"> and </script> delimiters are also available, as are the shortened forms <? or <?= (which is used to echo back a string or variable) and ?> as well as ASP-style short forms <% or <%= and %>. Short delimiters make script files less portable, since support for them can be disabled in the local PHP configuration, and they are therefore discouraged. The purpose of all these delimiters is to separate PHP code from non-PHP code, including HTML.

Variables are prefixed with a dollar symbol, and a type does not need to be specified in advance. Unlike function and class names, variable names are case sensitive. Both double-quoted ("") and heredoc strings provide the ability to interpolate a variable's value into the string. PHP treats newlines as whitespace in the manner of a free-form language (except when inside string quotes), and statements are terminated by a semicolon. PHP has three types of comment syntax: /* */ marks block and inline comments; // as well as # are used for one-line comments. The echo statement is one of several facilities PHP provides to output text, *e.g.*, to a web browser.

**Data types**

PHP stores whole numbers in a platform-dependent range, either a 64-bit or 32-bit signed integer equivalent to the C-language long type. Unsigned integers are converted to signed values in certain situations; this behavior is different from other programming languages. Integer variables can be assigned using decimal (positive and negative), octal, hexadecimal, and binary notations.

Floating point numbers are also stored in a platform-specific range. They can be specified using floating point notation, or two forms of scientific notation. PHP has a native Boolean type that is similar to the native Boolean types in Java and C++. Using the Boolean type conversion rules, non-zero values are interpreted as true and zero as false, as in Perl and C++.

Arrays can contain elements of any type that PHP can handle, including resources, objects, and even other arrays. Order is preserved in lists of values and in hashes with both keys and values, and the two can be intermingled. PHP also supports strings, which can be used with single quotes, double quotes, nowdoc or heredoc syntax.

**Functions**

PHP has hundreds of functions provided by the core language functionality and thousands more available via various extensions; these functions are well documented in the online PHP documentation. However, the built-in library has a wide variety of naming conventions and associated inconsistencies, as described under history above.

Additional functions can be defined by the developer:

```
FunctionmyAge($birthYear)                                   // defines a function, this one is named "myAge"
{
   $yearsOld = date('Y') - $birthYear;                      // calculates the age
return $yearsOld . ' year' . ($yearsOld != 1 ? 's' : '');   // returns the age in a descriptive form
}
echo 'I am currently ' . myAge(1981) . ' old.';             // outputs the text concatenated
                                                            // with the return value of myAge()
// As the result of this syntax, myAge() is called.
// In 2014, the output of this sample program will be 'I am currently 33 years old.'
```

PHP supports quasi-anonymous functions through the create_function() function. However, they are not truly anonymous because they can be referenced either by name, or indirectly as variable functions using$function_name() syntax, as seen in the following example:

```
$f = create_function('$a, $b = 5', 'var_dump($a, $b);');
$f(1);

// output:
// int(1)
// int(5)
```

PHP 5.3 added support for closures, which enable true anonymous functions. The syntax can be seen in the following example:

```
functiongetAdder($x)
{
return function($y) use ($x)
    {
return $x + $y;
    };
}
$adder = getAdder(8);
Echo $adder (2); // prints "10"
```

Here, the getAdder() function creates a closure using passed argument $x (the keyword use imports a variable from the lexical context), which takes an additional argument $y, and returns the created closure to the caller. Such a function is a first-class object, meaning that it can be stored in a variable, passed as a parameter to other functions, etc. For more details, see the Lambda functions and closures RFC.

The goto flow control statement is used as in the following example:

```
Functionlock()
{
    $file = fopen('file.txt', 'r+');
Retry:
If (! flock ($file, LOCK_EX | LOCK_NB))
goto retry;

fwrite($file, 'Success!');
fclose($file);
}
```

When flock() is called, PHP opens a file and tries to lock it. The target label retry: defines the point to which execution should return if flock() is unsuccessful and goto retry; is called. The goto statement is restricted and requires that the target label be in the same file and context.

## Objects

Basic object-oriented programming functionality was added in PHP 3 and improved in PHP 4. Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance. In previous versions of PHP, objects were handled like value types. The drawback of this method was that the whole object was copied when a variable was assigned or passed as a parameter to a method. In the new approach, objects are referenced by handle, and not by value.

The following is a basic example of object-oriented programming in PHP:

```
class Person
{
public $firstName;
public $lastName;

public function __construct($firstName, $lastName = ") { // optional second
argument
        $this->firstName = $firstName;
        $this->lastName  = $lastName;
    }

public function greet() {
return 'Hello, my name is ' . $this->firstName .
            (($this->lastName != ") ? (' ' . $this->lastName) : ") . '.';
    }
public static function staticGreet($firstName, $lastName) {
return 'Hello, my name is ' . $firstName . ' ' . $lastName . '.';
    }
}
$he    = new Person('John', 'Smith');
$she   = new Person('Sally', 'Davis');
$other = new Person('iAmine');
echo $he->greet(); // prints "Hello, my name is John Smith."
echo '<br />';
echo $she->greet(); // prints "Hello, my name is Sally Davis."
echo '<br />';
echo $other->greet(); // prints "Hello, my name is iAmine."
echo '<br />';
echo Person::staticGreet('Jane', 'Doe'); // prints "Hello, my name is Jane Doe."
```

The visibility of PHP properties and methods is defined using the keywords public, private, and protected. The default is public, if only var is used; var is a synonym for public. Items declared public can be accessed everywhere. protected limits access to inherited classes (and to the class that defines the item). private limits visibility only to the class that defines the item. Objects of the same type have access to each other's private and protected members even though they are not the same instance. PHP's member visibility features have sometimes been described as "highly useful." However, they have also sometimes been described as "at best irrelevant and at worst positively harmful."

## II. PRACTICAL PART

### 1.1 GettingStartedWithCodeIgniter

CodeIgniter is an open source rapid development web application framework, for use in building dynamic web sites with PHP. The first public version of CodeIgniter was released on February 28, 2006, and the latest stable version 2.1.4 was released July 8, 2013.

CodeIgniter is loosely based on the popular Model-View-Controller development pattern. While view and controller classes are a necessary part of development under CodeIgniter, models are optional.

CodeIgniter is most often noted for its speed when compared to other PHP frameworks. In a critical take on PHP frameworks in general, PHP creator RasmusLerdorf spoke at frOSCon in August 2008, noting that he liked CodeIgniter "because it is faster, lighter and the least like a framework."

CodeIgniter's source code is maintained at GitHub, and as of the preview version 3.0-dev, is certified open source software licensed with the Open Software License ("OSL") v. 3.0. Versions of CodeIgniter prior to 3.0 are licensed under a proprietary Apache/BSD-style open source license.

The decision to switch to an OSL license sparked some community controversy, especially about the GPL incompatibility of the new license, to which EllisLab has responded with a series of articles entitled Software License Awareness Week.

On July 9th 2013, EllisLab announced that it is seeking a new owner for its CodeIgniter, stating lack of involvement as a reason. It is not clear whether CodeIgniter will remain the backbone of its Expression Engine software.

