PHP

مسیر اشتباه

آخرین بروزرسانی: 2017-10-08

خوش آمدید

در دنیای برنامه‌نویسی PHP مجموعه‌ای از رویه‌ها به صورت گسترده‌ای توسط افراد (در کتاب‌ها و وبسایت‌هایشان) به عنوان PHP مدرن معرفی می‌شوند به طوری که رویکردهای دیگر را قدیمی، احمقانه یا غلط جلوه می‌دهند.

این افراد تلاش زیادی انجام می‌دهند تا رویه کاری خود را به دیگران تحمیل کنند.

این وبسایت به منظور ارائه دیدگاهی عملگرا در برنامه‌نویسی PHP ایجاد شده است. دیدگاهی که مبتنی بر تجربه و عمل قبلی باشد تا اینکه بر اساس نظریه، رویه‌های محبوب یا مفاهیم دشوار دانشگاهی پیش رود.

وبسایت PHP - مسیر اشتباه یک سند زنده است و طی مرور زمان با اطلاعات بیشتری بروزرسانی خواهد شد.

در این پروژه مشارکت کنید.

ترجمه‌ها

- اسپانیایی
- انگلیسی
- پرتغالی
- دانمارکی
- روسی
- فارسی

خطر افراطگرایی

یکی از مشکلات قوانین و قواعد برنامه‌نویسی این است که آن‌ها تنها تحت شرایطی خاص معنا می‌یابند. اگر این شرایط را حذف کنیم، یک قانون خوب می‌تواند به یک قانون وحشتناک تبدیل شود. در حقیقت، هر قانون خوبی که استفاده از آن جنبه افراطی پیدا کند بد می‌شود.

درک این نکته مهم است که بسیاری از قوانین و قواعد توسعه نرم‌افزار که به مرور زمان و توسط تعداد زیادی از افراد ایجاد شده‌اند، اغلب توسط افراط‌گرایان به شیوه‌ای نادرست استفاده می‌شوند.

تجربه نشان داده است که استفاده نادرست از قوانین و قواعد همیشه منجر به پیچیدگی، کاهش امنیت، نتایج پرخطا و در بعضی موارد نیز به فاجعه می‌انجامد.

اصل KISS، که مخفف عبارت Keep It Simple, Stupid است، یک اصل خردمندانه و خوب است که اغلب توسط افراد باتجربه به عنوان راهکاری مناسب در نظر گرفته می‌شود اما همین اصل نیز در صورت به افراط کشیده شدن نیز می‌تواند پروژه را به خطر بیندازد. گاهی اوقات نیز عبارت بسیار ساده منجر به نبود عملکرد مورد نظر می‌شود.

مسیر اشتباه: پیروی کورکورانه از قواعد و قوانین.

همیشه از یک چارچوب نرم‌افزاری استفاده کنید

> تمام چارچوب‌های نرم‌افزاری PHP بیخودن!
>
> — راسموس لردورف

در جامعه‌کاربری PHP یک گرایش بد به استانداردی برای توسعه نرم‌افزارهای وب تبدیل شده و آن استفاده از یک چارچوب نرم‌افزاری عمومی و همه منظوره است.

این گرایش طی مرور زمان ظهور یافته و محبوب شده است نه به این دلیل که نتیجه نهایی فرآیند توسعه را بهبود می‌بخشد یا ابزار مناسبی از نقطه نظر فناوری یا معماری به حساب می‌آید. این گرایش محبوب شده است چرا که توسعه‌دهندگان چارچوب‌های نرم‌افزاری برای گمراه کردن دیگران، عبارت‌هایی نظیر "چرخ را دوباره اختراع نکنید!" و "خودتان اینکار و نکنید، دیگران از شما تواناتر هستند"، شرایطی را علیه سبک‌های مختلف برنامه‌نویسی ایجاد کرده‌اند

بسیاری از برنامه‌نویسان امروز اصول ابتدایی مهندسی صدا را در کل نادیده گرفته و زمان زیادی را بابت طراحی لایه‌های پیچیده به منظور هوشمندانه‌تر و مقبول‌تر شدن چارچوب‌های نرم‌افزاری خود برای آن‌ها که همکار خود می‌دانند، صرف می‌کنند.

این افراد به نظر می‌رسد با این تفکر که دیگران را به پیروی از "شیوه حل مساله ایشان" متقاعد سازند، فریب خورده‌اند، با این تصور که به نوعی رهبران جامعه‌کاربری PHP تبدیل شوند و دیگران را مجبور سازند که از ابزار اوپن سورس به اصطلاح "جامع" ایشان استفاده کنند، با این حال فراموش کرده‌اند که توصیه ایشان به این سبک از حل مساله پایداری لازم را ندارد.

در صنعت نرم‌افزار می‌توان یک چارچوب نرم‌افزاری همه منظوره را به یک خانه از پیش ساخته شده تشبیه کرد. همان طور که کنار هم قرار دادن اجزای یک خانه از شما نجار نمی‌سازد، ایجاد نرم‌افزار با استفاده از چارچوب‌های نرم‌افزاری همه منظوره شما را کدنویس یا برنامه‌نویس نمی‌کند.

در این وبسایت، بین چارچوب‌های نرم‌افزاری و کتابخانه‌ها به ترتیب زیر تفاوت قائل می‌شویم:

- یک کتابخانه مجموعه‌ای از کدهای قابل استفاده است، مانند کتابخانه استاندارد در زبان C یا Go. کتابخانه شامل کدی است که شما به راحتی در پروژه خود وارد می‌کنید بدون آنکه کوچکترین محدودیتی اعمال شود. کتابخانه شامل تکه‌های کوچکی از کد است که هر کدام عملکرد مشخصی دارند.
- یک چارچوب نرم‌افزاری تنها مجموعه‌ای از کدهای قابل استفاده نیست؛ شما به سادگی نمی‌توانید تکه‌ای از کد چارچوب نرم‌افزاری را برداشته و در پروژه خود قرار دهید. چارچوب نرم‌افزاری سیستمی است که به شما در ساخت نرم‌افزار کمک کرده، اما در این میان محدودیت‌های خود را نیز بر شما تحمیل می‌کند. چارچوب نرم‌افزاری، خود از عملکردهای جداگانه‌ای تشکیل شده است که هر یک از آن‌ها اغلب نمی‌توانند بدون کمک سایر عملکردها به کار خود ادامه دهند.

