دستور العمل پیام commit

v1.0

فهرست مطالب

1. مقدمه
2. چرا Semantic Commit Messages؟
3. انواع تغییرات
4. ساختار پیغام کامیت
5. مثال‌ها
6. نکات و توصیه‌ها
7. ابزارهای مفید
8. خلاصه

مقدمه

Semantic Commit Messages یک قاعده استاندارد برای نوشتن پیغام‌های کامیت است. در ادامه به توضیحات کامل این روش پرداخته شده است.

چرا Semantic Commit Messages؟

- خوانایی: ساختار استاندارد برای تمامی تغییرات.
- پیوند به تسک‌ها: ارتباط مستقیم با تسک‌های مرتبط در JIRA.
- تولید مستندات: تولید خودکار مستندات تغییرات از طریق ابزارهای خاص.

انواع تغییرات

- feat: افزودن ویژگی جدید.
- fix: اصلاح باگ یا مشکل.
- chore: تغییرات عملیاتی و نگهداری.
- docs: تغییرات مستندات.
- style: تغییرات استیل کد بدون تغییر عملکرد.
- refactor: بازسازی کد بدون افزودن ویژگی یا اصلاح باگ.
- perf: تغییرات بهبود عملکرد.
- test: افزودن یا تغییر تست‌ها.
- build: تغییرات مرتبط با ساخت و ساز پروژه.
- ci: تغییرات مرتبط با فرایند توسعه مداوم (Continuous Integration).

ساختار پیغام کامیت

فرمت کلی پیغام کامیت به شکل زیر است:

Format: <type>(<scope>): <subject> (JIRA-ID)

مثال‌ها

- ویژگی جدید:
  - feat(login): Add login with Facebook (JIRA-123)
- اصلاح باگ:
  - fix(cart): Fix discount calculation (JIRA-456)
- تغییر مستندات:
  - docs(api): Update REST API documentation (JIRA-789)

نکات و توصیه‌ها

- فعل حال ساده: استفاده از فعل حال ساده.
- یکپارچگی: نگه‌داری یک ساختار واحد در تمام پروژه.

ابزارهای مفید

- Jira Smart Commits: اتوماسیون ارتباط با JIRA.
- Commitlint: اعتبارسنجی پیغام‌های کامیت در مرحله Commit.

خلاصه

Semantic Commit Messages ابزار قدرتمندی برای ارتقاء فرآیند توسعه نرم‌افزار است. با افزودن شناسه تسک JIRA، این سیستم توانمندی‌های بیشتری برای ارتباط بین کد و تسک‌های مدیریت پروژه ارائه می‌دهد.

برای اطمینان از اینکه توسعه‌دهندگان از ساختار درست پیام کامیت استفاده می‌کنند، می‌توانید از ابزارهایی استفاده کنید که قوانینی را اجباری می‌کنند که باید در پیام‌های کامیت رعایت شوند. یکی از روش‌های مؤثر برای این کار استفاده از Git Hooks می‌باشد، به خصوص از نوع commit-msg هوک.

راه‌اندازی هوک commit-msg:

1. ایجاد یک فایل هوک در مخزن محلی:
   در مسیر .git/hooks داخل مخزن خود، یک فایل به نام commit-msg ایجاد کنید.
2. نوشتن اسکریپت بررسی:
   در این فایل، اسکریپتی بنویسید که پیام کامیت را بررسی کند. می‌توانید از زبان‌های مختلفی مانند Bash یا Python استفاده کنید. مثلاً با Bash:

```
#!/bin/sh
commit_regex='^(feat|fix|chore|docs|style|refactor|perf|test): A-Z.+ (#0-9+)$'
commit_message=$(cat "$1")
if ! echo "$commit_message" | grep -E "$commit_regex" > /dev/null; then
echo "Your commit message does not match the pattern!"
exit 1
fi
```

1. الگوی بالا برای پیام‌های کامیتی است که با تگ‌های مخصوصی شروع می‌شوند و شامل یک شماره کار (مانند شناسه JIRA) هستند.
2. قابل اجرا کردن فایل:
از طریق ترمینال، اجازه اجرای فایل را فعال کنید:

```
chmod +x .git/hooks/commit-msg
```

استفاده از ابزارهای خارجی:

همچنین ابزارهایی مانند Commitizen وجود دارد که به صورت خودکار می‌توانند فرآیند ایجاد پیام‌های کامیت را راهنمایی کنند و ساختار درست را اجباری کنند.

خلاصه

با استفاده از هوک‌های Git یا ابزارهای خارجی، می‌توانید فرآیند ایجاد پیام‌های کامیت را اتوماتیک کنید و اطمینان حاصل کنید که تمام توسعه‌دهندگان از ساختار مشخصی پیروی می‌کنند. این روش باعث افزایش انسجام و قابل‌فهم بودن تاریخچه کامیت‌ها در پروژه می‌شود.