دانشکده فنی تویسرکان
گروه مهندسی رایانه

پایان‌نامه‌ی کارشناسی

عنوان

پیاده‌سازی چهارچوب نرم‌افزاری برای توسعه‌ی نرم‌افزارهای تحت وب

نگارش

محمدرضا طیبی

استاد راهنما

مهندس محمدحسین بام‌نشین

زمستان ۱۳۹۹

تقدیم به

شاعرانی که در دانشگاه مهندسی خواندند،
صنعت‌گران گرفتار در عصر دلالی،
اندیشمندان بی قلم،
و
کودکان کار

از

پدرم، که با ساختن و فهمیدن شادمان می‌شود،
و سرلوحه‌اش در ساختن، نوآوری، دقت، و ظرافت است؛

مادرم، که آهستگی و پیوستگی را شرط پیشرفت می‌داند؛
برادرم، که حامی شجاعی است؛

میثم نگه‌دار پنیرانی برای چهار سال همدلی؛

و

استاد محمدحسین بامنشین برای نمایشی بی بدیل از شرافتِ کاری و دلسوزیِ بی چشم‌داشت؛

سپاس‌گزارم.

چکیده

نرم‌افزار می‌تواند گلوگاه بخش بزرگی از قابلیت‌ها و یا محدودیت‌های یک کسب و کار مبتنی بر اینترنت باشد. هر کسب و کار از دیدگاه مدیریتی دارای چندین لایه با اقتضائات متفاوت از بازاریابی و فروش، تا مدیریت داخلی است؛ همچنین هر نرم‌افزار تحت وب دارای سطوح مختلفی از نیازها به مجرد اقتضائات مهندسی نرم‌افزار است. هدف از چهارچوب نرم‌افزاری حاضر، پیاده‌سازی سامانه‌ای قابل بسط و گسترش در تمامی سطوح مدیریتی و نرم‌افزاری است تا بتوان با صرفه‌جویی در هزینه‌های توسعه، سامانه‌ای برای پاسخ به نیاز صنعت در کسب و کارهای مبتنی بر اینترنت، با قابلیت‌های توسعه‌ی سریع نرم‌افزار فراهم نمود.

واژه‌های کلیدی

چهارچوب نرم‌افزاری، یک‌پارچه‌سازی، معماری سیستم‌ها، درگاه برنامه‌نویسی برنامه‌ی کاربردی (API)

فهرست

فهرست

فصل اول
مقدمه

۱ پیش‌گفتار

چهارچوب‌های نرم‌افزاری ساخته می‌شوند تا هزینه‌های مختلفی از فرایند تولید و بهره‌برداری از نرم‌افزار کاهش دهند. این هزینه‌ها شامل هزینه‌ی تحلیل، معماری، توسعه، تا استفاده از منابع سخت‌افزاری و سرعت پاسخگویی را شامل می‌شوند.

بسیاری از نرم‌افزارهای تحت وب (مانند برندهای شخصی، امور مدیریتی کسب‌وکارها، و برخی شبکه‌های اجتماعی) از الگوی چرخه‌ی حیات (Life Cycle) نسبتاً مشابهی پیروی می‌کنند؛ به این صورت که: درخواست (Request) در قالبی استاندارد، از سمت مخاطب (Client) ارسال می‌شود، پردازش می‌شود، و پاسخ (Response) به مشتری بازگردانده می‌شود، و در تمام این مدت، ارتباط برقرار (Persistent Connection) است.

تحلیل چنین برنامه‌هایی از نظر مهندسی نرم‌افزار مدت‌هاست که به مسائلی حل شده تبدیل شده‌اند و امروزه متخصصان و واحدهای صنعتی در پی بهینه‌سازی (Optimization) این راه‌کارهای نرم‌افزاری (Software Solutions) هستند. معطوف شدن توجه‌ها به معماری Model-View-Controller (MVC) و تعاملات RESTful با ساختار JSON از پیامدهای همین احساس نیاز در صنعت است.

چهارچوب نرم‌افزاری حاضر امکان توسعه‌ی سمت کارپرداز (Server-Side) را ساده‌تر ممکن می‌کند.

۲ توجیه

امروزه کسب‌وکارها و تشکیلات به دنبال پیاده‌سازی پایگاه‌های اطلاعاتی و تعاملی در اینترنت و سایر شبکه‌ها هستند. صنعت نرم‌افزار، همواره به دنبال اینجاد راه‌هایی برای توسعه‌ی سریع‌تر و همچنین اجرای سریع‌تر برنامه‌ها بوده است. برنامه‌های تحت وب نیز از این روال جدا نبوده‌اند.

از آن‌جا که طراحی و تولید برنامه‌های مشابه از الگوهای مشابه پیروی می‌کند، می‌توان آن‌ها را در قالب چهارچوب‌های نرم‌افزاری (Software Frameworks) پیاده‌سازی کرد.

زبان‌های برنامه‌نویسی برای این موضوع طراحی شده‌اند تا علاوه بر سهولت برنامه‌نویسی برای اپراتور، برنامه‌نویس را از دغدغه‌ی درخواست‌های سطح پایین سیستمی برهانند. سپس چهارچوب‌های نرم

افزاری در سطح کامپایلر به کار گرفته شدند تا برنامه‌نویس‌ها را از دغدغه‌های استفاده از منابع، i/o، شبکه، آرایه‌ها، معادلات ریاضی و غیره آزاد کنند.

زمانی که صحبت از تولید برنامه‌های سازمانی به میان آمد، معماری‌های پیشنهادی برای تولید نرم‌افزاره (Software Architectures) به سمتی حرکت کرد که در کنار Performance مناسب، خوانایی و قواعدی نظام‌مند داشته باشند تا مدیران پروژه بتوانند تمرکز توسعه‌ی نرم‌افزار را بر منطق‌های کسب‌وکار معطوف کنند. این معماری‌ها به نحوی طراحی شده‌اند که با یک‌بار پیاده‌سازی منطق‌های ارتباط با پایگاه‌داده، ارتباط با نرم‌افزارهای سوم شخص، ارتباط با انسان، و غیره، در قالب توابع و چرخه‌ی زندگی نرم‌افزار (Software Life Cycle)، تنها آن‌ها را فراخوانی کرد؛ برای مثال این فلسفه در معماری دولت الترونیک سنگاپور در سطح کلان و بین سیستمی به کار گرفته شد، و تبدیل به اصلی شد که آن را «اصلِ فقط یک بار» (Once Only Principal) می‌نامند و معنای آن این است که دولت حق ندارد داده‌ی مشابهی را بیش از یک‌بار از افراد دریافت کند.

در نتجه می‌توان با پیاده‌سازی یک چهارچوب نرم‌افزاری در سطحی بالاتر، دغدغه را از معماری و موارد اولیه‌ی امنیت هم در سطح بالاتری ببرد تا توجه توسعه‌دهندگان را به نیازهای کسب‌وکار معطوف کند. چنین سیستمی به اپراتور اجازه می‌دهد تا با یک‌بار پیاده‌سازی سطح دسترسی کاربران، و یک‌بار پیاده‌سازی برنامه‌ی اجرای فرایندی خاص، به سرعت نیازهای کسب‌وکار را مرتفع کند.

چهارچوب نرم‌افزاری حاضر، ارتباط با برنامه‌های موبایل، سرورهای شخص سوم، و برخی از قابلیت‌های توسعه‌ی سریع منطق کسب‌وکار مانند ایجاد اعمال CRUD بر روی جداول را در اولویت پیاده‌سازی خود دارد.

با استفاده از چهارچوب نرم‌افزاری حاضر که بر پایه‌ی معماری MVC RESTful API ایجاد شده است، می‌توان به سرعت منطق کسب‌وکار را بدون نگرانی از شیوه‌ی ارتباط با موبایل و پایگاه‌داده پیاده‌سازی کرده و برنامه را منتشر کرد. این چهارچوب نرم‌افزار، فرایند مسیردهی (Routing) آدرس در لایه‌ی ۷ مدل OSI و تبدیل ورودی خروجی‌های JSON را همچنین ممکن کرده است.

لازم به ذکر است که این چهارچوب نرم‌افزاری در حال حاضر توسط برخی واحدهای صنعتی در حال بهره‌برداری است.

۳ تاریخچه‌ی این پروژه

۳.۱ سال ۱۳۹۳

ابتدا سال ۱۳۹۳ خورشیدی، این پروژه تحت عنوان سیستم مدیریت محتوای «شهر برفی» آغاز به کار کرد. که سورس کد آن پروژه در آدرس زیر در دسترس است.

https://github.com/tayyebi/SnowCity

۳.۲ سال ۱۳۹۵

آغاز مطالعات ایجاد یک فریم‌ورک بهینه و سریع بر پایه‌ی زبان‌های C++ و PHP انجام شد.

https://github.com/tayyebi/SnowCity-v۲

۳.۳ سال ۱۳۹۷

پیاده‌سازی یک «سامانه‌ی مدیریت دانش» با هدف پوشش دادن نیازهای بازار برای وب‌سایت‌های آموزشی، برندها، و مجلات آغاز شد.

https://github.com/Gordarg/SnowFramework

۳.۴ سال ۱۳۹۹

پروژه‌ی حاضر با هدف استفاده از امکانات سامانه‌ی مدیریت دانش مذکور بر پایه‌ی یک معماری بهینه، و همچنین به عنوان پروژه‌ی پایانی دوره‌ی کارشناسی مهندسی کامپیوتر، گرایش فناوری اطلاعات، به دپارتمان مهندسی کامپیوتر دانشکده‌ی فنی تویسرکان، دانشگاه بوعلی سینا، ارائه شد.

https://github.com/Gordarg/SF

۴ متن نرم‌افزار

این برنامه به صورت متن‌باز (Open Source) با پروانه‌ی MIT در پایگاه اینترنتی گیت‌هاب به نشانی زیر در دسترس است:

https://github.com/Gordarg/SF

۵ مستندسازی‌های برخط

برای مستند سازی‌های به روز به نشانی زیر مراجعه کنید:

https://gordarg.github.io/SF

فصل دوم
مفاهیم و تعاریف

۶ منطق کسب‌وکار (Business Logic)[۱]

به قواعد (Rules) یا الگورتیم‌های شخصی‌سازی شده (Custom) تبادل (Exchange) داده‌ها بین واسط کاربری (User Interface) و پایگاه‌داده (Database)، منطق کسب‌وکار گفته می‌شود. این قواعد می‌توانند بر اساس جریان‌های کاری (Workflows) کسب‌وکار ایجاد شده باشند. منطق کسب و کار در لایه‌ی بالاتری از کدهایی قرار می‌گیرد که برای نگه‌داری از زیرساخت پایه‌ی رایانه (Basic Computer Infrastructure) مورد استفاده قرار می‌گیرند.

برای مثال: یک قاعده‌ی کسب‌وکار (Business Rule) در یک بنگاه اقتصادی وجود دارد که افرادی که پیشاپیش هزینه‌ی ماهانه‌ی سرویس خود را می‌پردازند را به عنوان مشتریان خوش‌حساب نشانه‌گذاری می‌کند.

توجه: منطق کسب‌وکار لزوماً به فرایندهای پولی محدود نمی‌شود.

۷ کارساز (Server)

تعریف این واژه در علم شبکه‌های کامپیوتری آن است که اگر در لحظه مبادله‌ی اطلاعات در یک شبکه‌ی کامپیوتری، یک گره (Node) که می‌تواند هرگونه کامپیوتری (مانند تلفن همراه، لپ‌تاپ، و یا حتی پرینتر) باشد اگر ارسال کننده‌ی داده‌ها باشد کارساز (Server) نامیده شود و اگر دریافت‌کننده‌ی داده باشد مشتری (Client) باشد / باشند.