در دنیای پایتون و روبی، ساختن وبسایت‌ها از ابتدای مسیر کار بسیار دشواری است چرا که پایتون و روبی در ابتدا به منظور ساخت وبسایت طراحی نشده‌اند. به همین دلیل، چارچوب‌های نرم‌افزاری همه منظوره مانند Django یا Ruby on Rails به سرعت محبوبیت خود را برای ایجاد وبسایت به این زبان‌ها پیدا کرده‌اند.

از طرف دیگر، PHP از ابتدا توسط راسموس لردورف به عنوان مجموعه‌ای از ابزارهای نوشته شده به زبان C به منظور ایجاد صفحات پویای HTML طراحی شده است. به همین دلیل، PHP**به خودی خود یک چارچوب نرم‌افزاری** بوده و هست.

از آن روز تاکنون PHP تغییرات زیادی داشته است و امروزه از آن می‌توان برای ساخت چیزهایی بیش از صفحات HTML و وبسایت‌ها استفاده کرد، اما این دید نسبت به PHP که خود یک چارچوب نرم‌افزاری به حساب می‌آید اشتباه نیست. PHP در عمل یک لایه انتزاعی برای توسعه نرم‌افزارهای وب است که کاملا به سبک برنامه‌نویسی رویه‌ای در زبان C نوشته شده‌اند.

استفاده از یک کتابخانه در پروژه، عملی طبیعی است. PHP به همراه کتابخانه‌هایی ارائه می‌شود که از آن‌ها می‌توان در کد خود استفاده کنید. برای نمونه، PDO یک کتابخانه سبک به منظور دسترسی جامع و یکپارچه به پایگاه‌داده‌ها در PHP است.

استفاده از یک چارچوب نرم‌افزاری در PHP خود مساله دیگری است.

زمانی که از یک چارچوب نرم‌افزاری در PHP استفاده می‌کنید در عمل یک لایه انتزاعی را روی لایه انتزاعی دیگری قرار داده، لایه‌ای که از ابتدا برای شما وجود داشته است. لایه انتزاعی اضافه شده که چارچوب نرم‌افزاری آن را فراهم کرده است ممکن است به منظور سازمان‌دهی به کد شما با الگوهایی ثابت عمل کرده یا ممکن است با تعامل بین صدها یا هزاران کلاس و متد به پیچیدگی کار بیفزاید و کابوسی از وابستگی‌ها را ایجاد نماید، در هر صورت، شما لایه‌های پیچیده‌ای را به کد خود اضافه می‌کنید که واقعا مورد نیاز نیستند!

تمام تجربه با رابط شروع می‌شود. تجربه کار با رابط چارچوب نرم‌افزاری نتیجه فناوری درون آن و حجم لایه‌های انتزاعی است. هر چه از لایه‌های انتزاعی بیشتری استفاده کنید، این رابط کارکرد خود را از دست می‌دهد و برنامه با خطاهای بیشتری مواجه خواهد شد. هر چه سطح انتزاعی بیشتر باشد، جزئیات و عملکرد بیشتری از بین می‌رود.

این نکته را خوب درک کنید**تعداد خطوط ایده‌آل کد در یک پروژه به اندازه‌ای کم باشد که تا جای ممکن شفاف و قابل خواندن باقی بماند!**

> چیزی که هیچکس به آن احتیاج ندارد یک چارچوب نرم‌افزاری همه منظوره است. هیچکس مشکل عمومی ندارد، هر کس بنا به نیاز خود مشکل خود را حل می‌کند.

> — راسموس لردورف

برخی شرکت‌ها به شایعات موجود درباره چارچوب‌های نرم‌افزاری PHP گوش کرده و پروژه بعدی خود را با یکی از این ابزارهای محبوب آغاز می‌کنند تنها به این منظور که به فاجعه ختم شود.طی گذشت زمان، نه تنها در می‌یابند که استفاده از چارچوب‌های نرم‌افزاری همه منظوره نیاز ایشان را رفع نمی‌کند، بلکه در حل این نیاز نیز عملکرد کندی دارد. مقیاس‌پذیری در این حالت غیرممکن می‌شود و به همین دلیل در یک تلاش ناامیدانه تصمیم به تجزیه اجزای چارچوب نرم‌افزاری می‌گیرند که به آن نیاز ندارند.

همیشه از رویکردی عملگرا استفاده کنید:

> اقدام یا عملی که با در نظر گرفتن عواقب عملی فوری در مقایسه با نظریه و عقیده دیکته می‌شود.
>
> — لغتنامه انگلیسی کالینز، ویرایش ۱۲، سال ۲۰۱۴

مسیر اشتباه: همیشه از یک چارچوب نرم‌افزاری در PHP استفاده کنید. 

همیشه از یک الگوی طراحی استفاده کنید

> من این آلرژی بزرگ به الگوهای طراحی رو دارم. پیتر نورویگ، زمانی که در Harlequin بود، مقاله‌ای درباره اینکه چطور الگوهای طراحی معایب زبان برنامه‌نویسی هستند را منتشر کرد. یک زبان برنامه‌نویسی بهتر انتخاب کنید. حق کاملا با اوست. پرستش الگوها و این تفکر که «وای، من از الگوی X استفاده می‌کنم.»
>
> — برندن آیک در کدنویس‌ها در کار - نقطه نظراتی در هنر برنامه‌نویسی

در مهندسی نرم‌افزار، یک الگوی طراحی راهکاری قابل استفاده برای شماری از مشکلات تکرارشونده در طراحی نرم‌افزار است. یک الگو یک طراحی کامل به منظور

انتقال مستقیم به کد نیست. توضیح یا ایده‌ای برای چگونگی حل یک مشکل در شرایط گوناگون است. الگوهای طراحی شی-گرا معمولا رابطه و تعامل بین کلاس‌ها و اشیا را نمایش می‌دهند، بدون اشاره به کلاس‌ها و اشیای نهایی که درگیر هستند.

PHP از سبک‌های برنامه‌نویسی دستوری، تابعی، شی-گرا، رویه‌ای و انعکاسی پشتیبانی می‌کند. PHP یک جعبه ابزار بزرگ با ابزارهای بسیار متنوع است که امکان حل بسیاری مسائل را به شیوه‌های گوناگون - نه تنها یک شیوه - به وجود می‌آورد.