### Getting Started

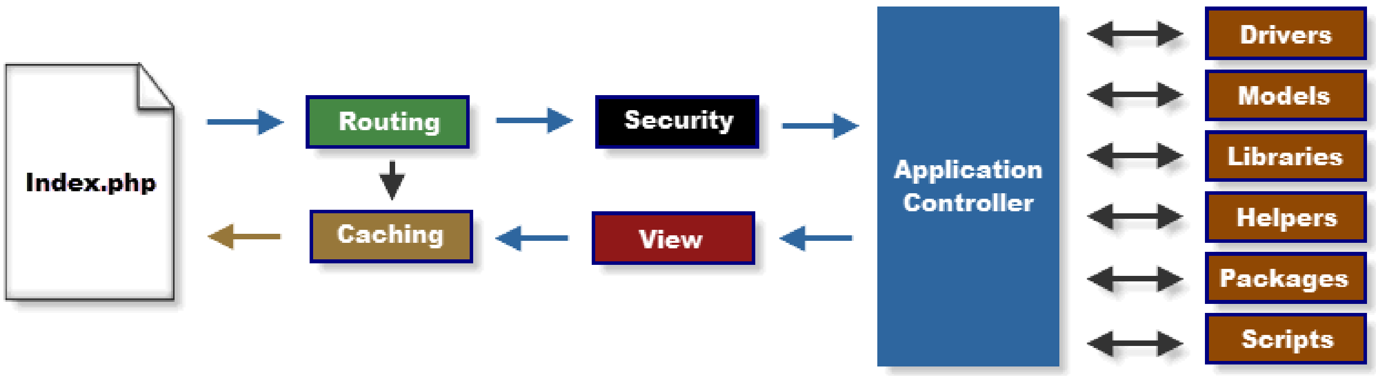
Any software application requires some effort to learn. We've done our best to minimize the learning curve while making the process as enjoyable as possible.The first step is to install CodeIgniter, then read all the topics in the **Introduction** section of the Table of Contents.Next, read each of the **General Topics** pages in order. Each topic builds on the previous one, and includes code examples that you are encouraged to try.Once you understand the basics you'll be ready to explore the **Class Reference** and **Helper Reference** pages to learn to utilize the native libraries and helper files.Feel free to take advantage of our Community Forums if you have questions or problems, and our Wiki to see code examples posted by other users.

## Troubleshooting

If you find that no matter what you put in your URL only your default page is loading, it might be that your server does not support the PATH_INFO variable needed to serve search-engine friendly URLs. As a first step, open your application/config/config.php file and look for the URI Protocol information. It will recommend that you try a couple alternate settings. If it still doesn't work after you've tried this you'll need to force CodeIgniter to add a question mark to your URLs. To do this open yourapplication/config/config.php file and change this:

$config['index_page'] = "index.php";

## Application Flow Chart



2.1.1

The following graphic illustrates how data flows throughout the system:

1. The index.php serves as the front controller, initializing the base resources needed to run CodeIgniter.
2. The Router examines the HTTP request to determine what should be done with it.
3. If a cache file exists, it is sent directly to the browser, bypassing the normal system execution.
4. Security. Before the application controller is loaded, the HTTP request and any user submitted data is filtered for security.
5. The Controller loads the model, core libraries, helpers, and any other resources needed to process the specific request.
6. The finalized View is rendered then sent to the web browser to be seen. If caching is enabled, the view is cached first so that on subsequent requests it can be served.

## 1.2 Building a CodeIgniter Web Application

In this series we're going to build a web billboard application from scratch, we're going to use CodeIgniter to handle the back-end service and BackboneJS for the web client. In the first two parts of the series we'll create the back-end service and then the client application in the last two.

Application Description

The application we are creating will be a simple billboard, where users can register, post tasks, and offer a reward for its completion. Other users can see the existing tasks, assign the task to themselves, and get the offered reward.

The tasks will have basic data like a title, description and reward (as required parameters) and an optional due date and notes. The user profile will simply consist of the user's name, email and website. So let's get started.

## Database Setup

First off, for the app data we're going to use MongoDB as the database server. MongoDB is a document oriented database and the leading NoSQL database out there. It's really scalable and fast, which makes it great to manage huge amounts of data.MongoDB is a document oriented database and the leading NoSQL database.In order to use MongoDB in this application, I'm going to use a MongoDBCodeIgniter driver that I wrote some time ago, it's just a wrapper of the MongoDB PHP driver to mimic the framework's SQL ActiveRecord. You can find the source files for this driver inmy public repository. For this driver to work properly, make sure that you have the PHP's MongoDB driver installed, if you don't, follow these steps to get it working.

Please note that explaining the drivers in CodeIgniter and such is out of the scope of this tutorial, refer to the documentation if you have any doubts. You just need to move the"mongo_db.php" in the "config" folder to the "config" folder of your application and the "Mongo_db" folder in the "libraries" folder to the "libraries" folder in your application.

Database Configuration

The only file we need to edit at this point is the "mongo_db.php" file under the"config" folder, since my mongo installation has all the default parameters, I'm just going to edit line 40 and give it the name of the database that I want to use:

```
$config['mongo_db'] = 'billboard';
```

That's it for the database, one of the many advantages of MongoDB is that the documents have no predefined structure, so it works without us needing to set anything up before using it, our database doesn't even have to exist, MongoDB will create it on the fly when we need it.

## Global Configuration

Other than your regular configuration options, that should include the base_url and theindex_page if any, we need to set the string and date helpers to autoload. I'm not going to walk you through this, since we have much more to cover, when in doubt refer to the documentation.

Other than the helpers, we need to set up the encryption class since we're going to use it for our app.

## URL Handling

This is going to be a RESTful service and we need a way to take the requests coming to the server and handle them accordingly. We could use an existing library (which is great by the way) but for the purposes of this demonstration, I'm going to create the functionality I need using CodeIgniter's core features.

## Handling RESTful Requests

In particular, we're going to use the ability to extend the core classes. We will start with the Controller, for the main part of this extension we're using the "_remap" method in the base controller so all the controllers of our app can use it. Start by creating aMY_Controller.php file inside the "core" folder in the "application" folder, we create this just like any other CodeIgniter controller, as follows:

```
<?php

if( !defined( 'BASEPATH' ) ) exit( 'No direct script access allowed' );

classMY_Controller extends CI_Controller {

}
```

Now in this controller we're going to use the CodeIgniter _remap method to preprocess every request made to the server. Inside the class we just created, add the following method:

```
        public function _remap( $param ) {
   $request = $_SERVER['REQUEST_METHOD'];
switch(strtoupper( $request ) ) {
case 'GET':
          $method = 'read';
break;
case 'POST':
          $method = 'save';
break;
case 'PUT':
          $method = 'update';
break;
case 'DELETE':
          $method = 'remove';
break;
case 'OPTIONS':
          $method = '_options';
break;
```

```
    }
     $this->$method( $id );
}
```

A couple of things to note here, first off, there are some REST verbs that we are ignoring (like PATCH), since I'm demonstrating building a REST app, I don't want to add things that may make this more complex than it needs to be. Secondly, we're not taking into account the case where a controller doesn't implement a particular method, which is very likely that this could happen. Now, we could add a default method to handle such requests, but so that we don't add too much complexity, let's leave it like this. Third, we're receiving a param variable in the method declaration, let's address that, and then I'll explain the OPTIONS request. Abovetheswitchstatement, addthefollowingcode:

```
        if ( preg_match( "/^(?=.*[a-zA-Z])(?=.*[0-9])/", $param ) ) {
   $id = $param;
} else {
   $id = null;
}
```

This regular expression matches any string consisting of uppercase and lowercase letters and any numbers. This is used to check if a MongoDB _id string is being given as a parameter, again, this is not the safest way nor the most thorough check, but for the sake of simplicity, we'll keep it as is.