اما در تعریف‌هایی که متخصصان با دید کلی‌تری از یک سیستم ارائه می‌دهند، به صورت کلی به کامپیوتری که ابزار لازم برای ارائه‌ی سرویس‌های مشخص در شبکه را دارد و توسط یک یا چند کامپیوتر دیگر قابل دسترسی است سرور گفته می‌شود. سرورها معمولاً برخط (Online) و معمولاً همیشه آماده‌ی پاسخگویی به درخواست‌های ارسال شده‌ی مشتریان هستند.

در این سند، منظور از سرور، کامپیوتری است که از طریق شبکه می‌توان به سرویس‌دهنده‌ی وب آن دسترسی پیدا کرد. در ادامه پس از ارائه‌ی چند تعریف به شرح «سرویس‌دهنده‌ی وب» نیز خواهیم پرداخت.

---

۱ https://www.investopedia.com/terms/b/businesslogic.asp

۸ مدل (OSI) Open Systems Interconnection[۲]

مدل OSI یک چهارچوب انتزاعی است که برای تشریح شیوه‌ی عمل‌کرد یک سامانه‌ی شبکه استفاده می‌شود. این مدل، توابع عملیاتی را به عنوان یک مجموعه سراسری از قوانین و نیازها توصیف می کند تا از عملکردگرایی بین محصولات و نرم افزارهای مختلف پشتیبانی کند. در مدل مرجع OSI ارتباطات بین یک سامانه‌ی کامپیوتری در هفت لایه تقلیل یافته است. که به ترتیب ۱- فیزیکی (Physical)، ۲- اتصال داده (Data link)، ۳- شبکه (Network)، ۴- انتقال (Transport)؛ ۵- نشست(Session)؛ ۶- ارائه (Presentation)؛ ۷- کاربرد (Application)؛ هستند.

۸.۱ لایه‌ی فیزیکی

پایین‌ترین لایه‌ی مدل OSI که انتقال داده‌های بی ساختار خام (raw unstructured) را به صورت سیگنال‌های رادیویی، نوری، و یا الکتریکی، از خروجی فیزیکی دستگاه ارسال کننده (sender) به دستگاه گیرنده (reciever) بر عهده دارد را لایه‌ی فیزیکی نامند.

این لایه شامل دستگاه‌هایی مانند هاب‌ها (hub)، کابل‌ها، تقویت‌کننده‌ها (repeaters)، و یا تبدیل کننده‌های آنالوگ به دیجیتال (modem) است.

۸.۲ لایه اتصال داده

در این لایه داده‌ها در قالب فریم‌ها (frame) بسته‌بندی (packaging) می‌شوند. این لایه همچنین وظیفه‌ی اصلاح خطاهای احتمالی که در لایه‌ی فیزیکی رخ داده‌اند را بر عهده دارد.

این لایه شامل دو زیر لایه‌ی ۱- کنترل دسترسی رسانه (MAC) و ۲- کنترل منطقی اتصال (LLC) می‌شود. مک (MAC) امکان مدیریت جریان و تسهیم (multiplexing) را برای انتقال داده در سطح شبکه فراهم می‌کند. و کنترل اتصال منطقی کنترل جریان و خطاها را برعهده دارد.

۸.۳ لایه شبکه

این لایه مسول دریافت فریم‌ها از لایه‌ی اتصال داده و رساندن آن‌ها به مقصدهایی که در داخل فریم تعیین شده هستند. این لایه، مقصدها را بر اساس آدرس‌های منطقی مانند آی‌پی (Internet Protocol – IP) می‌شناسد. در این لایه، مسیریاب‌ها (router) تصمیم می‌گیرند که داده را به کدام سمت هدایت کنند.

---

۲ https://www.forcepoint.com/cyber-edu/osi-mode

نکته: روتینگ لایه‌ی ۳ با روتینگ لایه‌ی ۷ متفاوت است. مفهوم روتینگ که در معماری MVC برای وب بحث می‌شود، به معنی آدرس‌دهی به کدهای مربوط به هر درخواست کاربر است که در ادامه آن را به صورت کامل بررسی می‌کنیم.

۸.۴ لایه‌ی کاربرد

در این لایه کاربر نهایی (end user) و لایه‌ی کاربرد، هر دو، مستقیماً با نرم‌افزارهای کاربردی تعامل (interact) می‌کنند. نرم‌افزارهای کاربردی می‌توانند شامل مرورگرهای وب، کارسازهای وب، نرم‌افزارهای ایمیل، نمایشگرهای ویدیوی تحت RTMP، و غیره باشند.

نکته: چهارچوب نرم‌افزاری حاضر، به عنوان یک نرم‌افزار کاربردی، با این لایه در ارتباط است.

۹ پایگاه‌داده (Database)[۳]

پایگاه‌داده کلکسیونی (collection) – کلکسیون با مجموعه متفاوت است – از اطلاعات ساختار یافته و یا داده است، که معمولاً به صورت الکترونیکی در یک سیستم کامپیوتری ذخیره شده است.

۹.۱ سامانه‌ی مدیریت پایگاه‌داده (DBMS)

یک پایگاه‌داده معمولاً توسط یک دی‌بی‌اماس مدیریت می‌شود. پایگاه‌داده، دی‌بی‌اماس، و برنامه‌هایی که با آن‌ها در ارتباط هستند را به عنوان سامانه‌ی پایگاه‌داده (Database System) می‌شناسند که به صورت خلاصه پایگاه‌داده نامیده می‌شوند.

داده‌هایی که امروز در پایگاه‌های داده‌ی عملیاتی مورد استفاده قرار می‌گیرند، به صورت سطر و ستون‌های جداول مدل می‌شوند تا به پردازش و تعامل (querying) با داده‌ها را موثرتر کنند. این داده‌ها می‌توانند راحت‌تر مدیریت شوند، دسترس‌پذیر باشند، ویرایش شوند، به‌روزرسانی شوند، کنترل شوند، و سازماندهی شوند. بیشتر پایگاه‌های داده از زبان پرسش‌وپاسخ ساختار یافته (Structured Query Language – SQL) برای نوشتن و تعامل با داده‌ها استفاده می‌کنند.

---

۳ https://www.oracle.com/database/what-is-database

۱۰ زبان برنامه‌نویسی PHP[۴]

پیش‌پردازش‌گر ابر متن (Hypertext Preprocessor) یک زبان پر استفاده متن‌باز (Open source) است. کدهای پی‌اچ‌پی که اسکریپت (script) نامیده می‌شوند، روی کارسازاجرا (execute) می‌شوند.

پی‌اچ‌پی همه منظوره (general-purpose) است که برای توسعه‌ی مناسب‌سازی شده است. این زبان سریع، منعطف، و عمل‌گرا (pragmatic) است که از وب‌لاگ‌های شخصی تا وب‌سایت‌های بزرگ را می‌توان با آن توسعه داد.

۱۱ کارساز وب

کارساز وب نرم‌افزاری است که وظیفه‌ی پردازش صفحات پویای وب و نرم‌افزارهای تحت وب را برعهده دارد. این کارساز همچنین می‌تواند با اجرای یک یا چند نمونه (Instance) از سامانه‌ی مدیریت پایگاه‌داده، پردازش پایگاه‌های داده را نیز به عهده بگیرد؛ و یا ساز و کاری برای ارتباط با کارساز پایگاه‌داده داشته باشد. این مورد برای کارسازهای دیگر مانند کارسازهای پست الکترونیک (Mail Server)، کارساز فایل‌ها (File Server) غیره نیز قابل تعمیم است. در صورتی که یک کارساز وب پشتیبانی کاملی از خدمات موردنیاز یک وب‌سایت انجام دهد، می‌توان آن را یک میزبان (Host) برای آن نرم‌افزار تحت‌وب دانست.

همچنین می‌توان به این کارسازها آدرس‌های دامنه (Domain Name Address) اختصاص داد. اگر این کارساز به طور همزمان میزبان چند نرم‌افزار تحت‌وب مهمان باشد، می‌توان آن را یک هاست اشتراکی (Shared Host) دانست که از قابلیت هاست مجازی (Virtual Host) بهره می‌برد. لازم به ذکر است که هاست مجازی به معنی سرویس‌دادن به چند برنامه‌ی تحت‌وب توسط یک میزبان است که هر برنامه می‌تواند با یک یا چند آدرس دامنه شناخته شود؛ و این مفهوم با سرور اشتراکی (Shared Server) که به معنی کارسازی است که منابع خود را بین چند کاربر (User) به اشتراک گذاشته است، و یا مفهوم سرور مجازی (Virtual Server) که به معنی کارسازی است که با استفاده از تکنولوژی‌های شبیه‌سازی (Virtualization) مانند هایپروایزر (Hypervisor)ـ های مختلف در دسترس قرار گرفته است.

---

۴ https://www.php.net

۱۲ معماری نرم‌افزار[۵]

معماری به معنای طرح‌اولیه‌ی یک سامانه با شناخت رفتار آن است که سازوکار تقلیل‌یافته‌ای برای مدیریت پیچیدگی یک سامانه را ارائه کرده، و مکانیزم ارتباط و همکاری بین بخش‌ها را برقرار می‌کند. معماری، راه‌کاری ساختاریافته را برای پاسخگویی به تمامی نیازهای عملیاتی و فنی تعریف می‌کند؛ در عین حال ویژگی‌هایی برای بهینه‌سازی سامانه مانند کارایی (performance) و امنیت (security) را ایجاد می‌کند. تصمیم‌هایی که در معماری نرم‌افزار گرفته می‌شود ارتباط تنگاتنگی با نیازهای کسب‌وکار (business objectives) دارد.

هدف معماری تشریح ساختار سامانه و پنهان کردن جزئیات پیاده‌سازی آن برای تحقق سناریوهای (scenario) مختلف و موارد استفاده‌ی (use-case) سامانه است.

۱۳ الگوی MVC[۶]

این الگوی معماری که برای طراحی و ساخت واسط‌ها (interface) و ساختار یک برنامه به کار می‌رود، برنامه را به سه بخش (Model-View-Controller) که مجزا از هم و متصل به هم هستند تقسیم می‌کند؛ تا داده‌هایی را که به کاربر ارائه می‌شوند، از داده‌هایی که از کاربر دریافت می‌شوند تمیز دهد.

این الگو در طراحی برنامه‌های تحت وب و واسط‌های گرافیکی پر استفاده است.

۱۳.۱ کنترلر (Controller)

در یک برنامه‌ی MVC بخش زیادی از وظیفه‌ها معمولاً بر عهده‌ی کنترلرهاست. کنترلر وظیفه‌ی پشتیبانی از ورودی و خروجی و تبدیل‌های آن‌ها را بر عهده دارد. این بخش بین مدل‌ها و ویوها ارتباط برقرار می‌کند و خروجی اجرای کنترلر در ویو بازتاب می‌شود.

۱۳.۲ ویو (View)

این بخش به کاربر اجازه می‌دهد تا داده‌های مدل را ببینند. ویو می‌تواند از جداول، فرم‌ها، نمودارها و غیره استفاده کند.