PHP درباره آزادی، راهکارهای سریع و مقیاس‌پذیر و دسترسی داشتن به شیوه‌های مختلف حل مساله است.

زمانی که تلاش می‌کنیم خود را ارتقا دهیم، و در این مورد بخصوص کد خود را، بعضی وقت‌ها در فلسفه یک الگوی طراحی مشخص طوری غرق می‌شویم که در عمل قدرت تفکر را از دست می‌دهیم.

> زمانی که الگوها را در برنامه خود می‌بینم، آن‌ها را مشکل فرض می‌کنم. شکل یک برنامه تنها باید مشکل مورد نیاز آن را پوشش دهد. هر تغییر دیگر در کد برای من نشانه‌ای است، که دارم از عوامل انتزاعی نه چندان قدرتمند استفاده می‌کنم - اغلب به طوری که با دست خودم برخی ماکروهای مورد نیاز را تولید می‌کنم.
>
> — پال گراهام

ما نباید در فلسفه یا ایده یک الگوی طراحی بخصوص غرق شویم. نگرانی اصلی باید ساده نگه داشتن کد برای پیمایش و درک آن باشد به این منظور که نگهداری و ایمن ساختن کد ساده گردد.

همچنین باید به یاد داشته باشیم که چیزی با نام ضد-الگو نیز وجود دارد. این الگویی است که ممکن است زیاد مورد استفاده قرار گیرد اما در عمل ناکارآمد و/یا بی‌فایده است.

> تصور من این است که الگوها در ابتدا برای حل مشکلات متداول بوجود آمده‌اند. اما اکنون که مدت زمان زیادی از آن‌ها می‌گذرد و تجربه برنامه‌هایی را داریم که ده‌ها بار پیچیده‌تر با این الگوها طراحی شده‌اند چرا که افراد اصرار دارند تمام الگوهایی که راجع به آن مطالعه می‌کنند را پیاده‌سازی کنند (برنامه من معماری خوبی داره، چون که با الگوها همراهه.) نظر من راجع به ارزش اولیه این الگوها تغییر کرده.
>
> پال ویتون در الگوهای طراحی شیطانی

همیشه از رویکردی عملگرا استفاده کنید:

> اقدام یا عملی که با در نظر گرفتن عواقب عملی فوری در مقایسه با نظریه و عقیده دیکته می‌شود.
>
> — لغتنامه انگلیسی کالینز، ویرایش ۱۲، سال ۲۰۱۴

مسیر اشتباه: جستجو برای الگویی که مشکل را حل کند. 

همیشه از برنامه‌نویسی شی-گرا استفاده کنید

> مشکل زبان‌های شی-گرا این است که همه آن‌ها یک محیط کامل را با خود حمل می‌کنند. شما یک موز می‌خواستید ولی چیزی که نصیبتان شد گوریلی به همراه موز بود که داخل یک جنگل قرار داشت.
>
> — جو آرمسترانگ در کدنویس‌ها در کار - نقطه نظراتی در هنر برنامه‌نویسی

> انتزاع قدرتمند است. چیزی که من به آن حساسیت دارم، و چیزی که نسبت به آن واکنش نشان دادم این مزخرفات دهه ۹۰ درباره CORBA، COM و DCOM بود. هر استارت‌آپی در آن زمان برای اینکه یک "Hello World" ساده چاپ کند از ۲۰۰ هزار فراخوانی متد استفاده می‌کرد. این کار مسخره است! شما نمی‌خواهید برنامه‌نویسی باشید که با این چیزا سر و کار داشته باشه.
>
> — برندن آیک در کدنویس‌ها در کار - نقطه نظراتی در هنر برنامه‌نویسی

بسیاری از توسعه‌دهندگان نرم‌افزار و شرکت‌های نرم‌افزاری، تصور می‌کنند که برنامه‌نویسی شی-گرا تنها راه معقول برای توسعه نرم‌افزار در این روزها است. هر کسی که درباره برنامه‌نویسی شی-گرا مخالفتی انجام دهد بلافاصله با این واکنش روبه‌رو می‌گردد که مقابل "خرد جمعی" این صنعت قرار گرفته است.

در وبلاگ‌ها و انجمن‌های برنامه‌نویسی، بسیاری افراد وقت خود را صرف دفاع از برنامه‌نویسی شی-گرا کرده، افرادی که با قاطعیت در این زمینه اظهار نظر می‌کنند، بدون اینکه از هیچ روش استاندارد در این زمینه بهره بگیرند!

واقعیت این است که برنامه‌نویسی شی-گرا به این شکل تنها پیچیدگی ناخواسته‌ای را با خود همراه می‌آورد.

به عنوان برنامه‌نویسان و محققان رایانه ما باید بیاموزیم که تعصب را کنار گذاشته و بهترین راه حل موجود را برای یک مساله انتخاب کنیم.

امروزه، یکی از نقاط قوت PHP پشتیبانی از سبک‌های برنامه‌نویسی دستوری، تابعی، شی-گرا، رویه‌ای و انعکاسی است. PHP یک جعبه ابزار بزرگ با ابزارهای بسیار متنوع است که امکان حل بسیاری مسائل را به شیوه‌های گوناگون به وجود می‌آورد -**نه تنها یک شیوه!**

** به محض اینکه مسائل مختلف در یک برنامه را تنها محدود به یک سبک از برنامه‌نویسی کنیم، قابلیت تفکر خلاقانه را از خود صلب کرده و عملکرد بهینه‌ای نخواهیم داشت!**

یک درس تاریخی کوچک

یکی از بهترین راه‌های درک یک سبک برنامه‌نویسی این است که به تکامل آن توجه کنیم. عامل توسعه آن چه بوده است؟ چه مشکلاتی با سایر سبک‌های برنامه‌نویسی وجود داشت که نیاز به شیوه‌ای جدید احساس می‌شد؟ یک مشکل در جهان واقعی بود یا تنها در محیط دانشگاهی خلاصه می‌شد؟ و اینکه چگونه تکامل یافته است؟

مهم نیست که شخص X چه می‌گوید یا شخص Y چه تعریفی ارائه می‌دهد، چیزی که اهمیت دارد محیطی است که سبک‌های مختلف طی زمان در آن بوجود آمده‌اند.

