UTF-8 IN PHP

HOW PHP SUPPORTS UTF-8

Chunki Jun @ Modern PUG

UTF-8 사용하기

이 섹션은 Alex Cabal이 작성한 PHP Best Practices에 포함되어 있는 내용을 기반으로 하여 작성되었습니다.

한 줄로 끝나는 팁은 없습니다. 주의깊고 일관성있게 작업해야 합니다.

PHP는 현재 유니코드 지원을 언어 내부적으로 해주고 있지 않습니다. UTF-8 문자열을 문제없이 처리할 수 있는 방법은 있지만, HTML 코드, SQL 처리, PHP 코드 등 웹 어플리케이션의 모든 레벨을 다뤄야 하는 수준의 이야기입니다. 여기서는 활용할 수 있는 간략한 설명을 제공하는데 초점을 맞추겠습니다.

PHP 코드 수준에서의 UTF-8

WHY?

PHP: The Right Way

문자열을 변수에 할당하거나 두 문자열을 이어붙이는 등의 기본적인 UTF-8 문자열이라고 해도 특별할 것이 없습니다. 하지만 strpos() 나 strlen() 과 같은 대부분의 문자열 함수들은 mb_strpos() 나 mb_strlen() 같이 앞에 mb_ 가 붙은 함수들이 별도로 존재합니다. 그런 함수들을 멀티바이트 문자열 함수라고 하고, 멀티바이트 문자열 익스텐션에 포함되어 있습니다. 멀티바이트 문자열 함수들은 유니코드 문자열을 다룰 때에 사용해야 합니다.

예를 들어 mb_substr() 함수를 사용하지 않고 substr() 함수를 사용하면, 결과물에는 이상하게 깨진 글자가 나온다는 사실을 알게 될 좋은 기회가 될 것입니다. UTF-8 문자열을 다룰 때에는 mb_substr() 함수를 사용해야 합니다.

유니코드 문자열을 다룰 때에는 항상 mb_ 로 시작하는 함수를 사용해야 한다는 걸 기억하는 게 어려운 일입니다. 한 군데라도 까먹고 일반 문자열 함수를 사용하면 깨진 유니코드 문자열을 보게 될 수가 있습니다.

모든 문자열 함수가 그에 해당되는 멀티바이트 버전의 mb_ 로 시작되는 함수를 가지고 있는 것은 아닙니다. 여러분이 사용하려고 했던 문자열 함수가 그런 경우라면, 운이 없었다고 할 수 밖에요.

모든 PHP 스크립트 파일(또는 모든 PHP 스크립트에 include 되는 공용 스크립트)의 가장 윗부분에 mb_internal_encoding() 함수를 사용해 내부 문자열 인코딩을 지정하고, 그 바로 다음에 mb_http_output() 함수로 브라우저에 출력될 문자열 인코딩도 지정해야 합니다. 이런 식으로 모

PHP와 UTF-8

한 줄로 끝나는 팁은 없습니다. 조심히 세세하고 일관성있게 작업하세요.

미안하지만, PHP에서 UTF-8은 형편없습니다.

현재 PHP는 내부적으로[low level] 유니코드를 지원하지 않습니다. UTF-8 문자열을 제대로 처리하도록 강제하는 방법은 있지만, 쉽지 않고, HTML부터 SQL과 PHP까지 웹 애플리케이션의 거의 모든 부분을 파헤쳐야 합니다. 간략하게 실용적인 내용을 정리해보겠습니다.

WHY?

PHP Best Practices

PHP에서의 UTF-8

두 문자열을 연결해서 변수에 문자열을 할당하는 것과 같은 기본 문자열 연산은 UTF-8을 위해서 아무 것도 필요없습니다. 하지만 대부분의 문자열 함수는 특별히 주의해야 합니다. 이런 함수들은 보통 mb_*로 시작하는 대응 함수가 있습니다. 예를 들자면, mb_strpos()나 mb_strlen()이 있습니다. 이렇게 대응하는 함수들을 묶어서 멀티바이트 문자열 함수[Multibyte String Functions]라고 부릅니다. 멀티 바이트 문자열 함수들은 유니코드 문자열에 동작하도록 만들어졌습니다.