---

۵ https://www.tutorialspoint.com/software_architecture_design/introduction.htm

۶ https://www.educba.com/what-is-mvc-design-pattern

۱۳.۳ مدل (Model)

این بخش اطلاعاتی در مورد کاربرد برنامه نگه‌داری می‌کند. وظیفه‌ی این بخش مدل‌سازی اطلاعات به اشیاء شناخته شده برای برنامه است. این بخش همچنین منطق و قوانین برنامه را کنترل می‌کند. در بسیاری از معماری‌ها، مدل‌ها وظیفه‌ی ارتباط با واسط Object-relational mapping (O/RM) و یا دی‌بی‌ام‌اس و یا دیگر شیوه‌های ذخیره و بازیابی داده‌ها را برعهده دارند.

۱۴ هسته‌ی نرم‌افزار (Software Core)

در یک معماری نرم‌افزار، «هسته» به آن بخشی از برنامه گفته می‌شود که عملکردها و توابع پایه‌ی برنامه در آن قرار دارد. دیگر بخش‌های برنامه، برای انجام عمل‌های پایه، توابع هسته را فراخوانی می‌کنند. هسته معمولاً در اولین مراحل چرخه‌ی زندگی یک برنامه اجرا می‌شود.

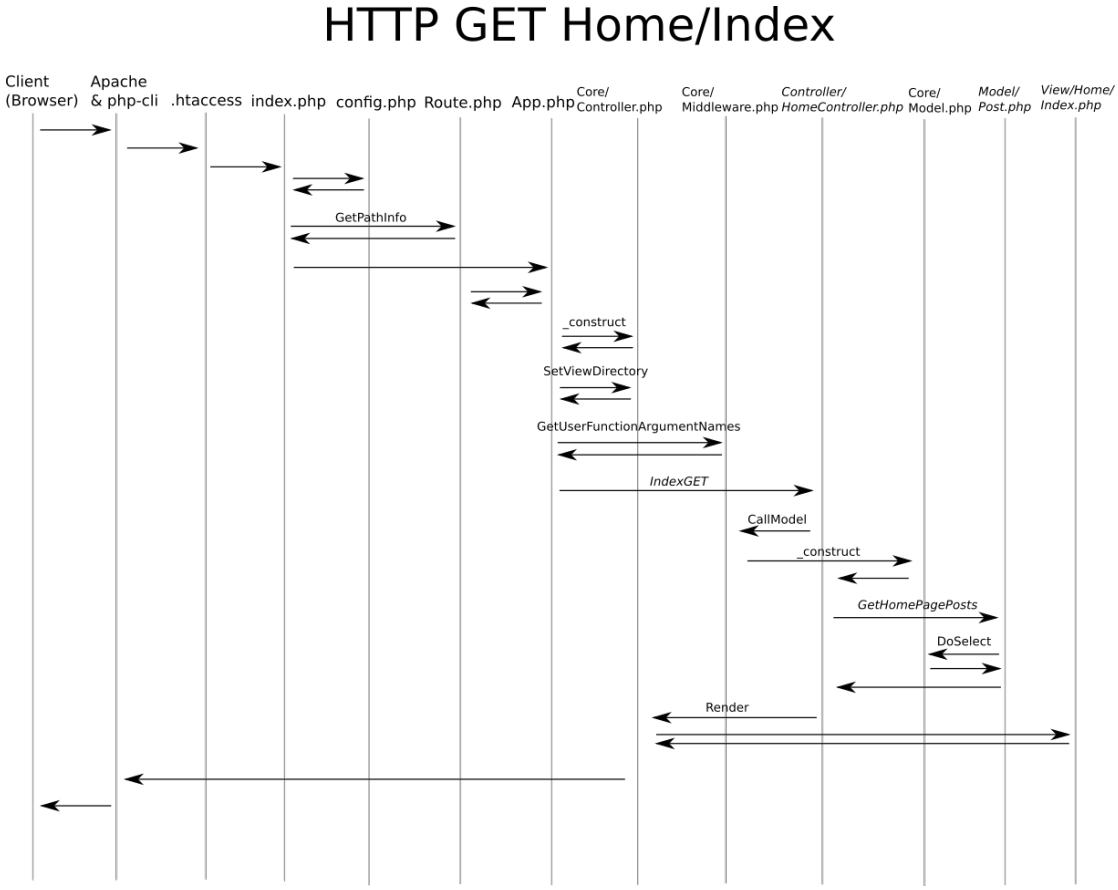
۱۵ قابلیت نرم‌افزاری (Feature)[۷]

در بخشی از تعریف‌های مربوط به مفهوم «توسعه‌ی قابلیت‌گرای نرم‌افزار» (Feature-Oriented Software Development – FOSD) به واحدی از عمل‌کرد برنامه که نیازی را مرتفع می‌کند، یک تصمیم طراحی را نمایندگی می‌کند، و یک قابلیت بلقوه‌ی پیکربندی را ارائه می‌کند، قابلیت نرم‌افزار گفته می‌شود.

۷ http://www.jot.fm/issues/issue_۲۰۰۹_۰۷/column۵

فصل سوم
شیوه کارکرد هر بخش

۱۶ چرخه‌ی حیات برنامه در یک درخواست



در این درخواست نمونه مشاهده می‌شود که هر بخش چه وظیفه‌ای را بر عهده دارد.

۱۷ ساختار پوشه‌بندی

حداقل فایل‌هایی که برای اجرای این پروژه مورد نیاز هستند در ساختار درختی زیر نمایش داده شده و همچنین در ادامه به شرح تفصیلی آن‌ها خواهیم پرداخت.

```
.
├── API
│   └── v1
│       ├── authController.php
│       └── postsController.php
├── Controller
│   ├── AuthenticationController.php
│   ├── HomeController.php
│   ├── MyController.CRUD.php
```

```
│   ├── MyController.FileManager.php
│   ├── MyController.People.php
│   ├── MyController.php
│   ├── PostController.php
│   └── RSSController.php
├── Core
│   ├── ApiApp.php
│   ├── ApiController.php
│   ├── App.php
│   ├── Auth.php
│   ├── Config.php
│   ├── Config.Sample.php
│   ├── Controller.php
│   ├── Exceptions.php
│   ├── Middleware.php
│   ├── Model.php
│   └── Route.php
├── dictionary.yaml
├── favicon.png
├── gui
│   ├── js
│   │   ├── Post.js
│   │   └── Posts.js
│   └── view
│       ├── Post.htm
│       └── Posts.htm
├── Helper
│   └── Welcome.php
├── index.php
├── Libs
│   ├── APR1.php
│   ├── Arrays.php
│   ├── Cryptography.php
│   ├── Email.php
│   ├── Exception.php
│   ├── filemanager
│   │   ├── filemanager.php
│   │   ├── LICENSE
│   │   ├── phpfm.png
│   │   └── README.md
│   ├── File.php
│   ├── jdf.php
│   ├── JSON.php
```

```
│   ├── Language.php
│   ├── Links.php
│   ├── ORM.php
│   ├── Parsedown.php
│   ├── Random.php
│   ├── Strings.php
│   ├── tiny-html-minifier.php
│   ├── Translate.php
│   └── TSQL.php
├── LICENCE
├── Model
│   ├── Authentication.php
│   ├── Comment.php
│   ├── File.php
│   ├── Person.php
│   ├── Post.php
│   ├── Statistics.php
│   ├── Todo.php
│   └── Visit.php
├── package.json
├── README.md
├── robots.txt
├── static
│   ├── css
│   │   ├── boostrap._variables.scss
│   │   ├── bootstrap.min.css
│   │   ├── ...
│   │   └── view.css
│   ├── errors
│   │   ├── HTTP400.html
│   │   ├── ...
│   │   ├── HTTP533.html
│   │   └── README.md
│   ├── fonts
│   │   ├── Sahel-Bold.ttf
│   │   ├── ...
│   │   └── Sahel.ttf
│   ├── img
│   │   ├── ...
│   │   └── abstract-background.jpg
│   ├── js
│   │   ├── bootstrap.js
│   │   ├── ...
```

```
|   |   └── view.js
|   ├── Logo.png
|   └── textyy
|       ├── core.css
|       ├── core.js
|       └── icons
├── TODO.md
└── View
    ├── Authentication
    |   └── Login.php
    ├── Home
    |   ├── Feed.php
    |   ├── Index.php
    |   ├── _Layout.php
    |   └── View.php
    ├── _Layout.php
    └── My
        ├── CRUD.php
        ├── Index.php
        ├── Interpreter.php
        ├── _Layout.php
        ├── People.php
        └── Person.php
```

۱۸ چرخه‌ی زندگی (Life Cycle)

اجرای هر برنامه‌ی کامپیوتری، توسط یک ماشه (Trigger) انجام می‌شود و ممکن است به پایان رسیده، یا در حلقه‌ای قرار بگیرد (مانند برنامه‌های مربوط به سخت‌افزار و یا سیستم‌عامل‌ها)، و یا دچار قفل مرگ (Dead lock) شود. برنامه‌های تحت وب، معمولاً با درخواست کاربر (Client Request) آغاز به کار می‌کنند و نهایتاً با ارائه‌ی پاسخ (Response) به پایان می‌رسند. پس می‌توان مراحلی که برنامه از درخواست تا پاسخ طی می‌کند را چرخه‌ی حیات آن دانست.

۱۹ فایل پیکربندی *.htaccess*

این فایل توسط کارسازهای وب مختلف نظیر Apache و IIS پشتیبانی می‌شود. وظیفه‌ی این فایل، پیکربندی شیوه‌ی رفتار کارساز وب با اسکریپت‌های مرتبط است. با استفاده از این فایل می‌توان تعیین کرد که چه قابلیت‌هایی که کارساز وب فعال و چه قابلیت‌هایی غیر فعال باشند.

نکته: در چهارچوب نرم‌افزاری این فایل وظیفه‌ی هدایت درخواست‌ها به فایل index.php را برعهده دارد.

۲۰ فایل پیکر بندی *Config.php*

این فایل که بنا است توسط توسعه‌دهندگان در مرحله‌ی نصب چهارچوب ویرایش شود، وظیفه‌ی تعریف و نگه‌داری متغیرهایی را دارد که برای شخصی‌سازی (Customization) مورد نیاز هستند. این متغیرها شامل اطلاعات ارتباط با پایگاه‌داده، ارتباط با سرور ایمیل، و آیا برنامه می‌تواند آمار درخواست‌های برنامه را ذخیره‌سازی کند یا خیر.

۲۱ فایل آغازین برنامه *index.php*

تمامی درخواست‌ها توسط فایل *.htaccess* به فایل *index.php* منتقل می‌شوند. این فایل سرآغاز چرخه‌ی زندگی برنامه است. وظیفه‌ی این اسکریپت کنترل کردن نوع و مبدأ درخواست‌هایی است که به برنامه ارسال می‌شود. بدیهی است که منظورمان از درخواست، درخواست‌های استاندارد اچ‌تی‌تی‌پی (HTTP) هستند. سپس این فایل چگونگی نمایش خطاهای برنامه را مدیریت می‌کند. کتاب‌خانه‌هایی را که برای اجرای توابع اصلی برنامه مورد نیاز است، فراخوانی می‌کند. سپس این برنامه نوع درخواست را از روی آدرس آن تشخیص می‌دهد و تصمیم می‌گیرد که کتاب‌خانه‌های مربوط به چه نوع کنترلر را اجرا کند.

۲۲ هسته (Core)

هسته‌ی این برنامه در پوشه‌ی Core قرار گرفته است. همانطور که در فایل *index.php* مشاهده می‌شود، فایل‌های مختلفی از این پوشه در مرحله‌ی آغازین، به صورت آماده‌به‌کار (included) قرار می‌گیرند.