> دو راه برای ایجاد یک طراحی نرم‌افزار وجود دارد. یک راه این است که به قدری ساده باشد که آشکارا هیچ کمبودی نباشد و راه دیگر این است که به قدری پیچیده باشد که هیچ کمبودی آشکار نباشد.
>
> — چارلز آنتونی ریچاردز هور

در گذشته، قبل از ظهور برنامه‌نویسی شی-گرا، تقریبا اواخر دهه ۱۹۵۰، نرم‌افزار با استفاده از زبان‌های برنامه‌نویسی توسعه می‌یافت که ساخت‌یافته نبودند، که گاهی اوقات به زبان‌های نسل اول و دوم نامیده می‌شوند. برنامه‌نویسی بدون ساختار (یا ساختار نیافته) به لحاظ تاریخی قدیمی‌ترین سبک برنامه‌نویسی است. از این سبک

به شدت برای تولید کد اسپاگتی انتقاد می‌شد.

برای این سبک برنامه‌نویسی زبان‌های سطح بالا و پایین وجود داشتند. این زبان‌ها شامل نسخه‌های اولیه JOSS، FOCAL، TELECOMP، Basic، COBOL، MUMPS، کد سطح ماشین، سیستم‌های اولیه اسمبلر (آن‌هایی که شامل عملگرهای رویه‌ای متا نبودند) و برخی زبان‌های اسکریپتی می‌شوند.

برنامه‌ای در زبان بدون ساختار معمولا شامل دستورات یا عبارت‌های ترتیبی و پشت سر هم است. خطوط معمولا دارای شماره یا برچسب هستند که امکان تغییر جریان برنامه را به قسمتی دیگر ممکن می‌سازد (مانند عبارت فراموش شده GOTO).

سپس، در دهه ۱۹۶۰ برنامه‌نویسی ساخت‌یافته ظهور یافت - اغلب به دلیل نامه مشهور ادگار دایجسترا عبارت‌های Go To مضر هستند

برنامه‌نویسی ساخت‌یافته سبکی از برنامه‌نویسی است که با استفاده از ساب‌روتین‌ها، ساختارهای بلاک و حلقه‌ها به وضوح، کیفیت و توسعه نرم‌افزار کمک می‌کند. این سبک در مقابل استفاده از عبارت GOTO برای پرش به قسمت‌های دیگر برنامه قرار می‌گیرد.

بعدها، برنامه‌نویسی رویه‌ای از برنامه‌نویسی ساخت‌یافته مشتق شد. برنامه‌نویسی رویه‌ای مبتنی بر مفهوم فراخوانی رویه است. یک فراخوانی رویه نام دیگری برای فراخوانی تابع است. رویه‌ها همچنین به نام‌های روتین، ساب‌روتین یا متد شناخته می‌شوند. یک رویه تنها شامل برخی دستورات محاسباتی است که باید فراخوانی شوند. هر رویه می‌تواند در هر نقطه از برنامه هنگام اجرای آن توسط خود یا سایر رویه‌ها فراخوانی شود.

در ابتدا تمام رویه‌ها به عنوان داده سراسری در هر قسمت از برنامه قابل دسترس بودند. در برنامه‌های کوچک این مشکل حادی نبود، اما با پیچیده‌تر شدن برنامه و افزایش حجم آن، یک تغییر کوچک در یک قسمت از برنامه روی سایر قسمت‌ها نیز تاثیرگذار بود.

هیچکس قصد ایجاد تغییر در برنامه‌ای با وابستگی‌های گوناگون را نداشت. یک تغییر کوچک در یک رویه منجر به بروز سلسله‌ای از خطا در سایر رویه‌های مبتنی بر کد اصلی بودند.

تکنیک جدیدی تکامل یافت که اجازه می‌داد داده به حوزه‌های کوچک‌تری بنام اشیا تقسیم شود. تنها رویه‌های خاصی که به آن حوزه اختصاص داشتند می‌توانستند به آن داده دسترسی داشته باشند. به این عمل پنهان‌سازی داده یا encapsulation گفته می‌شود. نتیجه این عمل شامل کد بهتر و مرتب‌تری می‌شد.

در ابتدا از اشیا به این نام یاد نمی‌شد، چرا که تنها به عنوان حوزه‌های جدا از هم دیده می‌شدند. بعدها که وابستگی‌ها کاهش یافتند و ارتباط بین رویه‌ها و متغیرهای یک حوزه به عنوان قسمت‌های مجزا در نظر گرفته شد، مفهوم شی و برنامه‌نویسی شی-گرا بوجود آمد.

بعدها، اغلب به دلیل توسعه جاوا، برخی اصطلاحات بوجود آمدند و یک رویه یا تابع دیگر با این نام صدا زده نمی‌شدند، از آنجا که در حوزه جداگانه‌ای قرار داشت، متد نام گرفت. متغیرها نیز دیگر با این نام صدا زده نمی‌شدند بلکه هنگام قرارگیری در یک حوزه جداگانه نام ویژگی به خود گرفتند.

بنابراین یک شی در اصل مجموعه‌ای از متغیرها و توابع است که اکنون ویژگی و متد نامیده می‌شود.

شیوه‌ای که ویژگی‌ها و متدها درون یک حوزه جداگانه قرار می‌گیرند تعریف یک کلاس است. زمانی که این کلاس مقداردهی اولیه شود، یک شی نامیده می‌شود.

اشیا می‌توانند به یکدیگر ارجاع داده شوند و با چنین ارجاعی، متدها (توابع) درونی می‌توانند با یکدیگر ارتباط برقرار کنند. اشیا همچنین می‌توانند متدهای سایر اشیا را به ارث ببرند که به این عمل وراثت گفته می‌شود. این راهی برای استفاده مجدد از کد و امکان ایجاد افزونه‌های مجزا از نرم‌افزار با استفاده از کلاس‌ها و رابط‌های عمومی است. روابط اشیا به یک سلسله‌مراتب جدید ختم شد. مفهوم وراثت در سال ۱۹۶۷ برای زبان برنامه‌نویسی Simula 67 ابداع شد.

اشیا همچنین می‌توانند متدهای سایر اشیا را به ارث برده و عملکرد اصلی آن را بازنویسی کنند که به این مفهوم چندریختی گفته می‌شود.