OPTIONS Request

Since we're building a web service and a client application as separate parts, it makes sense that both are going to be hosted on different domains, so we will enable CORS in the back-end and this means, among other things, that our app will respond properly toOPTIONS requests.

When a web app created with BackboneJS (and some other frameworks) tries to make an asynchronous request to a remote server, it sends an OPTIONS request before sending the actual request it's supposed to send. Among other things, the client tells the server from where it is sending the request, what type of request it is about to send, and the content that it's expecting. After that, it is up to the server to send the client a response where it acknowledges the request or rejects it.

Since our back-end service, no matter which controller is called, is going to receive thisOPTIONS request, it makes sense to implement the method to respond to it in our base controller. Add the following method below (or above) the _remap method in our controller.

deally, we would only allow some domains to make requests to us, we would check therequest_headers header to see if we accept it and we would check for the expected content type, by the client to see if we support it, but again, this is a not-so-complex app and we are skipping these edge cases.

Managing Output

To finish our base controller, let's create a method that every controller will use to send its results back to the client. Inthebasecontrollerclass, addthefollowingmethod:

```
protected function _format_output( $output = null ) {

   $this->output->set_header( 'Access-Control-Allow-Origin: *' );

if(isset( $output->status ) && $output->status == 'error' ) {

      $this->output->set_status_header( 409, $output->desc );

   }
   $this->_parse_data( $output );
    $this->output->set_content_type( 'application/json' );
   $this->output->set_output(json_encode( $output ) );
}
```

Again, in order for BackboneJS to process the server response it has to know that its host is accepted by the server, hence the Allow-Origin header, then, if the result is a faulty one, we set a status header indicating this. This status will become more clear when we create the back-end models. Next we use the parse_data helper, which will be a private method (that we will write in a moment) but let me skip that for the time being, then we set the content type as JSON and finally we encode the response as a JSON object. Again, here we could (and should) support other output formats (like XML).

Now let's create the parse_data helper method (and I'll explain it afterwards), add the following code to the base controller:

```
         private function _parse_data( &$data ) {
   if ( ! is_array( $data ) && ! is_object( $data ) )
return $data;
foreach ( $data as $key => $value ) {
if ( is_object( $value ) || is_array( $value ) ) {
if(is_object( $data ) ) {
              $data->{$key} = $this->_parse_data( $value );
            } else {
              $data[ $key ] = $this->_parse_data( $value );
            }
        }
if ( isset( $value->sec ) ) {
if(is_object( $data ) ) {
                $data->{$key} = date( 'd.m.Y', $value->sec );
            } else {
                $data[ $key ] = date( 'd.m.Y', $value->sec );
            }
        }
         if ( is_object( $value ) &&isset( $value->{'$id'} ) ) {
if(is_object( $data ) ) {
                $data->{$key} = $value->__toString();
            } else {
                $data[ $key ] = $value->__toString();
            }
        }
    }
return $data;
}
```

First off, note that we only parse the data for arrays and objects, and we're doing it recursively. This pre-parsing has to do with the fact that MongoDB uses dates and

IDs as objects, but our clients don't need this information. Now for the case of IDs, we just need its string value, hence the toString method call, then the value has an '$id'property. Afterwards we are converting the dates to a day.month.year format, this is being done for convenience in the design of the client application, again, not the most flexible approach but it works for this example.

**Handling Input**

Since we're sending JSON back to the client application, it is only logical that we accept data in JSON format as well. CodeIgniter doesn't support this by default likeLaravel does, as a matter of fact,CodeIgniter doesn't even support put and deleteparams. This is mainly because the framework is not intended for a RESTful service, however the effort that it takes to adapt it is minimal compared to the benefits, at least from my point of view.

So we will start by supporting the JSON data that BackboneJS sends. Create a new file inside the "core" folder, this time it is going to be named "MY_Input.php" and it will have the following basic structure:

Now every time we use $this->input in our application we'll be referring to this class, we will create some new methods and override a few existing ones. First off, we are going to add the support for JSON data, add the following method to the new class.

```
public function json() {

if ( !self::$request_params ) {

        $payload    = file_get_contents( 'php://input' );

if ( is_array( $payload ) ) {

self::$request_params   = $payload;
```

```
        } else if ( ( substr( $payload, 0, 1 ) == "{" ) && ( substr( $payload, ( strlen(
$payload ) - 1 ), 1 ) == "}" ) ) {
self::$request_params   = json_decode( $payload );
        } else {
parse_str( $payload, self::$request_params );
        }
    }
return (object) self::$request_params;
}
```

request_params is a static variable used to store the request string/data sent by the client. It is static in order to make it object independent so that we can access it from any controller at any given time. The data is obtained from the php://input stream rather than the $_POST global. This is done in order to obtain the data sent in via PUTand DELETE requests as well. Finally, the obtained payload is inspected to check if it's an array, a JSON encoded object, or a query string, and it's processed accordingly. The result is then returned as an object.

For this method to work, we need to create the static $request_params variable, add its declaration to the top of the class.

```
private static $request_params  = null;
```

Handling Regular Requests

Next, we need to override the post method of the regular input class to use the new JSON payload instead of the $_POST global, add the following method to the new Input class.

```
public function post( $index = NULL, $xss_clean = FALSE ) {
    $request_vars   = ( array ) $this->json();
if ( $index === NULL && !empty( $request_vars ) ) {
        $post       = array();
foreach(array_keys( $request_vars ) as $key ) {
            $post[$key]  = $this->_fetch_from_array( $request_vars, $key, $xss_clean );
        }
return $post;
    }
return $this->_fetch_from_array( $request_vars, $index, $xss_clean );
}
```

This is almost the same as the post method from the original CI_Input class, with the difference being that it uses our new JSON method instead of the $_POST global to retrieve the post data. Now let's do the same for the the PUT method.

```
          public function put( $index = NULL, $xss_clean = FALSE ) {
    $request_vars   = ( array ) $this->json();
if ( $index === NULL && !empty( $request_vars ) ) {
        $put = array();
foreach(array_keys( $request_vars ) as $key ) {
            $put[$key]    = $this->_fetch_from_array( $request_vars, $key, $xss_clean );
        }
return $put;    }
```

return $this->_fetch_from_array( $request_vars, $index, $xss_clean );

}And then we also need the DELETE method:

Now technically, there's really no need for these additional methods, since the postmethod can handle the params in the PUT and DELETE requests, but semantically it's better (in my opinion).