۲۲.۱ هسته‌ی مسیریاب (*Route.php*)

در این فایل توابعی وجود دارد که پس از پالایش (Parse) آدرس، مسیر کنترلری را بر می‌گرداند که مسول اجرای آن درخواست کاربر است.

۲۲.۲ هسته‌ی میان‌افزار (*Middleware.php*)

این فایل توابعی را در بر می‌گیرد که وظیفه‌ی فراخوانی‌های میان‌افزاری مانند اجـرای سـاختن یـک نمونه از مدل درخواستی کنترلر را برعهده دارد.

۲۲.۳ هسته‌ی خطا (*Exceptions.php*)

این اسکریپت وظیفه‌ی معرفی دسته‌بندی‌های خطـا در کـد برنامـه را بـر عهـده دارد. از آن‌جـا کـه مدیریت کردن شیوه‌ی خطا در این برنامه به صورت سراسری انجام می‌شـود، لازم اسـت تـا خطاهـا دارای طبقه‌بندی باشند. در این فایل کلاس‌هایی تعریف شده‌اند که از کلاس *Exception* مربوط به هسته‌ی PHP ارث‌بری می‌کند.

۲۲.۴ هسته‌ی احراز هویت (*Auth.php*)

این فایل وظیفه‌ی انجام احـراز هـویت اساسـی برنامـه را بـر عهـده دارد. در حـالت پیش‌فـرض این چهارچوب، احراز هویت با استفاده از فایل *.htpasswd* انجام می‌شود. فلسفه‌ی احراز هویت با فایـل این است که اگر بنا بود تا چهارچوب بدون وابستگی به دی‌بی‌ام‌اس مورد استفاده قـرار گـیرد، خللی در اجرای برنامه نباشد.

۲۲.۴.۱ فایل *.htpasswd*

در این فایل در هر خط یک کاربر تعریف شده است. اولین کاراکتر دونقطـه (کـالن – Colon) جـدا کننده‌ی (seperator) نام‌کاربری از کلمه‌ی عبور است. کلمه‌ی عبـور بـا اسـتفاده از الگـوریتم APR کدگذاری شده است.

```
tayyebi:$apr1$/8/d03Mm$0Y26FBpPRoR1rN9C8nlTh.
```

این فایل را می‌توان با استفاده دستورات پیش‌فرض لینوکس ایجاد کرد. پارامتر اول بعد از سـوییچ *c* آدرس فایل است، و دومین پارامتر نام‌کاربری است.

```
htpasswd -c /var/www/html/Sariab-V2/.htpasswd tayyebi
```

۲۲.۴.۲ تابع احراز هویت

```
$this->CheckAuth(); // Check login
```

این تابع در صورت عدم اجرای موفق احراز هویت، خطای مربوطـه را برگردانـده و از اجـرای برنامـه جلوگیری می‌کند.

۲۲.۵ هسته‌ی کاربرد (*App.php*)

این فایل وظیفه‌ی اجرای یک نمونه از کنترلر مربوط به درخواست کاربر را برعهده دارد. همچنین این اسکریپت وظیفه‌ی مدیریت (Handle) خطای پیش آمده در اجرای کنترلر را برعهده دارد.

۲۲.۶ هسته‌ی کاربرد واسط برنامه‌نویسی کاربردی (*ApiApp.php*)

این هسته مشابه هسته‌ی کاربرد پیشین است که با توجه به اینکه واسط‌های برنامه‌نویسی احتیاج به ویو ندارند، آن را اجرا نمی‌کند.

۲۲.۷ هسته‌ی مدل (Model.php)

این هسته در تابع سازنده‌ی (Constructor function) خود، ارتباط با پایگاه‌داده را انجام می‌دهد. و سپس با استفاده از دو تابع DoSelect و DoQuery انجام گفت‌وگوهایی را با پایگاه‌داده انجام می‌دهد که به ترتیب، خروجی آن‌ها به صورت رکورد است و خروجی آن‌ها به صورت رکورد نیست.

۲۲.۸ هسته‌ی کنترلر (Controller.php)

کلاس والد کنترل‌های موجود در پوشه‌ی controller این فایل هسته است. این فایل وظیفه‌ی مدیریت کردن هلپرها، ویوها، و قالب‌ها را بر عهده دارد. این کلاس با استفاده از تابع پیش‌فرض _ *call* قابلیت فراخوانی ساده‌تر هلپرها را فراهم می‌کند. در تابع _*construct* وظیفه‌ی آمارگیری را انجام می‌دهد. تابع GetPayload ابتدا و انتهای فایل‌های قالب را جدا می‌کند تا بتوان آن‌ها را با استفاده از تابع Evaluate بهم متصل کرد. وظیفه‌ی ادغام فایل‌های قالب و ویوها را تابع Render انجام می‌دهد.

۲۲.۹ هسته‌ی کنترلر API (ApiController.php)

این فایل مشابه فایل *Controller.php* است. با این تفاوت که توابع مربوط به پردازش ویوها و قالب‌ها وجود ندارد و به ازای آن توابعی وجود دارد که تبدیل فرمت خروجی JSON را انجام می‌دهد.

۲۳ پوشه‌ی view

در این پوشه پوشه‌های دیگری هم‌نام با کنترلرها قرار می‌گیرد که دربرگیرنده‌ی ویوهای هم‌نام با اکشن‌ها هستند. اکشن‌ها توابعی در کنترل‌ها هستند که وظیفه‌ای خاص را بر عهده دارد. برنامه‌نویس این فایل‌ها را ایجاد می‌کند.

۲۴ پوشه‌ی controller

در این پوشه کنترلرهایی که توسط برنامه‌نویس آماده شده‌اند قرار می‌گیرد.

۲۵ پوشه‌ی API

در این پوشه پوشه‌هایی قرار می‌گیرد که نسخه‌ی API را تعریف می‌کند و هرکدام از آن پوشه‌ها شامل کنترلرهای API می‌شود.

۲۶ پوشه‌ی gui

این پوشه فایل‌های سمت کاربری را نگه می‌دارد که با APIها ارتباط می‌گیرد و واسط تحت وبی را می‌سازد که از نظر فنی همانند برنامه‌های تلفن همراه عمل می‌کند.

۲۷ هلپرها (Helper)

هلپرها کدهایی هستند که اجرای آن‌ها باید در لایه‌ای بین ویو و کنترلر صورت بگیرد. هلپرها معمولاً با مدل‌ها در ارتباط نیستند. کاربرد هلپرها می‌تواند در ایجاد بخش‌هایی پرتکرار بین ویوهای مختلف است. برای مقال منوهایی که در بخش‌های مختلف یک وب‌سایت تکرار می‌شوند را می‌توان به صورت هلپر تعریف کرد.

۲۸ قالب‌ها (Layout)

ویوهایی که در دسته‌بندی‌های مشترک قرار می‌گیرد، معمولاً از نظر بصری مشابهت‌هایی را دارند. برای این منظور قالب‌ها پیاده‌سازی می‌شوند تا از نوشتن پرتکرار کدها جلوگیری شود.

در چهارچوب نرم‌افزاری حاضر، بخش متغیر قالب‌ها، با استفاده از کد زیر مشخص می‌شود. به بخش بالاتر از این خط Preload و به خط پایین‌تر Payload می‌گوییم.

```
<!--VIEW_CONTENT-->
```

فصل چهارم
نصب، توسعه و راه‌اندازی

۲۹ نصب برنامه

برای نصب این برنامه به بستر کارساز وب و پایگاه‌داده نیاز است. این بستر می‌توانـد Apache، IIS، Nginx و یا سایر برنامه‌های مشابه باشد.

۳۰ پیکربندی چهارچوب نرم‌افزاری

برای پیکربندی این چهارچوب باید فایل *Config.php* را ویرایش کرد.

۳۱ پیاده‌سازی یک مدل (Model)

قالب استاندارد را می‌توان از مثال زیر دریافت کرد:

```
class File extends Model {

function GetFiles() {
$Query = 'SELECT `Id`, `FileName`, `Submit` FROM `Files` ORDER BY
`Id` DESC';
$Result = $this->DoSelect($Query);
return $Result;
}

function InsertNewFile($Values) {
$Query = 'INSERT INTO `Files` (`FileName`, `Submit`) VALUES
(:FileName, NOW())';
$Result = $this->DoQuery($Query, $Values);
return $Result;
}
function GetFile($Values) {
$Query = 'SELECT `Id`, `FileName`, `Submit` FROM `Files` WHERE
`Id`=:Id';
$Result = $this->DoSelect($Query, $Values);
return $Result;
}

function DeleteFile($Values) {
$Query = 'DELETE FROM `Files` WHERE `Id`=:Id';
$Result = $this->DoQuery($Query, $Values);
return $Result;
}

}
```

اما می‌تواند برای پیاده‌سازی سریع اعمال CRUD از مثال زیر استفاده کرد:

```
class Todo extends Model {

// === Based on a template ===
function DescribeTable() {
return $this->DoSelect("DESCRIBE `Todos`");
}
function GetAdminPanelItems($Values = null) {

$Query = 'SELECT
CONCAT(\'<a class="btn btn-sm btn-default" href="' . _Root .
'My/CRUD/Todo/\', id , \'">\', \'Edit\', \'</a>\') as Edit,
Id
,`Title`
,`Submit`
FROM `Todos`
ORDER BY `Id` ASC';

return $this->DoSelect($Query);
}
function GetItemByIdentifier($Values) {
$Query = "SELECT *
FROM `Todos`
WHERE `Id` = :Id
LIMIT 1";

return $this->DoSelect($Query, $Values);
}
}
```

۳۲ پیاده‌سازی یک ویو (View)

می‌توان از متغیر Data که در کنترلر به صورت پیش‌فرض تعریف می‌شود و توصیه می‌کنیم از این قاعده پیروی کنید، برای پیاده‌سازی بخش‌های مختلف ویو استفاده کرد. برای مثال:

```
<div class="card">
<div class="card-header">
تغییر کلمه‌ی عبور
</div>
<div class="card-body">
<h5 class="card-title">/<?php echo $Data['Model']['Username'] ?
>/</h5>
<form class="form" method="post" enctype="multipart/form-data">
```

```
<div class="form-group">
<label for="FormerPassInput">نام کاربری</label>
<input type="text" name="UsernameInput" id="UsernameInput"
readonly
class="form-control" value="<?php echo $Data['Model']['Username'] ?>"
/>
</div>
<div class="form-group">
<label for="FormerPassInput">کلمه‌ی عبور پیشین</label>
<input type="password" name="FormerPassInput" id="FormerPassInput"
class="form-control" placeholder="کلمه‌ی عبور پیشین" />
</div>
<div class="form-group">
<label for="NewPassInput">کلمه‌ی عبور پسین</label>
<input type="password" name="NewPassInput" id="NewPassInput"
required
class="form-control" placeholder="کلمه‌ی عبور جدید خود را وارد کنید" />
</div>
<div class="form-group">
<label for="ConfirmPassInput">تکرار مکررات</label>
<input type="password" name="ConfirmPassInput" id="ConfirmPassInput"
class="form-control" placeholder="تکرار کلمه‌ی عبور جدید خود را وارد کنید" />
</div>
<button type="submit" class="btn btn-primary" name="ChangePassword">
تغییر گذرواژه</button>
</form>
</div>
</div>
```

۳۳ پیاده‌سازی یک کنترلر (Controller)