اینکه چگونه این ایده‌های مختلف پیاده‌سازی می‌شوند از یک زبان برنامه‌نویسی به دیگری متفاوت است.

برنامه‌نویسی شی-گرا درباره سازمان‌دهی کد نسبت به سبک‌های قبل از خود است. یک افزونه برای برنامه‌نویسی رویه‌ای به حساب می‌آید و درباره پنهان‌سازی داده (encapsulation) و جلوگیری از حوزه سراسری است. درباره توسعه توابع با قرض‌گرفتن ساختار آن‌ها و تغییر عملکرد بدون تاثیر روی کد اصلی است (inheritance). همچنین درباره بازنویسی توابع بدون تغییر در کد اصلی است (polymophism).

> مدل شی-گرا به تولید سریع برنامه‌ها کمک می‌کند. چیزی که در عمل اتفاق می‌افتد این است که روشی ساخت‌یافته برای تولید کد اسپاگتی فراهم می‌کند.
>
> پل گراهام در Ansi Common Lisp

مسیر اشتباه: همیشه از برنامه‌نویسی شی-گرا استفاده کنید.

ترس از کد دیگران

بحثی که اغلب برای کاربرد یک چارچوب نرم‌افزاری مطرح می‌شود این است که افراد نمی‌خواهند با کدهایی که از ابتدا توسط دیگران نوشته شده است سر و کار داشته باشند.

اگرچه این یک ذهنیت عجیب در میان توسعه‌دهندگان وب موجود در جامعه کاربری PHP است. تفکری که نشان از عدم وجود تجربه و کار حرفه‌ای دارد.

نوشتن نرم‌افزار و سر و کار داشتن با کد دیگران امری عادی است. در حقیقت بخشی از کار روزانه یک برنامه‌نویس حرفه‌ای است. چیزی نیست که از آن بترسیم.

یک برنامه‌نویس حرفه‌ای به کد دیگران با این دید نگاه نمی‌کند که وی چطور با استفاده از دانش برنامه‌نویس سابق کاری را انجام داده است، فردی که ممکن است اکنون در پروژه یا شرکت وجود نداشته باشد و اگر برنامه‌نویس سابق از چارچوب نرم‌افزاری فلان یا بمان استفاده می‌کرد همه کارها به خوبی انجام می‌شد.

این طرز تفکر یک برنامه‌نویس حرفه‌ای نیست. هیچکس اینکار را انجام نمی‌دهد.

شاید موانع کوچکی که در ابتدا برای توسعه نرم‌افزار وب با PHP وجود دارد به چنین طرز تفکری دامن می‌زند. صرف نظر از آن، این نشانه فردی است که در مسیر اشتباه کاری قرار گرفته است.

قسمت بزرگی از برنامه‌نویسی شامل سر و کار داشتن با کد دیگران است. قسمتی از کاری که تلاش برای ارتقا بخشی از کد یا بازنویسی کامل آن را داشته باشد.

با مطالعه کتاب کدنویس‌ها در کار - نقطه نظراتی در هنر برنامه‌نویسی ، یادداشت‌های بزرگترین اساتید برنامه‌نویسی را به خاطر بسپارید.

برخی از بزرگترین و موفق‌ترین پایگاه‌های کد موجود در دنیا توسط صدها نفر از افرادی که حتی همدیگر را ملاقات نکرده‌اند ایجاد شده است، پایگاه‌های کدی که بدون استفاده از هیچ چارچوب نرم‌افزاری توسعه یافته‌اند، پایگاه‌های کدی که کاملا در یک زبان برنامه‌نویسی رویه‌ای بدون استفاده از چیزی بجز سبک رویه‌ای طراحی شده‌اند و هیچگاه آرزوی انجام آن به شیوه‌ای دیگر وجود نداشته است.

کرنل لینوکس که بالغ بر ۲۰ میلیون خط کد است به طور کامل توسط برنامه‌نویسی رویه‌ای آن هم با مشارکت بیش از ۱۴ هزار توسعه‌دهنده بدون استفاده از هیچ چارچوب نرم‌افزاری توسعه یافته است.

توزیع‌های گوناگون در BSD و اکثر برنامه‌های گنو/لینوکس به طور کامل توسط برنامه‌نویسی رویه‌ای بدون استفاده از هیچ چارچوب نرم‌افزاری توسعه یافته‌اند.

همین فرآیند درباره صدها پروژه اوپن سورس در دنیا که توسط برنامه‌نویس اصلی رها شده‌اند تا سایر برنامه‌نویسان علاقه‌مند آن را ادامه دهند، صدق می‌کند. بسیاری از این پروژه‌ها شامل مستندات اندک (اگر مستنداتی وجود داشت)، عدم وجود توضیحات در کد و راهنمایی‌های اولیه بودند.

تمام کد PHP به زبان C، یک زبان برنامه‌نویسی رویه‌ای، بدون استفاده از هیچ چارچوب نرم‌افزاری نوشته شده است.

زمانی که قصد تعریف یک کلاس در PHP یا راه‌اندازی چارچوب نرم‌افزاری مورد علاقه خود را دارید، به کد رویه‌ای نفر دیگری اتکا کرده‌اید!

البته، کد وحشتناک هم وجود دارد، کدی که از شروع خوب طراحی نشده است یا کدی که بنا بر تصمیم کارفرما هیچگاه بازنویسی صحیح درباره آن صورت نگرفته است، کدی که اینقدر بد است که نمی‌توان سر و ته آن را دستکاری کرد، اما هیچ چارچوب نرم‌افزاری از این وضعیت پیشگیری نمی‌کند. این اغلب فرآیند رشد طبیعی برنامه است. در حقیقت، هر چارچوب نرم‌افزاری به تکه‌های کوچک‌تری تجزیه می‌گردد.

البته که کد اسپاگتی وحشتناک نیز وجود دارد، اما هیچکس با قصد قبلی این کد را تولید نمی‌کند. بعضی وقت‌ها نتیجه نبود تجربه کافی است، اغلب از خطای کارفرما نشات می‌گیرد که حین فرآیند توسعه چندین مرتبه دست به تغییر قوانین می‌زند. در هر صورت، حتی اگر از چارچوب نرم‌افزاری نیز استفاده شود، نتیجه همان کد اسپاگتی خواهد بود و اصلا اهمیتی ندارد که چه میزان از سبک شی-گرا استفاده شده باشد، نتیجه همان کد اسپاگتی خواهد بود.