This is all we need for our custom Input class. Again we're ignoring edge cases here, like multipart requests, even though it's not very hard to handle those and still maintain the functionality obtained here, but, for the sake of simplicity we'll keep it just the way it is.

Base Model

To end the extension of the core classes, let's create a base model that every model in the app will extend upon, this is just to avoid repetition of common tasks for every model. Like any other core class extension, here's our barebones base model:

```
<?php

if( !defined( 'BASEPATH' ) ) exit( 'No direct script access allowed' );

classMY_Model extends CI_Model {

 }
```

This base model will only serve the purpose of setting and retrieving errors. Addthefollowingmethodtoset a modelerror:

```
        protected function _set_error( $desc, $data = null ) {

   $this->_error          = new stdClass();

   $this->_error->status   = 'error';

   $this->_error->desc     = $desc;
```

```
if ( isset( $data ) ) {

        $this->_error->data = $data;

    }

}
```

As you can see, this method uses an instance variable $error, so let's add its declaration to the top of our base model class.

Finally, to keep it structured, let's create the getter method for this property.

```
public function get_error() {

return $this->_error;

}
```

HandlingSessions

SessionController

For the last part of this tutorial, we will create the controller and model to handle user sessions.

The controller for our session is going to respond to any POST request made to our Session resource, since the session can't be retrieved after creation, nor updated directly, this controller will only respond to POST and DELETE requests. Please note that sending any other request to the resource will result in a server error, we're not dealing with edge cases here but this could be easily avoided by checking if the method that's called exists in our MY_Controller file and setting a default method name if the resource doesn't support the request.

Below you'll find the structure for our Session controller:

```
          <?php

if ( !defined( 'BASEPATH' ) ) exit( 'No direct script access allowed' );
```

```
class Session extends MY_Controller {

public function __construct() {}

public function save() {}

public function remove( $id = null ) {}

}
```

Note that this controller is extending the MY_Controller class instead of the regularCI_Controller class, we do this in order to use the _remap method and other functionality that we created earlier. OK, so now let's start with the constructor.

```
        public function __construct() {
   parent::__construct();
    $this->load->model( 'session_model', 'model' );
}
```

This simple constructor just calls its parent constructor (as every controller in CodeIgniter must do) and then loads the controller's model. The code for the savemethod is as follows.

```
public function save() {
   $result = $this->model->create();
if ( !$result ) {
      $result = $this->model->get_error();
   }
   $this->_format_output( $result );
}
```

And then the code for the remove method:

```
        public function remove( $id = null ) {
```

```
   $result = $this->model->destroy( $id );
if ( !$result ) {
       $result = $this->model->get_error();
   }
   $this->_format_output( $result );
}
```

Both methods simply delegate the task at hand to the model, which handles the actual data manipulation. In a real world application, the necessary data validation and session checking would be done in the controller, and the common tasks such as session checking should be implemented in the base controller.

Advertisement

Session Model

Now let's move on to the session model. Here is its basic structure:

```
          <?php
if ( !defined( 'BASEPATH' ) ) exit( 'No direct script access allowed' );
classSession_Model extends MY_Model {
public function __construct() {}
public function create() {}
publicfunctiondestroy( $id ) {}
}
```

Like the controller, this model extends the MY_Model class instead of the regularCI_Model class, this is being done in order to use the common methods that we've created earlier. Again, let's start with the constructor.

```
public function __construct() {
   $this->load->driver( 'mongo_db' );
}
```

In this case, we just load the Mongo_dbdriver, that we discussed earlier. Now we'll continue with the method in charge of destroying the session.

In this method we check if there's a session for the given session_id, and if so we attempt to remove it, sending a success message if everything goes OK, or setting an error and returning false if something goes wrong. Note that when using the session_id we use the special method $this->mongo_db->gen_id, this is because like I mentioned earlier, IDs in MongoDB are objects, so we use the id string to create it.

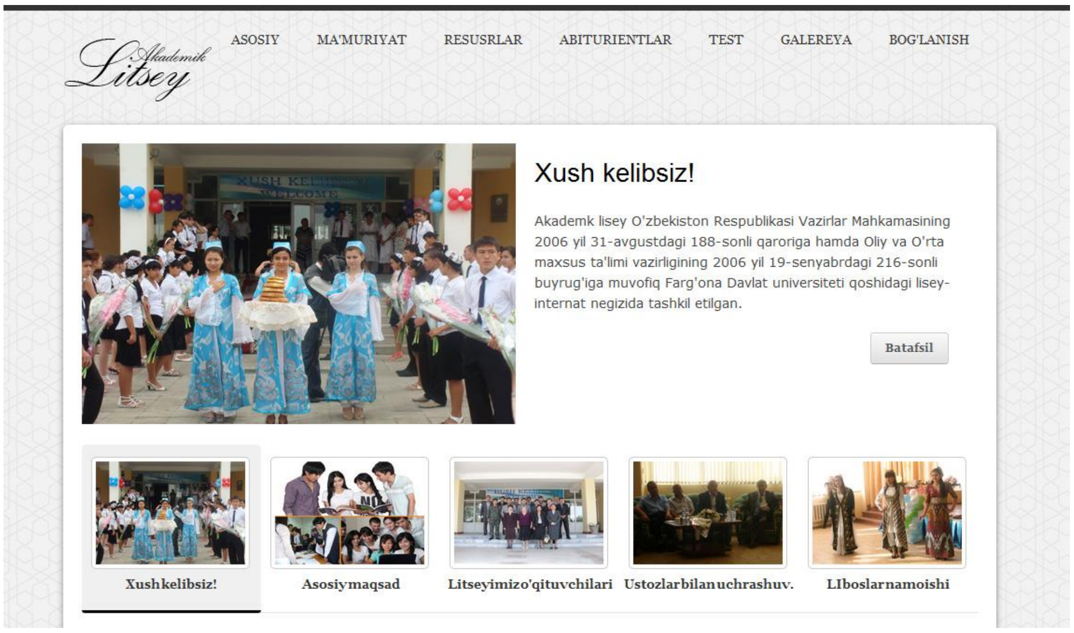
Finally, let's write the create method which will wrap up part one of this tutorial series.

```
        public function create() {
   $query          = $this->mongo_db->get_where( 'users', array( 'email' => $this->input->post( 'email' ) ) );
if ( $query->num_rows() != 1 ) {
      $this->_set_error( 'INVALID_CREDENTIALS' );
return false;
   }
    $this->load->library( 'encrypt' );
   $user   = $query->row();
   $salt   = $this->encrypt->decode( $user->salt );
if ( $user->pass != sha1( $this->input->post( 'pass' ) . $salt ) ) {
        $this->_set_error( 'INVALID_CREDENTIALS' );
return false;
    }
    $this->mongo_db->remove( 'sessions', array( 'user_id' => $user->_id->__toString() ) );
     $session    = array(
        'timestamp'     => now(),
        'user_id'       => $user->_id->__toString(),
        'persistent'    => $this->input->post( 'persistent' )
    );    if ( !$this->mongo_db->insert( 'sessions', $session ) ) {
        $this->_set_error( 'ERROR_REGISTERING_SESSION' );
return false;
    }
     $result                = new stdClass();
    $result->id            = $this->mongo_db->insert_id();
    $result->user_id       = $user->_id->__toString();
return $result;
}
```

First of all, we check that there's a user associated with the given email. Then we decode the user's associated salt (which I'll explain in the second part of this series when we cover user registration) and check that the given password matches the user's stored password.