می‌توان از مثال زیر برای پیاده‌سازی یک کنترلر استفاده کرد:

```
class People {

/**
* PeopleGET
*
* List the posts
*
* @return void
*/
function PeopleGET() {

$this->CheckLogin('admin'); // Check login
```

```php
// Ask database for data
$Model = $this->CallModel("Person");
$Rows = $Model->GetAllPeople();

// Package the response
$Data = [
'Title' => 'مدیریت کاربرها',
'Model' => $Rows
];
// Render the view
$this->Render('People', $Data);

}

/**
* PersonGET
*
* Get a Person details and managment stuff
*
* @return void
*/
function PersonGET($Id = 0) {

$this->CheckLogin(); // Check login

// Check if Id is passed to function
if ($Id != 0)
{
$Model = $this->CallModel("Person");
$Rows = $Model->GetPersonById([
'Id' => $Id,
]);

if (count($Rows) == 0)
goto the_notfound;

$Row = $Rows[0];

$Data = [
'Title' => 'مدیریت کاربر',
'Model' => $Row
];
$this->Render('Person', $Data);
```

```
}
// If it was insert
else
{
the_notfound: // A goto (evil) stuff // Just a programming fun.
throw new NotFoundException("شناسه‌ی کاربری مورد نظر پیدا نشد");
}
}

/**
* PersonPOST
*
* Update Person details
*
* @return void
*/
function PersonPOST($Id = 0) {
$this->CheckLogin(); // Check login

// Check if Id is passed to the function
if ($Id != 0)
{
// Call the model
$Model = $this->CallModel("Person");

// Initialy set message to String.Empty
$Message = '';

// Check if Person exist from parameters
$Rows = $Model->GetPersonById([
'Id' => $Id
]);
if (count($Rows) > 0)
{

// Check if new password and it's confirm match
if ($_POST['NewPassInput'] == $_POST['ConfirmPassInput'])
{

// Check former password
$Values = [
'Username' => $Rows[0]['Username'],
'Password' => $_POST['FormerPassInput']
```

```
];

// If former password was correct
if ((new Auth($this))->CheckLogin($Values, 'todo:notadmin!'))
{
// Update the password
$Response = $Model->UpdatePassword([
'Id' => $Id,
'Password' => (new Cryptography())->Encrypt($_POST['NewPassInput'])
]);

// Check for database errors
$Message = $Response ? 'خطای پایگاه داده' : 'کلمه‌ی عبور کاربر به روز شد';
}
else
{
$Message = 'کلمه‌ی عبور پیشین معتبر نیست';
}
}
else
{
$Message = 'کلمه‌ی عبور جدید با تکرار آن هم‌خوانی ندارد';
}

// Get the current record
$Row = $Rows[0];

// Return the view
$Data = [
'Title' => 'مدیریت کاربر',
'Message' => $Message,
'Model' => $Row
];
$this->Render('Person', $Data);

// Don't allow program to go to the next lines
return;
}
}

// If Person id did not exist in database or sent as paramter to this
method
throw new NotFoundException("شناسه‌ی کاربری مورد نظر پیدا نشد");
}
```

```
}
```

۳۴ پیاده‌سازی یک کنترلر (API Controller)

می‌توان از مثال زیر برای پیاده‌سازی یک کنترلر (API) استفاده کرد:

```
<?php

class postsController extends ApiController {
function GET($IdOrFrom = -1, $To = -1) {
$this->CheckLogin(); // Check login

// Select single post by id
if ((!is_array($IdOrFrom) && $IdOrFrom != -1) && $To == -1)
{
$Model = $this->CallModel("Post");
$Rows = $Model->GetPostById(
['Id' => $IdOrFrom]
);

if (count($Rows) == 0)
throw new NotFoundException('Post with passed Id not found.');

$Rows[0]['Title'] = utf8_decode($Rows[0]['Title']);
$Rows[0]['Body'] = utf8_decode($Rows[0]['Body']);

parent::SendResponse(200, $Rows);

}
// Select multiple posts with limitations
else
{

// Lazy load limitations
$From = 0;
if (!is_array($IdOrFrom) && $IdOrFrom>0)
$From = (int) trim($IdOrFrom);
if ($To == -1)
$To = 50;
else $To = (int) trim($To);

// Call the model
$Model = $this->CallModel("Post");
$Rows = $Model->GetAllPosts([
'From' => $From,
```

```
'To' => $To
]);

for ($i = 0 ; $i < count($Rows) ; $i++)
{
$Rows[$i]['Title'] = utf8_decode($Rows[$i]['Title']);
}

parent::SendResponse(200, $Rows);
}
}
function POST() {

$this->CheckLogin(); // Check login

$Values = [
'MasterId' => (new Random())::GenerateGUID(),
'Title' => $this->RequestBody['title'],
'Body' => $this->RequestBody['body'],
'PersonId' => 1,// $this->RequestBody['PersonId'], // TODO: Attention
'Status' => $this->RequestBody['status'],
'Language' => 'fa-IR' // $this->RequestBody['language'],
];

if (isset($this->RequestBody[0]['content'])) {
$Values['BinContent'] = base64_encode(file_get_contents($this-
>RequestBody[0]['content']['tmp_name'][0]));
} else {
$Values['BinContent'] = null;
}

$Model = $this->CallModel("Post");
$Model->InsertPost($Values);

parent::SendResponse(200, $this->RequestBody);

}
function PUT() {

$bincontent = null;
if (isset($this->RequestBody['content[]']))
// $bincontent = base64_encode(file_get_contents($this-
>RequestBody['content']['tmp_name'][0]));
$bincontent = base64_encode($this->RequestBody['content[]']);
```

```
$Values = [
'MasterId' =>$this->RequestBody['masterid'],
'Title' => $this->RequestBody['title'],
'BinContent' => $bincontent,
'Body' => $this->RequestBody['body'],
'PersonId' => 1,// $this->RequestBody['PersonId'], // TODO: Attention
'Status' => $this->RequestBody['status'],
'IsContentDeleted' => isset($this->RequestBody['deletecontent']),
'Language' => 'fa-IR' // $this->RequestBody['language'],
];

$Model = $this->CallModel("Post");
$Model->UpdatePost($Values);

parent::SendResponse(200, "Post '" . $this->RequestBody['masterid'] .
"' updated successfuly.");
}
function DELETE() {
$Values = [
'MasterId' =>$this->RequestBody['masterid'],
'Title' => $this->RequestBody['title'],
'Body' => $this->RequestBody['body'],
'PersonId' => 1,// $this->RequestBody['PersonId'], // TODO: Attention
'Language' => 'fa-IR' // $this->RequestBody['language'],
];

$Model = $this->CallModel("Post");
$Model->DeletePost($Values);

parent::SendResponse(200, "Post '" . $this->RequestBody['masterid'] .
"' deleted successfuly.");
}
}
```

فصل پنجم
جمع‌بندی و نتیجه‌گیری

۳۵ نتیجه‌گیری

استفاده از یک فریم‌ورک به شرط آگاهی از خواص پایگاه‌داده در حالت بهینه قرار می‌گیرد. این فریم‌ورک در برخی از پروژه‌های صنعت مورد استفاده قرار گرفته است و فرایند تولید نرم‌افزار را لذت بخش‌تر، و خروجی‌ها را به صورت کلی به برنامه‌های بهینه‌تری تبدیل کرده است.

۳۶ پیشنهاد

پیشنهاد می‌شود که بتوان با استفاده از Meta Programming امکان تولید برنامه با استفاده از واسط گرافیکی فراهم شود.

فصل ششم
پیوست

۳۷ کدها

i. .htaccess

```
# Enable authentication
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]

# Enable routing
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.+)$ index.php?/$1 [L]
```

ii. Config.php

```
/**
*
* App configuration
*
*/

// Application name
define('_AppName', 'SF2');
```

```
// Default URL (For redirects and etc.)
define('_Root', 'http://localhost/SF/');

// TODO: WARNING: Disable debug on production server
define('_Debug', false);

// To disable statistics, turn off the flag
define('_Statistics', false);

// The directory used by file manager to upload user files
define('_UploadDirectory', 'Uploads/');

// MySQL Server details
define('_DatabaseServer', 'localhost');
define('_DatabaseUsername', 'root');
define('_DatabasePassword', '');
define('_DatabaseName', 'SF2');

// API Result Type
define('_APIRESULTTYPE', 'application/json');

// Mail Server
define('_MailServer', '');
define('_MailUser', '');
define('_MailPassword', '');
```

iii. Index.php

```
// Read configuration
include('Core/Config.php');

// CORS
header('Access-Control-Allow-Origin: *');

// Allowed methods
header('Access-Control-Allow-Methods: GET, PUT, POST, DELETE, HEAD, VIEW');

// Debuf mode
if (_Debug)
{
// Report all PHP errors
ini_set('display_errors', '1');
ini_set('display_startup_errors', '1');
error_reporting(E_ALL);
}
else
// Turn off all error reporting
error_reporting(0);

// Exception handler
include('Core/Exceptions.php');
```

```
// Cryptography
include('Libs/Cryptography.php');

// Cryptography
include('Libs/APR1.php');

// Random
include('Libs/Random.php');

// Strings
include('Libs/Strings.php');

// Models core
include('Core/Model.php');

// Middleware
include('Core/Middleware.php');

// Jalali Date
include('Libs/jdf.php');

// Routing
include('Core/Route.php');

// Security
```

```
include('Core/Auth.php');

// Check if it's an MVC API request
if (count((new Route)::GetPathInfo()) > 0 &&
(new Route)::GetPathInfo()[0] == 'api')
{
// New JSON library to handle large arrays
include('Libs/JSON.php');

// Controllers core
include('Core/ApiController.php');

// Router
include('Core/ApiApp.php');

// Initialize
new ApiApp;
}
// If was a MVC request
else
{
// Markdown
include('Libs/Parsedown.php');

// Controllers core
include('Core/Controller.php');
```

```
// Router
include('Core/App.php');

// Initialize
new App;
}
```

iv. Route.php

در این فایل توابعی وجود دارد که پس از پالایش (Parse) آدرس، مسیر کنترلری را بر می‌گرداند که مسول اجرای آن درخواست کاربر است.

```
class Route {
/**
* GetPathInfo
*
* Returns URL requested by Person
*
* @return array
*/
static function GetPathInfo(){

$ActualLink = "http://$_SERVER[HTTP_HOST]$_SERVER[REQUEST_URI]";
// $ParsedUrl = parse_url($ActualLink);
```

```
// Use path info if exists
// index.php/value1/value2
if (isset($_SERVER['PATH_INFO']))
$PathInfo = trim(
$_SERVER['PATH_INFO']
, '/');
// Use query string if exists
// index.php?/value1/value2
else if (isset($_SERVER['QUERY_STRING']))
$PathInfo = trim(
$_SERVER['QUERY_STRING']
, '/');
// return null if no one exist
else
return [];
// convert to array and return
$Output = explode('/', $PathInfo);

// Add other query strings to the last parameter
if ($PathInfo == '')
return $Output;

$LeftOver = substr($ActualLink,
strpos($ActualLink, $PathInfo) + strlen($PathInfo));
if ($LeftOver != '' && str_replace($LeftOver, '', '/') != '')
array_push($Output, urldecode($LeftOver));
```

```
// return
return $Output;

// TODO: Allow dynamic routing

// === Returns:

// If api
// Index 0: 'api'
// Index 1: Version
// Index 2: Controller
// Else if not api
// Index 0: Controller
// Index 1: Action
}
}
```