به عنوان برنامه‌نویسان، تلاش ما این است که از این وضعیت پیشگیری کنیم، اما این یک **فرآیند عادی** است، این همان **هنر برنامه‌نویسی** است، که قسمتی از **برنامه‌نویس بودن** است!

مسیر اشتباه: ترس از کد دیگران. 

پیروی کورکورانه از استانداردهای PHP-FIG

کلمه FIG مخفف "Framework Interoperability Group" یا گروه تعاملی چارچوب نرم‌افزاری است.

PHP-FIG توسط تعدادی از توسعه‌دهندگان چارچوب نرم‌افزار در کنفرانس php|tek سال ۲۰۰۹ ایجاد شد. از آن زمان، افراد دیگری نیز در پروژه مشارکت کرده‌اند، که اندازه گروه را از ۵ به ۲۰ عضو افزایش داده است.

مباحث جنجالی زیادی مرتبط با PHP-FIG وجود دارد. برخی آن را بهترین اتفاق ممکن در جامعه کاربری PHP در حالی که برخی دیگر آن را لایق فراموشی می‌دانند.

یکی از مشکلات PHP-FIG این است که خود را در صفحه پرسش‌های متداول چنین معرفی کرده است:

> ایده اصلی گروه این است که نمایندگان پروژه بتوانند با یکدیگر در حل مشکلات عمومی خود مشورت کرده و راه حل ارائه دهند. مخاطب اصلی ما همدیگر هستیم، اما از این موضوع نیز آگاهیم که جامعه کاربری PHP ما را زیر نظر دارند. اگر سایر دوستان نیز به کار ما علاقه داشته باشند ما از آن‌ها استقبال می‌کنیم، اما این هدف ما نیست. هیچکس در گروه به عنوان برنامه‌نویس نمی‌خواهد به شما بگوید که چطور برنامه خود را بسازید.

با این حال، زمانی که کار برخی از اعضای گروه را دنبال می‌کنیم، به وضوح مشاهده می‌شود که اهداف آن‌ها مغایر با جمله بالا است. این افراد تلاش خستگی‌ناپذیری انجام می‌دهند که PHP-FIG به عنوان یک "گروه استاندارد PHP" پذیرفته شود**چیزی که سابق بر این نام اصلی گروه بود**. آن‌ها با طبقه‌بندی کار PHP-FIG به عنوان "پیشرفته" در کتاب‌ها، وبسایت‌ها، وبلاگ‌ها و انجمن‌های خود تلاش می‌کنند کار دیگران را "عقب‌افتاده" جلوه دهند.

یکی از مشکلات PHP-FIG این است که با وجود پذیرش تعدادی از چارچوب‌های نرم‌افزاری و پروژه‌های اوپن سورس از برخی استانداردهای آنان، این استانداردها اغلب با مشکلات موجود در "حوزه چارچوب نرم‌افزاری" سر و کار دارند، که آن‌ها را تقریبا بی استفاده برای بسیاری وضعیت‌های صنعتی موجود در دنیای واقعی می‌سازد.

بسیاری افراد باید نرم‌افزار بهینه، امن و مطابق با هزینه را برای بخش صنعتی توسعه دهند که مشتریان تمایل به خرید آن را داشته باشند. آن‌ها نمی‌توانند درگیر استانداردهایی شوند که نیازمند تایید افراد متعصب در چارچوب‌های نرم‌افزاری باشد. اگر اینکار را می‌کردند، فاجعه‌ای در دنیای تجارت اتفاق می‌افتاد.

اگر قرار باشد نوعی گروه استاندارد بوجود بیاد باید دغدغه‌های تمام جامعه کاربری PHP را شامل شود، نه فقط توسعه‌دهندگان چارچوب نرم‌افزاری یا سیستم‌های مدیریت محتوا را. باید توسط خود توسعه‌دهندگان زبان برنامه‌نویسی PHP نمایندگی شود و امکان عضویت و حق رای برای بخش بزرگتری از افراد موجود باشد.

اگر قصد تبعیت از استانداردهای توسعه‌یافته توسط PHP-FIG را دارید، باید درک کنید که برخی از این استانداردها - از جمله استانداردهای PSR-0 و PSR-4 - تاثیر مستقیمی بر چگونگی کدنویسی شما خواهند داشت.

بسیاری صنایع نیازمند نرم‌افزار با ویژگی‌های مقیاس‌پذیری بالا، اجرای صحیح در مواقع بحرانی و به صرفه‌بودن از لحاظ اقتصادی هستند که نمی‌تواند توسط استانداردهای PHP-FIG توسعه یابد.

مسیر اشتباه: پیروی کورکورانه از استانداردهای PHP-FIG.

بی‌توجهی به امنیت

> مشکل برنامه‌نویسان این است که تنها زمانی سر از کارشان در می‌آورید که بسیار دیر شده باشد.
>
> — سیمور کری در defprogramming.com

کدنویسی ایمن به عمل نوشتن برنامه‌ها به صورتی که مقابل حملات افراد یا برنامه‌های مهاجم یا مخرب از خود مقاومت نشان دهد، گفته می‌شود. کدنویسی ایمن به حفظ داده از دزدیده‌شدن یا تخریب کمک می‌کند. به علاوه، یک برنامه ناامن می‌تواند برای مهاجم قابلیت دسترسی به سرور یا هویت فرد دیگری را فراهم سازد، که نتیجه آن از عدم دسترسی یک فرد به سرویس تا افشای اطلاعات محرمانه، از بین رفتن سرویس یا خسارت به سیستم‌های هزاران کاربر متفاوت است.

هر برنامه رایانه‌ای یک هدف باالقوه برای حمله به حساب می‌آید. مهاجمین تلاش می‌کنند تا آسیب‌پذیری‌های موجود در برنامه‌های شما را شناسایی کنند. سپس از این آسیب‌پذیری‌ها برای سرقت اطلاعات، خراب کردن برنامه‌ها و داده‌ها و دسترسی به سرورها و شبکه‌ها استفاده می‌کنند. در این مرحله است که دارایی‌های کارفرمای شما و اعتبار خودتان به خطر می‌افتد.

امنیت چیزی نیست که بتوان به نرم‌افزار اضافه کرد!