We then remove any previous session associated with the user and create a new session object. If we were checking the session thoroughly, we would add things like the user agent, ip_address, and last activity field and so on to this object. Finally, we send back to the client the session and user IDs for the new session.

# III ANALYSE AND INTEGRATION OF THE RESULTS

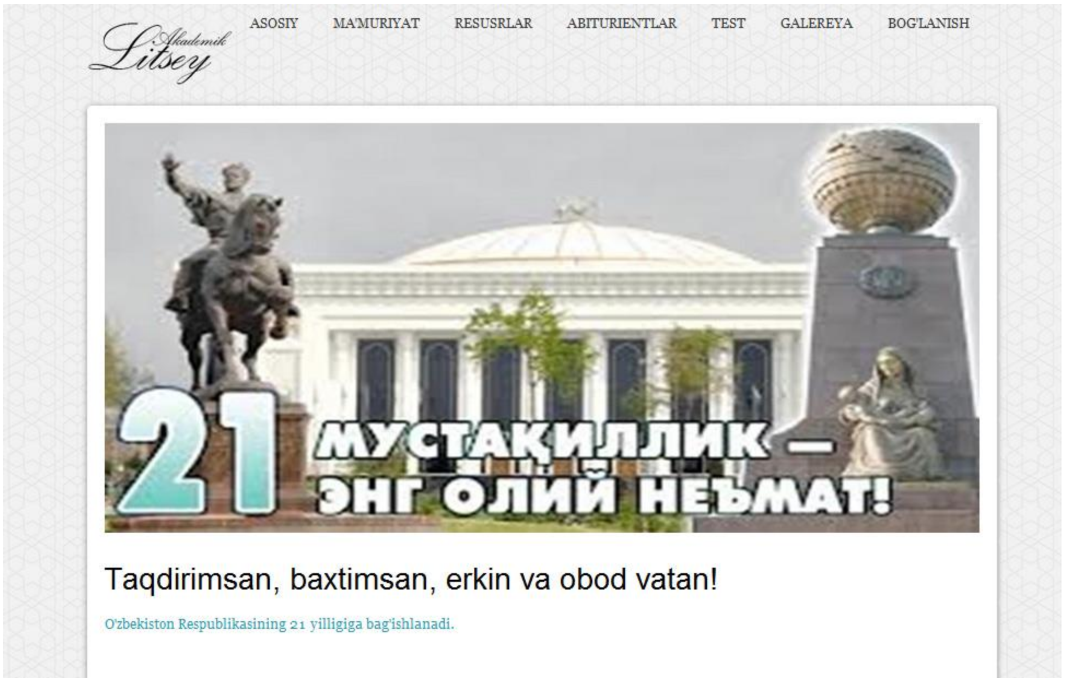


Xush kelibsiz!

Akademk lisey O'zbekiston Respublikasi Vazirlar Mahkamasining 2006 yil 31-avgustdagi 188-sonli qaroriga hamda Oliy va O'rta maxsus ta'limi vazirligining 2006 yil 19-senyabrdagi 216-sonli buyrug'iga muvofiq Farg'ona Davlat universiteti qoshidagi lisey-internat negizida tashkil etilgan.

3.1

The first page of web site – introduction to site and news. The short review of structure of lyceum and some information about education system are given on the page. The page is updated weekly.

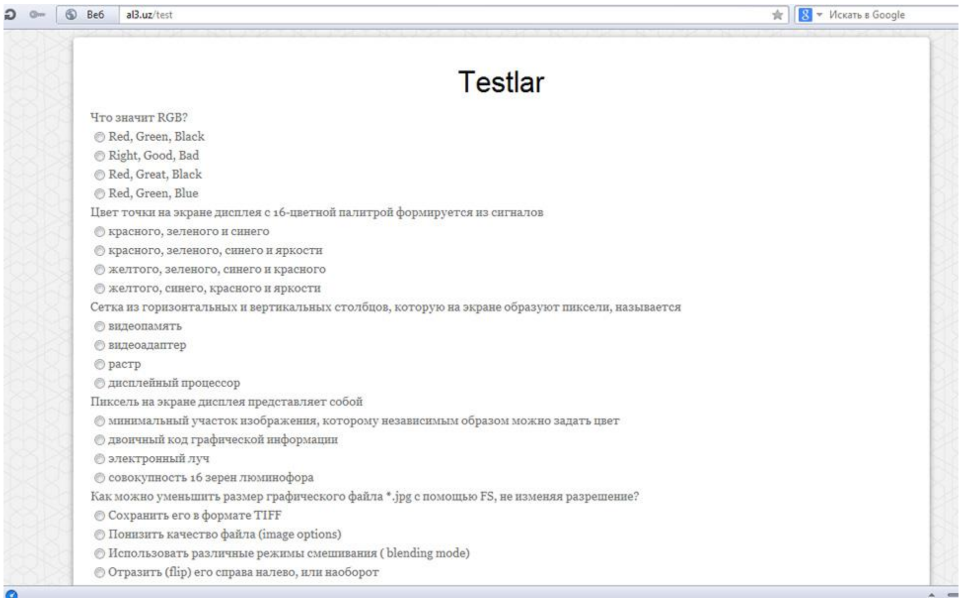


Taqdirimsan, baxtimsan, erkin va obod vatan!

O'zbekiston Respublikasining 21 yilligiga bag'ishlanadi.

3.2

On section recourses  the electronic library is located. There users can find many interesting books and textbooks. They can upload the individual and laboratory works, methodical grants, presentations and a lot of other useful materials and accordingly to download the necessary information.



