FRAMEWORKLARNING FARQLARI AVZALLIKLARI VA KAMCHILIKLARI

Farg'ona davlat unversiteti
Axborot texnologiyalari kafedrasi katta o'qituvchisi
Mamatqodirov Mahammadali
Farg'ona davlat universiteti 3-kurs talabasi
Rahmatjonova Marjona
rahmatjonovamarjona7@gmail.com

Annotatsiya: shbu maqola zamonaviy dasturiy ta'minot ishlab chiqish sohasida keng qo'llanilayotgan framework tushunchasi, uning vazifalari, tuzilishi va turli toifadagi frameworklar o'rtasidagi konseptual hamda funksional farqlarni chuqur tahlil qiladi. Frameworklar - bu dasturchilarga takrorlanuvchi kodlarni yozmasdan turib, tizimli, modullashtirilgan va tezkor dasturiy mahsulot yaratish imkonini beruvchi tayyor dasturlash muhitlaridir. Ushbu maqolada frontend, backend, mobil va krossplatformali frameworklarning ichki arxitekturasi, ishlash mexanizmi va qo'llanilish kontekstidagi asosiy farqlari mukammal tarzda tahlil qilinadi.Frameworklardan foydalanish dasturchilar uchun bir qator afzalliklarni, jumladan, ishlab chiqish tezligini oshirish, kodning yagona struktura asosida yuritilishini ta'minlash, xavfsizlik va texnik xizmat ko'rsatishdagi yengilliklarni taqdim etadi. Shu bilan birga, frameworklar bilan ishlashda vujudga keladigan ba'zi salbiy jihatlar – masalan, o'rganishdagi murakkablik, texnik qarzning to'planishi, mustaqillik cheklovlari va moslashuvchanlikning kamayishi – ham maqolada keng yoritiladi.

Tadqiqotning asosiy maqsadi - turli frameworklarni tanlashda ishlab chiquvchilar duch keladigan muhim texnik va metodologik omillarni ochib berish, shuningdek, dasturiy mahsulot sifatini ta'minlashda optimal framework tanlash bo'yicha ilmiy asoslangan yondashuvni ishlab chiqishdan iborat. Maqola amaliyotchi dasturchilar, dasturiy ta'minot muhandislari hamda IT sohasida ilmiy izlanish olib borayotgan tadqiqotchilar uchun nazariy va metodik ahamiyatga ega bo'lgan tahliliy material sifatida xizmat qiladi.

Kalit so'zlar: framework, dasturlash karkasi, dasturiy ta'minot ishlab chiqish, frontend, backend, mobil frameworklar, krossplatformali frameworklar, arxitektura, dasturiy mahsulot sifati, texnik qarz, dasturiy ta'minot xavfsizligi

Аннотация: Данная статья глубоко анализирует понятие framework широко применяемое в современной разработке программного обеспечения, его функции, структуру и концептуальные и функциональные различия между различными типами фреймворков. Framework - это готовая программная среда, которая позволяет разработчикам создавать системные, модульные и быстрые программные продукты без необходимости многократного написания повторяющегося кода. В

статье детально рассматриваются внутренняя архитектура, механизмы работы и основные различия в контексте применения фронтенд-, бэкенд-, мобильных и кроссплатформенных фреймворков.

Использование фреймворков предоставляет разработчикам ряд преимуществ, включая повышение скорости разработки, обеспечение единообразной структуры кода, облегчение безопасности и технического обслуживания. В то же время в статье подробно освещаются и некоторые недостатки работы с фреймворками - сложности в обучении, накопление технического долга, ограничения свободы и снижение гибкости.

Основная цель исследования - выявить ключевые технические и методологические факторы, с которыми сталкиваются разработчики при выборе различных фреймворков, а также разработать научно обоснованный подход к оптимальному выбору фреймворка для обеспечения качества программного продукта. Статья служит аналитическим материалом теоретического и методологического характера для практикующих разработчиков, инженеров программного обеспечения и исследователей в области ИТ.