v. Middleware.php

```
class Middleware {

/**
* CallModel
*
* Sets, calls, and loads the model
*
```

```
* @param string $Entity
*
* @return void
*/
static function CallModel($Entity, $UsePDO = true){
// include_once the model
include_once('Model/' . $Entity . '.php');
// if (!@include('Model/' . $Entity . '.php')) // Just a programming joke
// include('Model/' . $Entity . '.php');
// Create an instance of object
return new $Entity($UsePDO);
}

/**
* GetUserFunctionArgumentNames
*
* Gets attributes/paramters/arguments names
* of a function in runtime dynamically.
*
*
*/
static function GetUserFunctionArgumentNames($pair){
$ReflectClass = new ReflectionClass($pair[0]);
$ReflectMethod = $ReflectClass->getMethod($pair[1]);
$Paramters = $ReflectMethod->getParameters();
$ParamterNames = array();
```

```
foreach($Paramters as $Paramter)
{
array_push($ParamterNames, $Paramter->getName());
}
return $ParamterNames;
}
}
```

vi. Exceptions.php

```
// HTTP 401
class UnauthException extends Exception{ }
// HTTP 403
class AuthException extends Exception{ }
// HTTP 404
class NotFoundException extends Exception{ }
```

vii. Auth.php

```
class Auth {

protected $ParentController;

function __construct($ParentController) {
$this->ParentController = $ParentController;
}
```

```
/**
* CheckLogin
*
* Checks the Person role and login
*
* @param mixed $Data
* @param mixed $Role
*
* @return void
*/
function CheckLogin($Data, $Role = 'admin')
{

// If php_auth_user is denied on server and
// RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
// is enabled in .htaccess
// then manualy set the auth info
if (isset($_SERVER['HTTP_AUTHORIZATION'])
and $_SERVER['HTTP_AUTHORIZATION'] != '')
list($_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW']) =
explode(':', base64_decode(substr($_SERVER['HTTP_AUTHORIZATION'], 6)));

// Check admins with the .htpasswd file
if ($Role == 'admin')
{
```

```
// Read the passwords file
$Lines = array();
if ($file = fopen(".htpasswd", "r")) {
while(!feof($file)) {
array_push($Lines,fgets($file));
}
fclose($file);
}

// Break to lines to two dimensional array
$Credits = array_map(function($val) {
if ($val)
return explode(':', $val);
}, $Lines);
// Check if pasword is sent
if (!isset($Data['Username'])) {
a:
throw new UnauthException();
} else {
// Check passwords
foreach ($Credits as $Credit)
{
// Check username
if ($Data['Username'] != $Credit[0])
continue;
```

```
// Check plaintext password against an APR1-MD5 hash
$plain_text_passwd = $Data['Password'];
$check_result = APR1_MD5::check($plain_text_passwd, rtrim($Credit[1]));

// If not correct
if (!$check_result)
throw new UnauthException();

// If correct
return true;
}
// If failed
goto a;
}
}

// Check others with database
else
{

$Values = [
'Username' => $Data['Username'],
'Password' => (new Cryptography())->Encrypt($Data['Password'])
];

$Model = $this->ParentController->CallModel('Authentication');
```

```
$Entity = $Model->ValidatePersonPass($Values);

// TODO: Check sessions

return (count($Entity) == 1);
}
}

}
```

viii. App.php

این فایل وظیفه‌ی اجرای یک نمونه از کنترلر مربوط به درخواست کاربر را برعهده دارد. همچنین این اسکریپت وظیفه‌ی مدیریت (Handle) خطای پیش آمده در اجرای کنترلر را برعهده دارد.

```
class App
{

/**
*
* Values are default values
*
*/
private $Controller = 'HomeController';
private $DefaultViewDirectoryOfController = 'Home';
private $View = 'Index';
```

```php
private $Params = [];

/**
* HandleError
*
* Throw low level errors which are log-able from web server
*
* @return array
*/
function HandleError($HttpStatusCode, $Message = '')
{
switch ($HttpStatusCode)
{
case 401:
header('WWW-Authenticate: Basic realm="My Realm"');
header('HTTP/1.0 401 Unauthorized');
Controller::RedirectResponse(_Root . 'static/errors/HTTP401.html');
break;
case 403:
header('HTTP/1.0 403 Forbidden');
Controller::RedirectResponse(_Root . 'static/errors/HTTP403.html');
break;
case 404:
header('HTTP/1.0 404 Not Found');
// Controller::RedirectResponse(_Root . 'static/errors/HTTP404.html');
include('static/errors/HTTP404.html');
```

```
break;
default:
throw new Exception($Message);
die();
}
}

/**
* __construct
*
* Class Constructor which is called on any request
*
* @return void
*/
function __construct()
{
// Get URL
$URL = Route::GetPathInfo();

// Routing
// TODO: Use a routing mechanism which allows configuration
// Set Controller
if (isset($URL[0]) and $URL[0] != '')
{
// If it was reseved folders names
if ($URL[0] == 'static'
```

```
or $URL[0] == 'gui'
or $URL[0] == 'docs'
) return;
// Else
$this->DefaultViewDirectoryOfController = $URL[0];
$this->Controller = $URL[0] . 'Controller';
unset($URL[0]);
}
// Set View
if (isset($URL[1]) and $URL[1] != '')
{
$this->View = $URL[1];
unset($URL[1]);
}
// Define public variables
define('_Controller', $this->Controller);
define('_View', $this->View);
// Get other parameters
$this->Params = array_values($URL);
// Call the method form class
$ControllerFilePath = 'Controller/' . $this->Controller . '.php';
// If controller file does not exist
if (!file_exists($ControllerFilePath)) $this->HandleError(404);
// Include the controller file
include ($ControllerFilePath);
// Create an instance of controller class
```

```
$ClassObject = new $this->Controller();
// Set the view folder
$ClassObject->SetViewDirectory($this->DefaultViewDirectoryOfController);
// Get the method
$HttpMethod = $_SERVER['REQUEST_METHOD'];
// Find the function
$ControllerMethod = $this->View . $HttpMethod;
// Call the method if exists
if (!method_exists($ClassObject, $ControllerMethod)) $this->HandleError(404);

// Convert overloaded query strings to $_GET items
$MethodDefinedParamters = Middleware::GetUserFunctionArgumentNames([$ClassObject, $ControllerMethod]);
$sizeofPassedParams = sizeof($this->Params);
// Use path_info instead of 'overloaded query string' parameters
// Controller/Action/Param1Value/?Param2=Param2Value
// index.php/Controller/Action/Param1Value/?Param2=Param2Value
// index.php?/Controller/Action/Param1Value/?Param2=Param2Value
// In the three examples above, the Param2 is overloaded querystring
// Which it will be replaced by function paramter name for simpler
// programming of plugins and third-party software

// Check if there is a 'overloaded query string' included
// Logic: the last paramter must contain a question mark (?)
if ($sizeofPassedParams > 0)
if (strpos($this->Params[$sizeofPassedParams - 1], '?') !== false)
{
```

```
// Parse overloaded query strings
$KeyValuePairs = explode( '&' , // Delimiter
substr($this->Params[$sizeofPassedParams -1] // string
,strpos(
$this->Params[$sizeofPassedParams - 1]
, '?' // Begining of query string
) + 1 // start position
) // key pairs
); // key pairs array

// Clear the last paramter value from overloaded
// query strings with substring
$this->Params[$sizeofPassedParams - 1] =
substr($this->Params[$sizeofPassedParams - 1], 0,
strpos($this->Params[$sizeofPassedParams - 1], '?'));
// replace '/' with ''
$this->Params[$sizeofPassedParams - 1] =
str_replace('/', '', $this->Params[$sizeofPassedParams - 1]);

$i = 0;
foreach ($MethodDefinedParamters as $MethodDefinedParamter)
{
$i++; // loop counter
// Skip previously defined values of
// method paramters from passed parameters
```

```
if ($i < $sizeofPassedParams)
{
// if (strpos($KeyValuePairs[$i], '=') === false)
continue;
}
// then decied for other paramters
// and set their values from query strings
// NOTE: OVERLOADED QUERY STRING VALUES
// WHICH ARE PASSED BEFORE AS PATH_INFO
// WILL NOT BE REPLACED.
$found = false;
foreach ($KeyValuePairs as $KeyValue)
{
// If it's a valid keyval pair
// Logic: string contains equal mark (=)
if (strpos($KeyValue, '=') !== false)
{
if (explode('=', $KeyValue)[0] == $MethodDefinedParamter)
{
$found = true;
// Add the value to paramters
$this->Params[$i-1] = explode('=', $KeyValue)[1];
}
}
}
if (!$found)
```

```
{
$this->Params[$i-1] = '';
}
}
// If overloaded query string was passed
// but not mentioned as a function argument
// load it as $_GET.

foreach ($KeyValuePairs as $KeyValue)
{
$found = false;
foreach ($MethodDefinedParamters as $MethodDefinedParamter)
{
if (strpos($KeyValue, '=') !== false)
{
if (explode('=', $KeyValue)[0] == $MethodDefinedParamter)
$found = true;
}
else
{
if ($KeyValue == $MethodDefinedParamter)
$found = true;
}

}
if (!$found)
```

```
{
if (strpos($KeyValue, '=') !== false)
{
$_GET[explode('=', $KeyValue)[0]] = explode('=', $KeyValue)[1];
}
else
{
$_GET[$KeyValue] = 'true';
}
}
}
}

// Call the function
if (_Debug) call_user_func_array([$ClassObject, $ControllerMethod], $this->Params);
else try
{
// Call the view
call_user_func_array([$ClassObject, $ControllerMethod], $this->Params);
}
catch(AuthException $exp)
{ // On auth error
$this->HandleError(403);
}
catch(NotFoundException $exp)
{ // on not found error
```

```
$this->HandleError(404);
}
catch(UnauthException $exp)
{ // on unauthorized exception
$this->HandleError(401);
}
catch(UnauthException $exp)
{
// TODO:
}
}

}
```