Testlar

Что значит RGB?
- ◯ Red, Green, Black
- ◯ Right, Good, Bad
- ◯ Red, Great, Black
- ◯ Red, Green, Blue

Цвет точки на экране дисплея с 16-цветной палитрой формируется из сигналов
- ◯ красного, зеленого и синего
- ◯ красного, зеленого, синего и яркости
- ◯ желтого, зеленого, синего и красного
- ◯ желтого, синего, красного и яркости

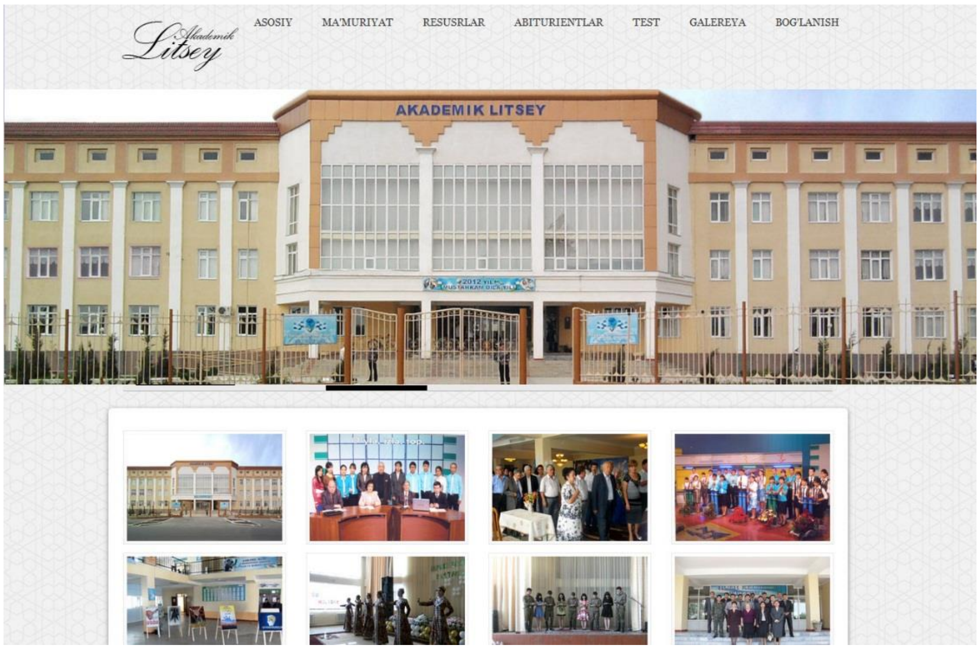
Сетка из горизонтальных и вертикальных столбцов, которую на экране образуют пиксели, называется
- ◯ видеопамять
- ◯ видеоадаптер
- ◯ растр
- ◯ дисплейный процессор

Пиксель на экране дисплея представляет собой
- ◯ минимальный участок изображения, которому независимым образом можно задать цвет
- ◯ двоичный код графической информации
- ◯ электронный луч
- ◯ совокупность 16 зерен люминофора

Как можно уменьшить размер графического файла *.jpg с помощью FS, не изменяя разрешение?
- ◯ Сохранить его в формате TIFF
- ◯ Понизить качество файла (image options)
- ◯ Использовать различные режимы смешивания ( blending mode)
- ◯ Отразить (flip) его справа налево, или наоборот

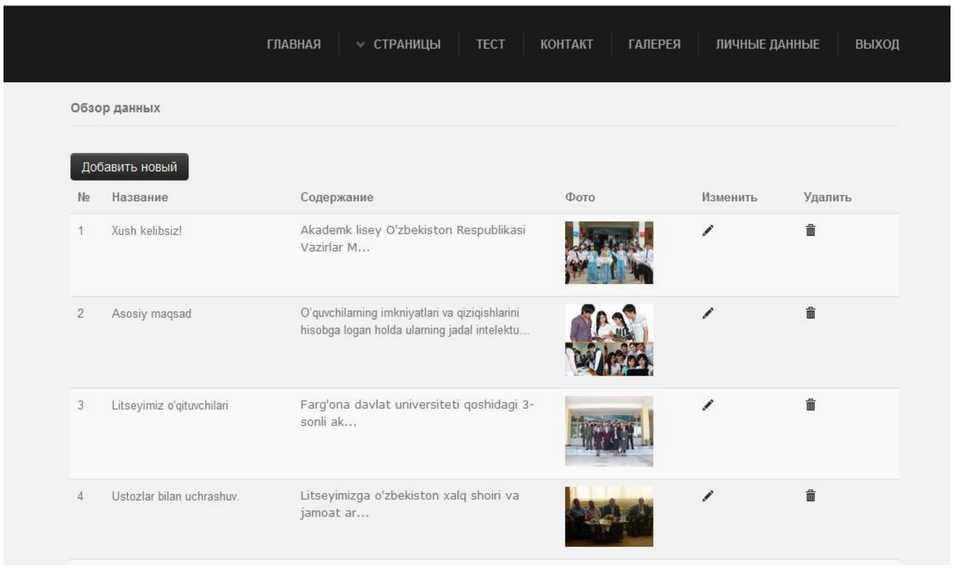
3.3

For entrants specially developed online testing is given. There they can check knowledge anytime.



3.4

There is a photo-gallery available andcontaining memoirs from the lyceum life.



Обзор данных

| № | Название | Содержание | Фото | Изменить | Удалить |
|---|---|---|---|---|---|
| 1 | Xush kelibsiz! | Akademk lisey O'zbekiston Respublikasi Vazirlar M... | | ✎ | 🗑 |
| 2 | Asosiy maqsad | O'quvchilarning imkniyatlari va qiziqishlarini hisobga logan holda ularning jadal intelektu... | | ✎ | 🗑 |
| 3 | Litseyimiz o'qituvchilari | Farg'ona davlat universiteti qoshidagi 3-sonli ak... | | ✎ | 🗑 |
| 4 | Ustozlar bilan uchrashuv. | Litseyimizga o'zbekiston xalq shoiri va jamoat ar... | | ✎ | 🗑 |

3.5

The administration part is quite comfortable and easy to use.

# IV LABOUR PROTECTION AND SAFETY TECHNIQUE

## 4.1 Ministry of Labor and Social Security of the Republic of Uzbekistan

1. Ministry of Labor and Social Security of the Republic of Uzbekistan has been established by the Decree of the President of the Republic of Uzbekistan № UP-2810 dated February 13, 2001 and the Resolution of the Cabinet of Ministers "On organization of activities of the Ministry of Labor and Social Security of the Republic of Uzbekistan" № 75 dated 13.02.2001

2. The Regulations of the Ministry of Labor and Social Security of the Republic of Uzbekistan have been approved by the Resolution of the Cabinet of Ministers "On approval of the Regulations of the Ministry of Labor and Social Security of the Republic of Uzbekistan" № 162 dated 06.04.2001

3. Organizational structure of the Ministry of Labor and Social Security of the Republic of Uzbekistan has been approved by the Resolution of the Cabinet of Ministers "On organization of activities of the Ministry of Labor and Social Security of the Republic of Uzbekistan" № 75 dated 13.02.2001

**Main functions and rights**

Ministry of Labor and Social Security of the Republic of Uzbekistan (Hereinafter - Ministry) is a central government administration body in the area of employment, labor, social protection and social maintenance of population and in its activities reports to the Cabinet of Ministers.