Ключевые слова: фреймворк, каркас программирования, разработка программного обеспечения, фронтенд, бэкенд, мобильные фреймворки, кроссплатформенные фреймворки, архитектура, качество программного продукта, технический долг, безопасность программного обеспечения

Annotation: This article provides an in-depth analysis of the concept of a framework, widely used in modern software development, its functions, structure, and the conceptual and functional differences among various types of frameworks. Frameworks are ready-made programming environments that enable developers to create systematic, modular, and efficient software products without repeatedly writing the same code. The article thoroughly examines the internal architecture, operational mechanisms, and main distinctions in the application context of frontend, backend, mobile, and cross-platform frameworks.

Using frameworks offers developers a range of advantages, including accelerated development speed, maintaining uniform code structure, and facilitating security and maintenance. At the same time, the article also highlights some drawbacks of working with frameworks - learning complexity, accumulation of technical debt, restrictions on independence, and reduced flexibility.

The main goal of the study is to identify key technical and methodological factors that developers face when selecting different frameworks and to develop a scientifically based approach to optimal framework selection to ensure software product quality. The article serves as an analytical resource with theoretical and methodological significance for practicing developers, software engineers, and IT researchers.

Keywords: framework, programming framework, software development, frontend, backend, mobile frameworks, cross-platform frameworks, architecture, software product quality, technical debt, software security

Zamonaviy dasturiy ta’minot ishlab chiqish jarayonlari tobora murakkablashib borayotgani sababli, ishlab chiquvchilar samarali va tezkor vositalarga muhtojlik sezmoqda. Shu munosabat bilan, frameworklar (dasturlash karkaslari) dasturiy mahsulotlarni yaratish jarayonini soddalashtirish, kodni qayta foydalanish imkoniyatini oshirish va sifatni kafolatlashda muhim ahamiyat kasb etmoqda. Frameworklar - bu takrorlanuvchi vazifalarni avtomatlashtirish va dasturiy ta’minot arxitekturasini standartlashtirishga qaratilgan tayyor dasturlash muhitlaridir.Frameworklar nafaqat dastur ishlab chiqish jarayonini tezlashtiradi, balki ishlab chiqilgan kod bazasining uzluksizligi va texnik xizmat ko‘rsatishning yengilligi uchun ham poydevor yaratadi. Shu bilan birga, har bir turdagi framework o‘zining arxitekturasi, ishlash mexanizmi va qo‘llanilish doirasi bo‘yicha farqlanadi, bu esa ishlab chiquvchilardan to‘g‘ri tanlov qilishni talab qiladi.

Frameworklarning farqlari ularning maqsadi, ichki arxitekturasi, qo‘llanilish sohalari va texnologik yondashuvidan kelib chiqadi. Eng ko‘p uchraydigan turlari quyidagilardir:

Frontend frameworklari: Bu frameworklar foydalanuvchi interfeyslarini yaratishda ishlatiladi. Masalan, React, Angular, Vue.js kabi frameworklar asosan komponentlar asosida ishlaydi, foydalanuvchi tajribasini yaxshilashga qaratilgan. Ularning asosiy farqi - DOM manipulyatsiyasi, data binding, virtual DOM va komponentlarni boshqarish usullari.

Backend frameworklari: Server tomonidagi biznes logikasini amalga oshirish uchun mo‘ljallangan. Django, Laravel, Spring kabi backend frameworklar MVC (Model-View-Controller), MVVM kabi arxitektura uslublarini qo‘llaydi. Ularning farqi asosan ma’lumotlar bazasi bilan ishlash, autentifikatsiya, marshrutlash va server logikasini boshqarishda ko‘zga tashlanadi.

Mobil frameworklar: Bu frameworklar mobil ilovalar yaratishda qo‘llaniladi. Swift (iOS), Kotlin (Android) kabi natif frameworklar bilan bir qatorda, React Native, Flutter kabi krossplatformali frameworklar ham mavjud. Natif frameworklar platformaga mos va yuqori tezlikda ishlaydi, krossplatformali frameworklar esa kodni qayta ishlatish imkonini beradi, ammo ba’zida performans jihatdan orqada qoladi.