یک برنامه ناامن به طراحی بسیار زیادی برای امن شدن نیاز دارد. شما باید طبیعت تهدیدات موجود برای نرم‌افزار خود را شناسایی کرده و با استفاده از الگوهای کدنویسی ایمن از ابتدا و در زمان طرح‌ریزی و توسعه نرم‌افزار به فکر مقابله با این تهدیدات باشید.

ایمن‌سازی منابع بحرانی نرم‌افزار نسبت به گذشته اهمیت بیشتری یافته است، چرا که تمرکز مهاجمین به سمت لایه برنامه معطوف شده است. یک گزارش SANS در سال ۲۰۰۹ نشان می‌دهد که حملات علیه برنامه‌های وب بیش از ۶۰٪ کل حملات موجود در اینترنت را شامل می‌شود.

PHP از این نظر که هم یک زبان برنامه‌نویسی است هم یک چارچوب نرم‌افزاری وب، غیرعادی است. این بدان معنی است که PHP شامل بسیاری ویژگی‌های وب درون خود است که نوشتن کد ناامن را بسیار ساده می‌کند.

ایمن به صورت پیشفرض

> پیچیدگی در نرم‌افزار کشنده است. چیزی است که جان توسعه‌دهندگان را می‌گیرد، باعث دشواری فرآیند طراحی، ساخت و آزمون می‌شود و بسیاری چالش‌های امنیتی را برای کاربر و مدیر سیستم بوجود می‌آورد.

به منظور طراحی و پیاده‌سازی برنامه‌ها با استفاده از پیشنیازهای مناسب امنیتی، الگوهای کدنویسی ایمن و تمرکز بر خطرات امنیتی باید به چرخه روزانه تفکر، عمل و فرآیند توسعه اضافه گردد.

در حالت کلی، مقرون به صرفه است که از ابتدا به فکر مشکلات امنیتی در نرم‌افزار باشیم قبل از اینکه نسخه نهایی آن کامل شود، همچنین هزینه‌های مرتبط با نقص‌های امنیتی نیز ماجرای خود را دارند.

مسیر اشتباه: توسعه ندادن نرم‌افزار امن به صورت پیشفرض. 

پرسش‌های متداول

سوءتفاهم نسبت به یک نوشته کار راحتی است، پس بهتر است برخی مسائل را روشن کنیم.

پرسش: هدف این وبسایت چیه و چرا رویکردی مخالف داره؟

پاسخ: برای ایجاد بحث و گفتگو و تفکر درباره رویکردهای فعلی و دیدگاه‌های افراطی.

پرسش: می‌خوای بگی برنامه‌نویسی شی-گرا بد و اشتباهه؟

پاسخ: البته که نه! ما می‌گوییم اینکه همیشه برای حل مسائل از برنامه‌نویسی شی-گرا استفاده شود بد است. وقتی که فقط به سیاه و سفید فکر می‌کنید، بد است.

حتی درون یک برنامه مشکلات مختلفی وجود دارد. استفاده از سبک‌های مختلف بعضی وقت‌ها بهترین گزینه است، تمام برمی‌گردد به مشکلی که قصد حل آن را دارید.

زمانی که یک راه حل ناقص را برای مساله‌ای مشخص به کار می‌برید اتفاقات بدی می‌افتد.

پرسش: می‌خوای بگی تمام چارچوب‌های نرم‌افزاری بدن؟

پاسخ: ما قصد نداریم که چارچوب نرم‌افزاری خاصی را قضاوت کنیم. مشکل ما استفاده همیشگی از یک چارچوب نرم‌افزاری در PHP است.

پرسش: اگه یه چارچوب نرم‌افزاری بهم کمک کنه که زود راه بیفتم، مشکلش چیه؟

پاسخ: اگر موقعیت را بررسی و تحلیل کرده باشید و به پیامدهای بلند مدت آن بیندیشید و دغدغه شما این باشد که زود راه بیفتم، اصلا بد نیست. اما اینجا دیگر صحبتی از برنامه‌نویسی یا توسعه نرم‌افزار در میان نیست، چرا که داریم درباره راهکارهای نشانه بگیر و کلیک کن صحبت می‌کنیم.

زود راه افتادن ارتباطی با طراحی نرم‌افزار ندارد، بیشتر به این معنی است که مشکل پیش روی خود را تحلیل نکرده‌اید و درکی از پیامدهای بلند مدت انتخاب خود ندارید.

پرسش: می‌خوای بگی بسته‌های شخص ثالث بدن؟

پاسخ: نه، ما استفاده از کتابخانه‌های شخص ثالث را توصیه می‌کنیم. کدی که به راحتی می‌توانید در پروژه خود وارد کرده بدون آنکه کوچکترین محدودیتی برای شما به همراه داشته باشد. این عالی است!

پرسش؛ تو کی هستی؟

پاسخ: این وبسایت درباره ایده‌ها و مقابله با افراط‌گرایی در جامعه کاربری PHP است، درباره شهرت شخصی و شناخته شدن نیست. نام بردن از افراد باعث می‌شود که تمرکز از روی موضوعات مطرح شده در وبسایت بر روی افراد مطرح کننده آن معطوف شود. فقط روی ایده‌های مطرح شده تمرکز کنید.

پرسش: تجربه‌ات تو توسعه نرم‌افزار چطوره؟

پاسخ: ایده‌ها، تفکرات و نتیجه‌گیری‌های موجود در این وبسایت نیاز به تجربه زیادی ندارد اگر روی موضوع اصلی تمرکز کنید که آن همان انجام یک کار مشخص است چرا که دیگران می‌گویند.

توصیه به مطالعه

PHP مسیر اشتباه در Hacker News

- زمانی که PHP - مسیر اشتباه منتشر شد، دیدگاه‌های مختلفی را در Hacker News بوجود آورد که ارزش مطالعه دارند.

چرا کد بد در محیط علمی بر کدی که بهترین رویکرد را دنبال کرده غلبه می‌کند

- ذهنیت منفرد، بدون مراقبت و ساده می‌تواند نسبت به قواعد خوب و قدرتمند صنعتی که بزرگراهی به جهنم را سنگفرش کرده‌اند، بهتر باشد. دنیای واقعی بیرون رایانه پر از چنین نمونه‌هایی است.