ix. ApiApp.php

```
class ApiApp
{
protected $HttpStatus;
protected $Controller;
protected $Version;
function __construct(){

// Get URL
$URL = Route::GetPathInfo();

// Routing
```

```php
// Version
$this->Version = $URL[1];
// Controller
if (strpos($URL[2], '?') !== false)
{
$URL[2] = substr($URL[2], 0, strpos($URL[2], '?'));
// As $_GET is passed as keyvalue array
// which is not simple editable
$key = array_keys($_GET)[0];
$value = array_values($_GET)[0];
unset($_GET[$key]);
$key = substr($key, strpos($key, '?') + 1);
$_GET = array_reverse($_GET);
$_GET[$key] = $value;
$_GET = array_reverse($_GET);
}
$this->Controller = $URL[2].'Controller';
// Call the method form class
$ControllerFilePath = 'API/' . $this->Version . '/' . $this->Controller.'.php';
// Include the controller file
include($ControllerFilePath);
// Create an instance of controller class
$ClassObject = new $this->Controller();
// Set the method function
$ControllerMethod = $_SERVER['REQUEST_METHOD'];
// Set request body
```

```
switch ($ControllerMethod)
{
case "DELETE":
case "PUT":
$raw_data = file_get_contents('php://input');
$ClassObject->RequestBody = array();
$boundary = substr($raw_data, 0, strpos($raw_data, "\r\n"));
if ($boundary == null && $raw_data != 'null') // x-www-form-urlencoded
{
$split_parameters = explode('&', $raw_data);

for($i = 0; $i < count($split_parameters); $i++) {
$final_split = explode('=', $split_parameters[$i]);
$ClassObject->RequestBody[$final_split[0]] = $final_split[1];
}
}
else if ($raw_data != 'null')
{
$parts = array_slice(explode($boundary, $raw_data), 1);
foreach ($parts as $part) {
if ($part == "--\r\n") break;
$part = ltrim($part, "\r\n");
list($raw_headers, $body) = explode("\r\n\r\n", $part, 2);
$raw_headers = explode("\r\n", $raw_headers);
$headers = array();
foreach ($raw_headers as $header) {
```

```
list($name, $value) = explode(':', $header);
$headers[strtolower($name)] = ltrim($value, ' ');
}
if (isset($headers['content-disposition'])) {
$filename = null;
preg_match(
'/^(.+); *name="([^"]+)"(; *filename="([^"]+)")?/',
$headers['content-disposition'],
$matches
);
list(, $type, $name) = $matches;
isset($matches[4]) and $filename = $matches[4];
switch ($name) {
case 'Personfile':
file_put_contents($filename, $body);
break;
default:
$ClassObject->RequestBody[$name] = substr($body, 0, strlen($body) - 2);
break;
}
}

}
}
else
{
```

```
$url = $_SERVER['REQUEST_URI'];

$split_parameters = explode('&', $url);

for($i = 0; $i < count($split_parameters); $i++) {
$final_split = explode('=', $split_parameters[$i]);
$ClassObject->RequestBody[$final_split[0]] = $final_split[1];
}
}
break;

case "POST":
$ClassObject->RequestBody = $_POST;
array_push($ClassObject->RequestBody, $_FILES);
break;

default:
$ClassObject->RequestBody = $_GET;
array_push($ClassObject->RequestBody, $_FILES);

}
// Call the method if exists
if (!method_exists($ClassObject, $ControllerMethod))
$ClassObject->SendResponse(404,'Controller Method Not Found.');
// Set url passed paramters
unset($URL[0]);
```

```
unset($URL[1]);
unset($URL[2]);
$Params = array_values($URL);

// Call the function
if (_Debug) call_user_func_array([$ClassObject, $ControllerMethod], $Params);
else try {
// Call the method
call_user_func_array([$ClassObject, $ControllerMethod], $Params);
} catch (AuthException $exp ){ // On auth error
$ClassObject->SendResponse(401, $exp->getMessage());
} catch (NotFoundException $exp ){ // on not found error
$ClassObject->SendResponse(404, $exp->getMessage());
}

}
}
```

x. Model.php

```
class Model{

public static $Connection = '';

/**
* __toString
*
```

```
* Returns the connection stirng
*
* @return string ConnectionString
*/
public function __toString()
{
return 'mysql:host=' . _DatabaseServer . ';dbname=' . _DatabaseName;
}

/**
* __construct
*
* Create a connection to database
*
* @return void
*/
function __construct($PDO=true)
{
if ($PDO)
{
$ConnectionParameters = array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES latin1'); // utf8
self::$Connection = new PDO((string)$this, _DatabaseUsername, _DatabasePassword, $ConnectionParameters);
if (_Debug)
self::$Connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
else
```

```
{
self::$Connection = @new mysqli(_DatabaseServer
, _DatabaseUsername
, _DatabasePassword
, _DatabaseName);
mysqli_set_charset(self::$Connection,"latin1"); //utf8
}
}

/**
* DoSelect
*
* Runs a select query
*
* @param mixed $Query
* @param mixed $Values
* @param mixed $FetchStyle
*
* @return array Query outputs
*/
function DoSelect($Query, $Values = [], $FetchStyle = PDO::FETCH_ASSOC)
{
$LiveConnection = self::$Connection->prepare($Query);
foreach ($Values as $Key => $Value) {
if (gettype($Value) == "integer" || gettype($Value) == "boolean") // Recommended for bit(1) values
$LiveConnection->bindValue($Key, $Value, PDO::PARAM_INT);
```

```
else
$LiveConnection->bindValue($Key, $Value);
}
$LiveConnection->execute();
$Result = $LiveConnection->fetchAll($FetchStyle);
return $Result;
}

/**
* DoQuery
*
* Runs a executing query
*
* @param mixed $Query
* @param mixed $Values
*
* @return void
*/
function DoQuery($Query, $Values = [])
{
$LiveConnection = self::$Connection->prepare($Query);
foreach ($Values as $Key => $Value) {
if (gettype($Value) == "integer" || gettype($Value) == "boolean") // Recommended for bit(1) values
$LiveConnection->bindValue($Key, $Value, PDO::PARAM_INT);
else
$LiveConnection->bindValue($Key, $Value);
```

```
}
return $LiveConnection->execute();
}
}
```

xi. Controller.php

```
class Controller extends Middleware{

protected $ViewDirectory;

/**
* __call
*
* Dynamic functions
*
* Calls helpers and etc...
* with functions in style of
* call_XXXXXX
*
*
* @return void
*/
function __call($func, $params) {
$prefix = substr($func, 0, 5);
if ($prefix == 'call_')
{
```

```
$func = substr($func, 5);
foreach (glob("Helper/*.php") as $key=>$value)
{
if ($value == 'Helper/' . $func . '.php')
{
include('Helper/' . $func . '.php');
}
}
}
}

/**
* __construct
*
* Responsible for statistics
*
*
* @return void
*/
function __construct() {

// If system was configured to disable statistics
if (!_Statistics)
return;

// Call the model
```

```
$Model = $this->CallModel("Statistics");
// Bind values
$Values = [
"CONTEXT_DOCUMENT_ROOT" => isset($_SERVER["CONTEXT_DOCUMENT_ROOT"]) ? $_SERVER["CONTEXT_DOCUMENT_ROOT"] : "",
"CONTEXT_PREFIX" => isset($_SERVER["CONTEXT_PREFIX"]) ? $_SERVER["CONTEXT_PREFIX"] : "",
"DOCUMENT_ROOT" => isset($_SERVER["DOCUMENT_ROOT"]) ? $_SERVER["DOCUMENT_ROOT"] : "",
"GATEWAY_INTERFACE" => isset($_SERVER["GATEWAY_INTERFACE"]) ? $_SERVER["GATEWAY_INTERFACE"] : "",
"HTTP_ACCEPT" => isset($_SERVER["HTTP_ACCEPT"]) ? $_SERVER["HTTP_ACCEPT"] : "",
"HTTP_ACCEPT_ENCODING" => isset($_SERVER["HTTP_ACCEPT_ENCODING"]) ? $_SERVER["HTTP_ACCEPT_ENCODING"] : "",
"HTTP_ACCEPT_LANGUAGE" => isset($_SERVER["HTTP_ACCEPT_LANGUAGE"]) ? $_SERVER["HTTP_ACCEPT_LANGUAGE"] : "",
"HTTP_CACHE_CONTROL" => isset($_SERVER["HTTP_CACHE_CONTROL"]) ? $_SERVER["HTTP_CACHE_CONTROL"] : "",
"HTTP_CONNECTION" => isset($_SERVER["HTTP_CONNECTION"]) ? $_SERVER["HTTP_CONNECTION"] : "",
"HTTP_COOKIE" => isset($_SERVER["HTTP_COOKIE"]) ? $_SERVER["HTTP_COOKIE"] : "",
"HTTP_HOST" => isset($_SERVER["HTTP_HOST"]) ? $_SERVER["HTTP_HOST"] : "",
"HTTP_REFERER" => isset($_SERVER["HTTP_REFERER"]) ? $_SERVER["HTTP_REFERER"] : "",
"HTTP_SEC_FETCH_DEST" => isset($_SERVER["HTTP_SEC_FETCH_DEST"]) ? $_SERVER["HTTP_SEC_FETCH_DEST"] : "",
"HTTP_SEC_FETCH_MODE" => isset($_SERVER["HTTP_SEC_FETCH_MODE"]) ? $_SERVER["HTTP_SEC_FETCH_MODE"] : "",
"HTTP_SEC_FETCH_SITE" => isset($_SERVER["HTTP_SEC_FETCH_SITE"]) ? $_SERVER["HTTP_SEC_FETCH_SITE"] : "",
"HTTP_SEC_FETCH_Person" => isset($_SERVER["HTTP_SEC_FETCH_Person"]) ? $_SERVER["HTTP_SEC_FETCH_Person"] : "",
"HTTP_UPGRADE_INSECURE_REQUESTS" => isset($_SERVER["HTTP_UPGRADE_INSECURE_REQUESTS"]) ?
$_SERVER["HTTP_UPGRADE_INSECURE_REQUESTS"] : "",
"HTTP_Person_AGENT" => isset($_SERVER["HTTP_Person_AGENT"]) ? $_SERVER["HTTP_Person_AGENT"] : "",
"PATH" => isset($_SERVER["PATH"]) ? $_SERVER["PATH"] : "",
"PATH_INFO" => isset($_SERVER["PATH_INFO"]) ? $_SERVER["PATH_INFO"] : "",
"PATH_TRANSLATED" => isset($_SERVER["PATH_TRANSLATED"]) ? $_SERVER["PATH_TRANSLATED"] : "",
"PHP_SELF" => isset($_SERVER["PHP_SELF"]) ? $_SERVER["PHP_SELF"] : "",
```

```
"QUERY_STRING" => isset($_SERVER["QUERY_STRING"]) ? $_SERVER["QUERY_STRING"] : "",
"REDIRECT_STATUS" => isset($_SERVER["REDIRECT_STATUS"]) ? $_SERVER["REDIRECT_STATUS"] : "",
"REDIRECT_URL" => isset($_SERVER["REDIRECT_URL"]) ? $_SERVER["REDIRECT_URL"] : "",
"REMOTE_ADDR" => isset($_SERVER["REMOTE_ADDR"]) ? $_SERVER["REMOTE_ADDR"] : "",
"REMOTE_PORT" => isset($_SERVER["REMOTE_PORT"]) ? $_SERVER["REMOTE_PORT"] : "",
"REQUEST_METHOD" => isset($_SERVER["REQUEST_METHOD"]) ? $_SERVER["REQUEST_METHOD"] : "",
"REQUEST_SCHEME" => isset($_SERVER["REQUEST_SCHEME"]) ? $_SERVER["REQUEST_SCHEME"] : "",
"REQUEST_TIME" => isset($_SERVER["REQUEST_TIME"]) ? $_SERVER["REQUEST_TIME"] : "",
"REQUEST_TIME_FLOAT" => isset($_SERVER["REQUEST_TIME_FLOAT"]) ? $_SERVER["REQUEST_TIME_FLOAT"] : "",
"REQUEST_URI" => isset($_SERVER["REQUEST_URI"]) ? $_SERVER["REQUEST_URI"] : "",
"SCRIPT_FILENAME" => isset($_SERVER["SCRIPT_FILENAME"]) ? $_SERVER["SCRIPT_FILENAME"] : "",
"SCRIPT_NAME" => isset($_SERVER["SCRIPT_NAME"]) ? $_SERVER["SCRIPT_NAME"] : "",
"SERVER_ADDR" => isset($_SERVER["SERVER_ADDR"]) ? $_SERVER["SERVER_ADDR"] : "",
"SERVER_ADMIN" => isset($_SERVER["SERVER_ADMIN"]) ? $_SERVER["SERVER_ADMIN"] : "",
"SERVER_NAME" => isset($_SERVER["SERVER_NAME"]) ? $_SERVER["SERVER_NAME"] : "",
"SERVER_PORT" => isset($_SERVER["SERVER_PORT"]) ? $_SERVER["SERVER_PORT"] : "",
"SERVER_PROTOCOL" => isset($_SERVER["SERVER_PROTOCOL"]) ? $_SERVER["SERVER_PROTOCOL"] : "",
"SERVER_SIGNATURE" => isset($_SERVER["SERVER_SIGNATURE"]) ? $_SERVER["SERVER_SIGNATURE"] : "",
"SERVER_SOFTWARE" => isset($_SERVER["SERVER_SOFTWARE"]) ? $_SERVER["SERVER_SOFTWARE"] : "",
// "HTTP_CLIENT_IP" => isset($_SERVER["HTTP_CLIENT_IP"]) ? $_SERVER["HTTP_CLIENT_IP"] : "",
// "HTTP_X_FORWARDED_FOR" => isset($_SERVER["HTTP_X_FORWARDED_FOR"]) ? $_SERVER["HTTP_X_FORWARDED_FOR"] : "",
];

// Insert rows
$Rows = $Model->InsertVisit($Values);
}
```

```
/**
* SetViewDirectory
*
* A simple setter for $ViewDirectory
* which we call it from App.php
*
* @param mixed $Value
*
* @return void
*/
function SetViewDirectory(string $Value){
$this->ViewDirectory = $Value;
}

/**
*
* GetPayload
*
* Extract head and tail from file
*
*/
private function GetPayload($ViewFile, $ExcludePayload = false){
$Separator = '<!--PAYLOAD_CONTENT_END-->';
$TextInsideFile = file_get_contents($ViewFile);
// If text does not contains the payload pointer
```

```
if (strpos($TextInsideFile, $Separator) == false)
if ($ExcludePayload)
return $TextInsideFile;
else return "";
// If head
if (!$ExcludePayload and (strpos($TextInsideFile, $Separator) !== false))
$TextInsideFile = substr($TextInsideFile, 0, strpos($TextInsideFile, $Separator));
// If separator does not exist
else if (!$ExcludePayload and !(strpos($TextInsideFile, $Separator) !== false))
$TextInsideFile = '';
// If tail
else
$TextInsideFile = substr($TextInsideFile, strpos($TextInsideFile, $Separator) + strlen($Separator));
return $TextInsideFile;
}

/**
* RenderBody
*
* Returns header and footer of the layout file
*
* @param mixed $LayoutFile
* @param mixed $Head
*
* @return void
*/
```

```
private function RenderBody($LayoutFile, $Head){
$Separator = '<!--VIEW_CONTENT-->';
$TextInsideFile = file_get_contents($LayoutFile);
// If head
if ($Head and (strpos($TextInsideFile, $Separator) !== false) )
$TextInsideFile = substr($TextInsideFile, 0, strpos($TextInsideFile, $Separator));
// If separator does not exist
else if ($Head and !(strpos($TextInsideFile, $Separator) !== false))
$TextInsideFile = '';
// If tail
else
$TextInsideFile = substr($TextInsideFile, strpos($TextInsideFile, $Separator) + strlen($Separator));
return $TextInsideFile;
}

/**
* Evaluate
*
* An `include`-based equivalent to php eval code
*
* @param mixed $Code
*
* @return void
*/
private function Evaluate($Code, $Data = []) {
```

```
// Algorithm
// 1. Create a temp file from payload
// 2. Include the payload
$TempPointer = tmpfile();
fwrite($TempPointer, $Code);
$MetaData = stream_get_meta_data($TempPointer);
$TempFileName = $MetaData['uri'];
chmod($TempFileName, 775);
include($TempFileName);
fclose($TempPointer);
}

/**
* Render
*
* Renders the view
*
* @param mixed $View
* @param mixed $Data
*
* @return void
*/
function Render($View, $Data = [], $IsPartial = false)
{
// The current view file
$CurrentViewFile = 'View/' . $this->ViewDirectory . '/' . $View . '.php';
```

```
// Run the payload for current file
$this->Evaluate($this->GetPayload($CurrentViewFile, false), $Data);
// Get master layout head
if (!$IsPartial)
$this->Evaluate($this->RenderBody('View/_Layout.php', true), $Data);
// Get slave layout head
if (!$IsPartial)
if (file_exists('View/' . $this->ViewDirectory . '/_Layout.php'))
$this->Evaluate($this->RenderBody('View/' . $this->ViewDirectory . '/_Layout.php', true), $Data);
// Render the view body
$this->Evaluate($this->GetPayload($CurrentViewFile, true), $Data);
// Get slave layout tail
if (!$IsPartial)
if (file_exists('View/' . $this->ViewDirectory . '/_Layout.php'))
$this->Evaluate($this->RenderBody('View/' . $this->ViewDirectory . '/_Layout.php', false), $Data);
// Get master layout tail
if (!$IsPartial)
$this->Evaluate($this->RenderBody('View/_Layout.php', false), $Data);

}

/**
* RedirectResponse
*
* Sets the header to redirect
*
```

```
* @param mixed $Route
*
* @return void
*/
function RedirectResponse($Route)
{
header("Location: " . $Route);
}


/**
* CheckLogin
*
* Check the auth for Person
* Automatially detects where to find username and password
*
* @param mixed $Role
*
* @return void
*/
function CheckLogin($Role = 'admin')
{
// If values not set
if (isset($_SERVER['PHP_AUTH_USER']))
{
// Get values from HTTP Authenticate
```

```
$Values = [
'Username' => $_SERVER['PHP_AUTH_USER'],
'Password' => $_SERVER['PHP_AUTH_PW']
];
// Check with DB
if (!(new Auth($this))->CheckLogin($Values, $Role))
throw new AuthException('Invalid Login.');
else return true;
}
else if (isset($_COOKIE['Username']))
{
// Get values from cookies
$Values = [
'Username' => $_COOKIE['Username'],
'Password' => $_COOKIE['Password']
];
// TODO: check with token instead of password
// Check with DB
if (!(new Auth($this))->CheckLogin($Values, $Role))
throw new AuthException('Invalid Login.');
else return true;
}
else
throw new AuthException('Login Required.');
return false;
}
```

```
}
```