**Main tasks of the ministry are as follows:**

- Forming and realization of a single social policy, dedicated for maintaining stable and progressive improvement of material welfare of the people, strengthening social protection of population in conditions of further liberization of the economy;
- Working out and control over accomplishment of demographic development forecasts, carrying out effective policies for forming the labor market and maintaining full employment, especially among young generation, working out

target programs for creating new labor vacancies and staff training on the basis of labor resources balances;

- Drawing up measures for regulating labor relationships in accordance with the market conditions; improving conditions, systems for organization safety and norms of the work;
- Implementation of the activities for maintenance of functioning social security for pensioners, incapable people, families with many children and poor families, development and realization of effective techniques of targeted social protection of socially vulnerable part of the population;
- Organization and administration of social guarantees, supervision of activities of medic-social expertise and rehabilitation of disabled services, organization of the prosthesis orthopedic assistance to the population.

**In accordance with assigned tasks the Ministry:**

- With the Ministry of Economy undertakes forecasts of demographic development of the republic and regions, works out estimated balances of labor resources with grounded combination of actions for maintaining effective employment of the population;
- Analyses migration processes, develops proposals for organizing external labor migration and labor activities for foreign citizens in the territory of the Republic of Uzbekistan;
- Organizes registration of unemployed citizens;
- Organization and administration of social guarantees, supervision of activities of medic-social expertise and rehabilitation of disabled services, organization of the prosthesis orthopedic assistance to the population.
- Organizes social surveys for discovering professional preferences and work orientation of youth and unemployed population;
- Together with the ministers and agencies concerned, local government bodies participates in programs development for creating new labor vacancies and coordinates their implementation;

- Develops proposals for the stimulation and assistance to self-employment of population;
- Maintains improvement of the infrastructure of labor market functioning;
- Maintains organization of paid public services for unemployed;
- Organizes professional training and education for unemployed;
- Participates in preparation of proposals for improvement of methods in social policies implementation, aimed to the strengthening efficiency and effectiveness of social assistance to pensioners, incapable, lonely aged;
- Participates in the development of the drafts legislative acts on realization of government social policy and social security system functioning;
- Organizes maintenance of social assistance to unemployed people and members of their families at the expenses of the Government fund for employment assistance;
- Organizes work for provision of fulfillment compulsory duties and develops suggestions for improvement of wages mechanism, single tariffs, regional regulation of salary;
- Gives conclusions on suggestions of ministers and agencies concerning changes in lists of productions, works and professions, positions and indicators for privileged pension funding, additional vacation, part-time working day;
- Prepares proposals to the Cabinet of Ministers on introduction of changes in current privileges and compensations lists for work in unfavorable conditions;
- Works together with the taxation agencies of the Republic of Uzbekistan on issues of guaranteeing fulfillment compulsory duties and payments under the social insurance to the non-budget Pension fund of the Republic of Uzbekistan, timely financing of social payments stipulated by legislation;
- Maintains financing of expenses for payment of pensions and grants for social insurance from the funds of the non-budget Pension fund in amount and order set by the legislation;
- Controls over target use of funds of the non-budget Pension fund;

- Carries out and organizes control-auditing work by social security departments in checking compliance with target and payment of pensions and grants for social insurance;
- Carries out organization and methodological administration of the work on prescription and payment of pensions and grants.

## 4.2. Grounding facility

Grounding is arranged in accordance with the requirements of SAE, Snip-W-33-76 and instructions for grounding device networks and vanishing in electrical (CH 102-76).

Grounding should be performed:

a) At voltages of 380 V AC and DC and up DC 440 V and above in all installations;

b) When the AC voltage above 42 V DC and 110 V above only in electrical placed in areas with high risk or high-risk, as well as in outdoor settings;

c) At any voltage and AC and DC current into explosive devices;

Grounding can be used by both natural and artificial. Moreover, if the natural grounding has spreading resistance meeting the requirements of the SAE, then the device does not require artificial grounding.

a) Laid in the ground water and other metal piping, except piping of flammable and combustible liquids, flammable or explosive gases and mixtures;

b) Casing, metal and concrete structures of buildings and structures, which are in direct contact with the ground;

c) Lead sheath cables laid in the ground, etc.

As artificial grounding most commonly used angular steel 60x60 mm diameter steel pipes with 35-60 mm tires and steel section not less than 100 mm$^2$.

Rods with 2.5 ... 3m plunge (hammered) into the ground vertically in a specially prepared trench (Fig. 1).

$$U_{\ddot{\imath}\eth} = U_{ç}$$

Where $\varphi_{\hat{\imath}\tilde{n}\acute{\imath}}$ – potential of the ground.

Tanker Grounding protects only due to the low resistance of the soil.

Vertical grounders are connected with the steel strip, which is welded to each earth electrode.

By location relative to grounding relatively grounding equipment the system is divided into Tanker and contour.

Tanker grounding equipment is shown in Figure 2. At external grounding system grounding located at some distance from the equipment grounding. Therefore grounded equipment is beyond the current spreading and a person touching it will be under full voltage respectively to ground

Ground contour is shown in Fig. 3. Grounding switches are arranged along the contour in small grounding equipment (several meters) away from each other. In this case, the field spreading grounding overlap, and any point within the contour of the earth's surface has significant potential. Contact voltage will be less than when mounted grounding.

## 4.3. Normalization the parameters of protective grounding

Protective grounding intended for human security when touching conductive equipment accidentally arising, and when exposed to a voltage step. These values should not exceed the long-term allowable

$$U_{ïð} \le U_{ïð.ä.ä.}$$

$$U_{ø} \le U_{ø.ä.ä.}$$

In the SAE standardized grounding resistance versus voltage electrical installations.

In installations up to 1000 V resistance grounding device should be no higher than 4 ohms; if the total power sources does not exceed 100 kVA, grounding resistance should be less than 10 ohms.

In 1000 electrical circuit with a current of 500 A is allowed grounding resistance but not more than 10 ohms.

## 4.4. Calculation of grounding

Calculation grounding reduces to determining the number and length of the connecting grounding strip based on limit resistance.

Source data:

| Type of grounding | Remote |
|---|---|
| Length of grounding l, m | 2,7 |
| Depth of the earth electrode in the soil h, | 0,65 |
| Seasonality factor Ks | 2,0 |
| Soil resistivity ρ, Ohm · m | 70 |
| Grounding diameter d, m | 55 |
| The width of the connecting strip b, m | 50 |
| Permissible resistance grounding system PUE RE.N. 4 ohm | 4 |

1. As a grounding, we chose as steel pipe with a diameter $d = 55$ *ìì* and as a coupling element - the steel strip with the width $b = 50$ *ìì* .

2. Choose the resistivity of the soil or nearly corresponding resistivity values of soil in a given area of designed installation.

3. Determine the value of the electrical resistance of current spreading in the ground with a single grounding

$$R_{ç} = 0{,}366\,\frac{\rho \cdot K_{c}}{l}\left(\lg\frac{2 \cdot l}{d} + \frac{1}{2}\lg\frac{4t + l}{4t - l}\right) =$$

$$= 0{,}366\,\frac{70 \cdot 2}{2{,}7}\left(\lg\frac{2 \cdot 2{,}7}{0{,}055} + \frac{1}{2}\lg\frac{4 \cdot 2 + 2{,}7}{4 \cdot 2 - 2{,}7}\right) = 40{,}62\ \hat{I}\grave{\imath}\ .$$

Where $\rho = 70\ Ом$ - resistivity of the soil,

$\hat{E}_{ñ} = 2{,}0$ - Factor of seasonality,

$l = 2{,}7\ \grave{\imath}$ - The length of grounding

$d = 55\ ìì$ - Diameter grounding

$t = h + 0{,}5l = 0{,}65 + 0{,}5 \cdot 2{,}7 = 2\ \grave{\imath}$ - Distance from the ground surface to the middle of the grounding.