Krossplatformali frameworklar: Turli platformalarda (mobil, veb, desktop) ishlay oladigan ilovalarni yaratishga mo‘ljallangan. Ular bir nechta platformalar uchun yagona kod bazasidan foydalanishni ta’minlaydi.

Frameworklarning ichki arxitekturasi va ishlash mexanizmi ham ularni ajratib turadigan muhim omillardir. Misol uchun, React virtual DOM asosida ishlaydi, bu esa tezkor interfeys yangilanishini ta’minlaydi. Django esa o‘zining kuchli ORM tizimi va ko‘p funksiyali admin paneli bilan ajralib turadi. Flutter esa o‘zining grafika-rendering

tizimi orqali yuqori sifatli interfeys yaratadi. Bunday ichki arxitektura farqlari frameworkning samaradorligi, moslashuvchanligi va qo‘llanilish sohasini belgilaydi.

Frameworklar ishlatiladigan dasturlash tili va platformaga bog‘liq holda ham farq qiladi. Masalan, JavaScript asosidagi frontend frameworklari brauzerda ishlaydi, backend uchun ko‘proq Python, PHP, Java, C# kabi tillar qo‘llaniladi. Mobil uchun esa Swift, Kotlin, Dart (Flutter) asosiy tillar hisoblanadi.

Frameworklar dasturiy ta’minot ishlab chiqish jarayonini sezilarli darajada tezlashtiradi va samaradorligini oshiradi. Ular dasturchilarga oldindan tayyorlangan funksiyalar, komponentlar va standart kod strukturasini taqdim etib, asosiy biznes logikasiga e'tibor qaratish imkonini beradi. Shu orqali takrorlanuvchi kodlarni yozishdan qochib, loyiha yaratish vaqti va xarajatlarini kamaytiradi. Frameworklar yordamida modul va komponentlar qayta ishlatilishi mumkin bo‘lib, bu loyiha murakkabligi oshganida ham samarali natijalar beradi.

Shuningdek, frameworklar kod sifatining yaxshilanishiga va standartlashtirilishiga xizmat qiladi. Ular ko‘pincha ma’lum arxitektura va kodlash qoidalarini o‘z ichiga oladi, bu esa loyiha kodining bir xilda bo‘lishini ta’minlaydi. Kodni o‘qish, tushunish va qo‘llab-quvvatlash osonlashadi, ayniqsa jamoaviy ishlarda. Koddagi xatoliklar kamayadi, chunki ko‘plab funksiyalar va kod strukturalari framework tomonidan nazorat qilinadi va avtomatlashtiriladi.

Xavfsizlik sohasida ham frameworklar katta yordam beradi. Ular ko‘plab keng tarqalgan zaifliklarga qarshi himoya vositalarini standart ravishda taqdim etadi. Masalan, SQL injeksiyalari, XSS va CSRF hujumlaridan himoya qilish mexanizmlari frameworklar tarkibida mavjud. Bu dasturchilarga qo‘shimcha xavfsizlik kodlarini yozmasdan yuqori darajada himoyalangan dasturlar yaratish imkonini beradi. Natijada dasturiy mahsulot ishonchliligi oshadi va xavfsizlik xavflari kamayadi.

Frameworklarning yana bir muhim jihati — modullik va kengaytirilish imkoniyati. Ular modullar va pluginlar yordamida qurilgan bo‘lib, dasturchilar yangi funksiyalarni oddiy qo‘shimchalar orqali loyihaga kiritishlari mumkin. Bu loyiha rivojlanishini osonlashtiradi va yangi talablar yuzaga kelganda moslashuvchanlikni oshiradi. Shu bilan birga, ko‘plab komponentlar boshqa loyihalarda ham qayta ishlatilishi mumkin, bu esa ishlab chiqish samaradorligini yanada oshiradi.