xii. ApiController.php

```
class ApiApp
{
protected $HttpStatus;
protected $Controller;
protected $Version;
function __construct(){

// Get URL
$URL = Route::GetPathInfo();

// Routing
// Version
$this->Version = $URL[1];
// Controller
if (strpos($URL[2], '?') !== false)
{
$URL[2] = substr($URL[2], 0, strpos($URL[2], '?'));
// As $_GET is passed as keyvalue array
// which is not simple editable
$key = array_keys($_GET)[0];
$value = array_values($_GET)[0];
unset($_GET[$key]);
$key = substr($key, strpos($key, '?') + 1);
```

```
$_GET = array_reverse($_GET);
$_GET[$key] = $value;
$_GET = array_reverse($_GET);
}
$this->Controller = $URL[2].'Controller';
// Call the method form class
$ControllerFilePath = 'API/' . $this->Version . '/' . $this->Controller.'.php';
// Include the controller file
include($ControllerFilePath);
// Create an instance of controller class
$ClassObject = new $this->Controller();
// Set the method function
$ControllerMethod = $_SERVER['REQUEST_METHOD'];
// Set request body
switch ($ControllerMethod)
{
case "DELETE":
case "PUT":
$raw_data = file_get_contents('php://input');
$ClassObject->RequestBody = array();
$boundary = substr($raw_data, 0, strpos($raw_data, "\r\n"));
if ($boundary == null && $raw_data != 'null') // x-www-form-urlencoded
{
$split_parameters = explode('&', $raw_data);

for($i = 0; $i < count($split_parameters); $i++) {
```

```
$final_split = explode('=', $split_parameters[$i]);
$ClassObject->RequestBody[$final_split[0]] = $final_split[1];
}
}
else if ($raw_data != 'null')
{
$parts = array_slice(explode($boundary, $raw_data), 1);
foreach ($parts as $part) {
if ($part == "--\r\n") break;
$part = ltrim($part, "\r\n");
list($raw_headers, $body) = explode("\r\n\r\n", $part, 2);
$raw_headers = explode("\r\n", $raw_headers);
$headers = array();
foreach ($raw_headers as $header) {
list($name, $value) = explode(':', $header);
$headers[strtolower($name)] = ltrim($value, ' ');
}
if (isset($headers['content-disposition'])) {
$filename = null;
preg_match(
'/^(.+); *name="([^"]+)"(; *filename="([^"]+)")?/',
$headers['content-disposition'],
$matches
);
list(, $type, $name) = $matches;
isset($matches[4]) and $filename = $matches[4];
```

```
switch ($name) {
case 'Personfile':
file_put_contents($filename, $body);
break;
default:
$ClassObject->RequestBody[$name] = substr($body, 0, strlen($body) - 2);
break;
}
}

}
}
else
{
$url = $_SERVER['REQUEST_URI'];

$split_parameters = explode('&', $url);

for($i = 0; $i < count($split_parameters); $i++) {
$final_split = explode('=', $split_parameters[$i]);
$ClassObject->RequestBody[$final_split[0]] = $final_split[1];
}
}
break;

case "POST":
```

```php
$ClassObject->RequestBody = $_POST;
array_push($ClassObject->RequestBody, $_FILES);
break;

case "GET":
$Params = array_values($URL);
$sizeofPassedParams = sizeof($Params);
// Check if there is a 'overloaded query string' included
// Then add it to $RequestBody
// Logic: the last paramter must contain a question mark (?)
if ($sizeofPassedParams > 0)
if (strpos($Params[$sizeofPassedParams - 1], '?') !== false)
{

// Parse overloaded query strings
$KeyValuePairs = explode( '&' , // Delimiter
substr($Params[$sizeofPassedParams -1] // string
,strpos(
$Params[$sizeofPassedParams - 1]
, '?' // Begining of query string
) + 1 // start position
) // key pairs
); // key pairs array

foreach ($KeyValuePairs as $KeyValue)
{
```

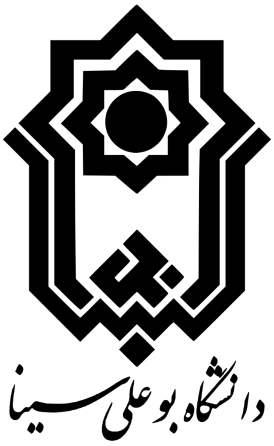
```
// If it's a valid keyval pair
// Logic: string contains equal mark (=)
if (strpos($KeyValue, '=') !== false)
{
$Exploded = explode('=', $KeyValue);
$ClassObject->RequestBody[$Exploded[0]] = $Exploded[1];
}
}
}
array_push($ClassObject->RequestBody, $_GET);
array_push($ClassObject->RequestBody, $_FILES);

}
// Call the method if exists
if (!method_exists($ClassObject, $ControllerMethod))
$ClassObject->SendResponse(404,'Controller Method Not Found.');
// Set url passed paramters
unset($URL[0]);
unset($URL[1]);
unset($URL[2]);
$Params = array_values($URL);

// Call the function
if (_Debug) call_user_func_array([$ClassObject, $ControllerMethod], $Params);
else try {
// Call the method
```

```
call_user_func_array([$ClassObject, $ControllerMethod], $Params);
} catch (AuthException $exp ){ // On auth error
$ClassObject->SendResponse(401, $exp->getMessage());
} catch (NotFoundException $exp ){ // on not found error
$ClassObject->SendResponse(404, $exp->getMessage());
}

}
}
```



Bu-Ali Sina University
Tuyserkan Faculty of Engineering
Department of Computer Engineering

B. Sc. Thesis

title
Implementation of back-end software framework

author
Mohammad R. Tayyebi

supervisor
Mohammad H. Bamneshin

date ۲۰۲۰