4. We calculate the number of groundings without interference, provided grounding at each other, the so-called phenomenon of mutual "screening"

$$n^{/} = \frac{R_{ç}}{R_{çí}} = \frac{40{,}62}{4} = 10{,}15 \approx 10.$$

5. We calculate the number of groundings by considering the shielding factor

6. $n = \dfrac{n^{/}}{\eta_{ç}} = \dfrac{10}{0{,}58} = 17{,}24$

Where $\eta_{ç} = 0{,}58$ - the shielding factor (Annex, Table 1.).

Accept the distance between grounding $a = l = 2{,}7\ \grave{\imath}$

7. Determine the length of the connecting strip

$$l_{Ï} = 1{,}05 \cdot n \cdot a = 1{,}05 \cdot 18 \cdot 2{,}7 = 51{,}03\ ì\ .$$

8. Calculate the total resistance value of the current spreading with connecting strip

$$R_{Ï} = 0{,}366 \frac{\rho \cdot Ê_{ñ}}{l_{Ï}} \lg \frac{2 \cdot l_{Ï}^{2}}{b \cdot h} = 0{,}366 \frac{70 \cdot 2}{51{,}09} \lg \frac{2 \cdot 51{,}03^{2}}{0{,}05 \cdot 0{,}65} = 5{,}2\ Îì\ .$$

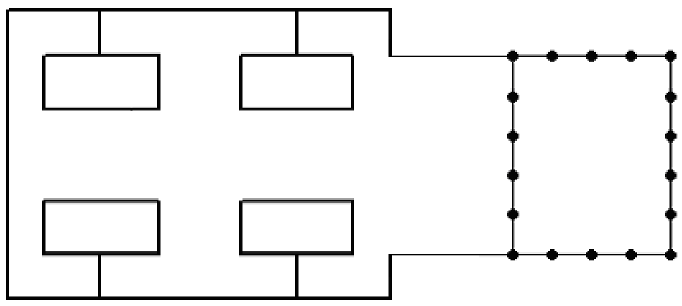
9. Calculate the total value of the resistance of the grounding system

$$R_{çó} = \frac{R_{ç} \cdot R_{ï}}{R_{ç} \cdot \eta_{ï} + R_{ï} \cdot \eta_{ç} \cdot n} = \frac{40{,}62 \cdot 5{,}2}{40{,}62 \cdot 0{,}51 + 5{,}2 \cdot 0{,}58 \cdot 18} = 2{,}82\ Îì\ .$$

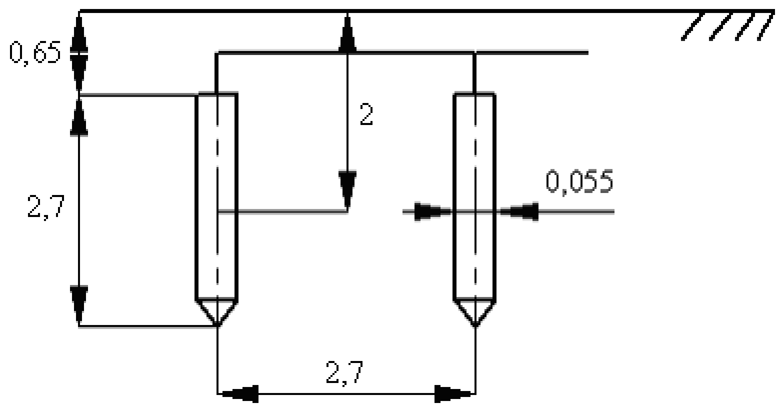
Where = 0.51 - strip the shielding factor (Annex, Table 2.).

Conclusion to Chapter 4

Resistance *Rzu* = 2.82 ohms less allowable resistance equal 4 ohms. Consequently, the diameter d = grounding 55 mm including grounding at n = 18 is sufficient to provide protection in a remote location grounding circuit.



4.4.4 The scheme received external grounding.



4.4.5. Scheme of groundings location

# CONCLUSION

Nowadays the internet is getting more and more important for contemporary people. It has changed the face of the lives of people turning them completely into the modern and latest lifestyle with its developments. Today instead of the newspapers people use the internet to access the e-news which provides with not only the newspapers completely but also various different news channels from all over the world.

In the world of modern computer technologies we gain all the information via internet. That's why developing sites is increasing enormously. The term web standards is now more widely used by web professionals that any other term applied by the W3C (in fact the W3C have started to use the term on their own pages), so that is what we are going to teach you — the *standards* way to build web sites

In this work we discussed issues related to the creation of web site for the Lyceum. This web site gives an opportunity for everybody to gain the information about this lyceum. It contains e-library. On-line tests and news blogs. All students of this lyceum can gain the information and to check their knowledge in on-line format. They also may find each other on the site and have a chat.

You could see and understand the process of creating the lyceum web-site. This site is developed with the help of FrameworkCodeIgniter and technology MVC.

## LIST OF CITED BIBLIOGRAPHIES

1. “Uzbekistan on the Threshold of the Twenty-first Century” Islam Karimov 1995
2. “Designing for the Web” by Mark Boulton. Sidney 2012
3. “Handcrafted CSS: More Bulletproof Web Design” by Dan Cederholm and Ethan Marcotte, published by Amazon.
4. “Offline address book of programmers (C++, PHP, Java & etc.)” by Neil Hobson. 2008. London.
5. “HTML and CSS: Design and Build Websites” by Jon Duckett, published by Amazon.
6. “Creating and Configuring Custom book with Web-Based Distribution in Exchange 2010 using PowerShell” by Krishna Kumar. 2012.
7. “PHP Master, Write Cutting-Edge Code” by Lorna Mitchell, Davey Shafik& Matthew Turland publisher SitePoint.
8. “CSS is code” by Nat Torkington.
9. “Ajax: The Complete Reference” by Thomas Powell.
10. “The Power of HTML” by Simon St. Laurent, 2013.
11. “PHP 6 and MySQL 5 for Dynamic Web Sites: Visual Quick Guide” by Larry Ullman.
12. “PHP Objects, Patterns and Practice” by Matt Zandstra published by Apress.
13. “Tailoring CSS for performance” by Roseanne Fallin, 2012.
14. “PHP 5 Power Programming” published by www.tutorial.com.
15. “Introducing HTML5” by Bruce Lawson and Remy Sharp, published by Amazon.
16. “HTML as foundation” by Simon St. Laurent, 2014.
17. Web site htmlbook.ru, address: http://www. htmlbook.ru/ .
18. “PHP Advanced Object Oriented Programming” by Larry Ullman.