چگونه بدون مدل شی-گرا برنامه‌نویسی کنید

- به عنوان رویکردی تازه و جایگزین، برایان ویل طی سه ویدیو بیان می‌کند چرا برنامه‌نویسی شی-گرا در آغاز ایده بدی است و مجموعه را با نکاتی درباره چگونگی کدنویسی بدون شی-گرایی به اتمام می‌رساند.

کدنویس‌ها در کار - نقطه نظراتی در هنر برنامه‌نویسی

- بر اساس بیش از ۸۰ ساعت مصاحبه با ۱۵ تا از بهترین برنامه‌نویسان و دانشمندان علوم رایانه، پرسش و پاسخ‌های موجود در این کتاب دیدگاهی جامع درباره چگونگی برنامه‌نویس شدن این افراد، چگونگی تمرین کردن آن‌ها و اینکه درباره آینده برنامه‌نویسی چه فکری می‌کنند را شامل می‌شود.

ویژگی‌های یک برنامه‌نویس کارآمد

- صلاحیت یعنی تجربه و دانش کافی داشتن درباره انجام یک کار؛ کارآمدی شامل آگاهی شما از چگونگی انجام یک کار است و اینکه چطور در تصویر بزرگتری قرار می‌گیرد. به عبارت دیگر، یک متخصص کارآمد همیشه یک متخصص باصلاحیت است، اما خلاف آن ممکن است صحیح نباشد.

راهنمای کدنویسی ایمن در OWASP

- این سند به شدت فنی، مجموعه‌ای از الگوهای کدنویسی ایمن در نرم‌افزار را به صورت چک‌لیست بیان می‌کند که می‌توانند درون چرخه تولید نرم‌افزار قرار گیرند. پیاده‌سازی این الگوها به کاهش متداول‌ترین آسیب‌پذیری‌های نرم‌افزاری می‌انجامد.

اصول امنیت در طراحی

- امنیت برنامه وب یک جز ضروری از هر پروژه موفق است، خواه برنامه‌های اوپن سورس PHP باشد خواه وبسایت‌های تجاری انحصاری. سرویس‌های میزبانی از کد ناامن استقبال نمی‌کنند و کاربران نیز از چنین سرویس‌هایی که منجر به کلاه‌برداری می‌شود، دوری می‌کنند. هدف این راهنمای توسعه

امکان ایجاد برنامه‌های ایمن وب برای کسب و کارها، توسط توسعه‌دهندگان، طراحان و معماران نرم‌افزار است. نرم‌افزار امن، اگر از مراحل ابتدایی شروع شود، تقریبا همان هزینه‌ای را در بر دارد که نرم‌افزارهای ناامن تقبل می‌کنند اما در ادامه بسیار مقرون به صرفه‌تر خواهند بود.

PHP Security: از انتهای عمیق نجات یابید

- از آنجا که هر رخنه جدی امنیتی به سرعت در کنفرانس‌های خبری و وبسایت‌ها منتشر می‌شود؛ امنیت برای آن‌ها خیلی مهم است و آن را بسیار جدی می‌گیرند. جدی گرفتن این توصیه‌ها قبل از اینکه کار از کار بگذرد، امری ضروری است.

بهینه‌سازی موجب بهبود در طراحی کد فعلی می‌شود

- بهینه‌سازی کد درباره بهبود عملکرد کد فعلی است. درباره تغییر در ساختار نرم‌افزار است به گونه‌ای که عملکرد خارجی آن تغییر نکند ولی منجر به بهبود ساختار درونی آن شود. با بهینه‌سازی کد حتی می‌توان یک طراحی بد را به شیوه‌ای خوب بازنویسی کرد. این کتاب به بررسی اصول بنیادی در بهینه‌سازی کد می‌پردازد، از جمله اینکه در کدام قسمت اینکار صورت بگیرد و چگونه آزمون‌های مورد نیاز را برپا کنید. همچنین کاتالوگ بیش از ۴۰ روش بهینه‌سازی کد به همراه جزئیات پیاده‌سازی هر کدام به صورت قدم به قدم وجود دارد به همراه نمونه‌هایی که کاربرد آن‌ها را نشان می‌دهند. این کتاب با تمرکز بر جاوا به عنوان زبان مرجع آن نوشته شده است اما مفاهیم آن به تمام زبان‌های شی-گرا قابل تعمیم هستند.

تمرین برنامه‌نویسی

- خلاصه‌ای از مجموعه تمرین‌های مهم برای برنامه‌نویسان فعال.

برنامه‌نویس عملگرا

- برنامه‌نویس عملگرا: از مرد مسافر تا استاد، فرآیند درونی برنامه‌نویسی را بیان می‌کند؛ انتخاب یک نیازمندی و تولید کد کارآمد و قابل بازبینی که نیاز کاربران را رفع می‌کند. این کتاب شامل موضوعاتی از مسئولیت‌پذیری انفرادی و توسعه محیط کار تا تکنیک‌های معماری برای نگهداری کد به شیوه‌ای منعطف، آسان و قابل استفاده مجدد را شامل می‌شود.

درک زبان‌های برنامه‌نویسی

- انتخاب یک زبان برنامه‌نویسی یکی از مهم‌ترین عواملی است که در کیفیت نهایی سیستم نرم‌افزاری تاثیر می‌گذارد. متاسفانه، بسیاری برنامه‌نویسان توانایی کمی در قابلیت‌های زبانی خود دارند: آن‌ها به شدت عاشق زبان بومی خود هستند اما قادر به تحلیل محدودیت‌های آن نیستند. درک زبان‌های برنامه‌نویسی با این هدف نوشته شده است که چه گزینه‌هایی برای طراحان زبان موجود است؛ ساختارهای زبانی چگونه باید به شکلی امن و خوانا استفاده

گردند؛ ساختارهای زبانی چگونه پیاده‌سازی شده‌اند و کدام یک به شیوه‌ای موثر می‌توانند کامپایل گردند؛ و اینکه نقش زبان در بیان شیوه‌های انتزاعی چگونه است.

چگونگی مشارکت

در گیت‌هاب مشارکت کنید.

- این مخزن را پس از Clone می‌توانید ویرایش کنید.
- برای اعمال تغییرات خود Pull Request دهید.

زبان مورد نظر خود را می‌توانید در قسمت *sections/LANGUAGE* ایجاد کرده یا به ویرایش قسمت‌های موجود بپردازید.