유니코드 문자열을 다룰때에는 *반드시* mb_* 함수를 사용하세요. UTF-8 문자열에 substr()을 사용하면, 알아볼 수 없는 반쪽짜리 문자를 포함한 결과를 얻게될 가능성이 높습니다. 이럴때 사용해야할 함수는 대응하는 멀티바이트 문자열 함수인 mb_substr()입니다.

어려운 점은, *항상* mb_* 함수를 사용해야 한다는 것을 기억해야한다는 것입니다. 단 한번이라도 잊게되면, 여러분의 유니코드 문자열은 처리과정에서 알아볼 수 없게 되어버릴 것입니다.

UTF-8 @ PHP BEST PRACTICES

1. PHP

   *mb_**

2. Database(MySQL)

   utf8mb4

3. Browser

   mb_http_output(),
   ...

1. PHP

MULTIBYTE STRING FUNCTIONS

substr() ➨ mb_substr()

strpos() ➨ mb_strpos()

strlen() ➨ mb_strlen()

But, *Not all of them!*

http://php.net/ref.mbstring

1. PHP

SET ENCODING AS *UTF-8* EXPLICITLY

at script

```
mb_internal_encoding("UTF-8");
```

But, *mb_internal_encoding()* uses PHP settings:

- default_charset
- ~~mbstring.internal_encoding~~ - deprecated (PHP 5.6)

1. PHP

SET ENCODING AS UTF-8 EXPLICITLY

at functions

```
// example at http://php.net/function.htmlentities
echo htmlentities($str, ENT_QUOTES | ENT_IGNORE, "UTF-8");
```

But, functions use **default_charset**

```
string htmlentities ( string $string
        [, int $flags = ENT_COMPAT | ENT_HTML401
        [, string $encoding = ini_get("default_charset")
        [, bool $double_encode = true ]]] )
```

2. DATABASE (MYSQL)

Use **utf8mb4** (MySQL >= 5.5.3)

*the utf8mb4 character set uses a maximum of **four bytes per character** supports supplemental characters*

SHOULD I USE UTF8MB4?

Plane	Detail	Result
BMP	Basic Multilingual Plane	utf8 == utf8mb4
SMP	Supplementary Multilingual Plane	utf8 != utf8mb4

BMP

A ぁ 入 갊

SMP

http://en.wikipedia.org/wiki/Plane_(Unicode)

BROWSER

- mb_http_output()

```
mb_http_output('UTF-8');
```

  No need if file saved w/ UTF-8

- Use charset

```
<meta charset="utf-8">
```

CONCLUSION

- check php.ini - default_charset, ~~mbstring.internal_encoding~~
- Use utf8mb4 if you use
- Save files w/ UTF-8

PHP와 UTF-8

한 줄로 끝나는 팁은 없습니다. 조심히 세세하고 일관성있게 작업하세요.

미안하지만, PHP에서 UTF-8은 형편없습니다.

현재 PHP는 내부적으로[low level] 유니코드를 지원하지 않습니다. UTF-8 문자열을 제대로 처리하도록 강제하는 방법은 있지만, 쉽지 않고, HTML부터 SQL과 PHP까지 웹 애플리케이션의

CONCLUSION

UTF-8 in PHP ~~sucks~~ is just fine.

예를 들자면, mb_strpos()나 mb_strlen()이 있습니다. 이렇게 대응하는 함수들을 묶어서 멀티바이트 문자열 함수[Multibyte String Functions]라고 부릅니다. 멀티 바이트 문자열 함수들은 유니코드 문자열에 동작하도록 만들어졌습니다.

유니코드 문자열을 다룰때에는 *반드시* mb_* 함수를 사용하세요. UTF-8 문자열에 substr()을 사용하면, 알아볼 수 없는 반쪽짜리 문자를 포함한 결과를 얻게될 가능성이 높습니다. 이럴때 사용해야할 함수는 대응하는 멀티바이트 문자열 함수인 mb_substr()입니다.

어려운 점은, *항상* mb_* 함수를 사용해야 한다는 것을 기억해야한다는 것입니다. 단 한번이라도 잊게되면, 여러분의 유니코드 문자열은 처리과정에서 알아볼 수 없게 되어버릴 것입니다.

RESOURCES

- PHP: The Right Way
- PHP Best Practices
- Is unicode support going to make it into PHP7?
- PHP RFC: UString

THANK YOU