Mashhur frameworklar katta va faol hamjamiyatga ega bo‘lib, ular dasturchilar uchun doimiy yordam, yangiliklar va hujjatlar taqdim etadi. Hujjatlar va o‘quv materiallarining ko‘pligi yangi foydalanuvchilar uchun o‘rganish jarayonini soddalashtiradi. Frameworklar muntazam yangilanishi tufayli barqaror va zamonaviy texnologiyalar bilan uyg‘un ishlaydi.

Biroq, frameworklardan foydalanish ba’zi qiyinchiliklarni ham yuzaga keltiradi. Ularning murakkabligi yangi boshlovchilar uchun o‘rganish jarayonini qiyinlashtiradi.

Ba’zi frameworklar ko‘plab konseptlar va arxitektura qatlamlariga ega bo‘lib, ularni to‘liq o‘zlashtirish uchun ko‘p vaqt va resurslar talab etiladi.

Frameworklarning yangilanishi va versiyalar orasidagi farqlar texnik qarzga olib kelishi mumkin. Eski loyihalarni yangi framework versiyasiga ko‘chirish murakkab jarayon bo‘lib, kodni moslashtirish va sinovdan o‘tkazish ko‘p vaqtni oladi. Bu esa loyihaning rivojlanish tempini sekinlashtiradi.

Frameworklar o‘zlarining qat’iy arxitekturasi va qoidalari bilan dasturchilarning erkinligini cheklashi mumkin. Ba’zan innovatsion yoki noan’anaviy yechimlarni amalga oshirish uchun framework imkoniyatlari yetarli bo‘lmasligi, yoki bunday funksiyalarni qo‘shish qiyinligi sababli loyiha moslashuvchanligi pasayadi.

Shuningdek, frameworklarning abstraktsiya qatlamlari ba’zan tizimning ishlash tezligiga salbiy ta’sir ko‘rsatadi. Masalan, krossplatformali mobil frameworklar yoki murakkab frontend frameworklar native ilovalarga nisbatan ko‘proq resurs sarflashi mumkin. Bu yuqori talab qiluvchi real vaqtdagi ilovalar uchun muammolarni keltirib chiqaradi.

Yana bir muammo - frameworkga qaramlik (vendor lock-in). Loyiha frameworkning o‘ziga xos arxitekturasi va APIlariga bog‘liq bo‘lib qoladi, bu esa kelajakda boshqa texnologiyaga o‘tishni qiyinlashtiradi. Natijada loyiha texnologik mustaqillikni yo‘qotadi.

Xulosa qilib aytganda, frameworklar dasturiy ta’minot ishlab chiqish jarayonini sezilarli darajada tezlashtiradi va sifatini yaxshilaydi. Ular dasturchilarga standart kod strukturasi, xavfsizlik va modul funksiyalarini taqdim etib, ishlab chiqish samaradorligini oshiradi hamda xatoliklarni kamaytiradi. Shu bilan birga, frameworklarning murakkabligi, o‘rganish qiyinligi, texnik qarz va platformaga qaramlik kabi kamchiliklari ham mavjud.Shuning uchun framework tanlashda loyiha ehtiyojlari, jamoaning malakasi va uzoq muddatli qo‘llab-quvvatlash imkoniyatlari hisobga olinishi lozim. To‘g‘ri tanlangan framework dasturiy mahsulot sifatini ta’minlashda va rivojlantirishda muhim vosita bo‘ladi.Umuman olganda, frameworklar dasturiy ta’minot ishlab chiqishni yanada samarali, xavfsiz va barqaror qilishga yordam beradi, ammo ularning kamchiliklari ham e’tibordan chetda qolmasligi kerak.

Foydalanilgan adabiyotlar:

1.Ro‘ziyev R.A. Dasturlash asoslari // O‘quv qo‘llanma.Toshkent, 2020. – 159 b.

2.Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley Professional.

3.Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.