**«** **Microsoft Azure DevOps (Session6)»**

**معاونت** توسعه و برند

واحد نرم‌افزار

****

**فهـرســت**

[**مقدمه 2**](#_Toc64471080)

[**بررسی WIKI 2**](#_Toc64471081)

[**ایجاد WIKI 4**](#_Toc64471082)

[**بررسی مفهوم Query Result 5**](#_Toc64471083)

[**معمولا می توان Wiki را در سه لایه ایجاد نمود 5**](#_Toc64471084)

[**مبحث REPO 6**](#_Toc64471085)

[**فرآیند انتقال Git Repository به Repo در Azure 7**](#_Toc64471086)



مقدمه

در جلسه قبل Backlog‌های ایجاد شده در Board، متناسب با نیاز سازمان و پروژه‌ها مورد بررسی قرار گرفت. یعنی ما ساختار کلی را به صورت زیر تعریف نمودیم و این ساختار مورد بررسی قرار گرفت:

Product

Epic

(1…n) Feature

(1…n)Pbi

(1…n)Task

به طور عمومی Pbi‌ها باید به Pbi‌های ریزتر شکسته شوند و ریز مسائل در Acceptance Criteria مطرح گردد. همچنین Effort‌ها نیز به Effort‌های ریزتر (مثلا 5، 3 و ...) تبدیل گردند.

در گام‌های بعدی این سؤال مطرح می‌شود که اولویت‌بندی این موارد به چه صورت می‌تواند باشد؟ به عنوان مثال یک اولویت‌بندی به نام CDP وجود دارد که اولویت‌بندی براساس هزینه می‌باشدکه مثلا این آیتم چند درصد برای سازمان مفید است و ... . این مورد حداکثر چیزی است که می توان اولویت‌بندی براساس هزینه و درآمد را کنترل نمود، که این Effort‌ها را می‌توان براساس آن‌ها تنظیم کرد تا سازمان بیشترین سود را ببرد.



بررسی WIKI

در این جلسه می‌خواهیم مفهوم Wiki را مورد بررسی قرار دهیم. می‌خواهیم بررسی کنیم که Wiki چگونه می‌تواند به سازمان کمک کند. به طور کلی Board‌ها خوب هستند ولی زمانی که شما می‌خواهید یک دید کلی نسبت به پروژه‌ها یا یک صفحه Landing داشته باشید، اسکرام این دید کلی را به شما نمی دهد که بتوانید نقطه شروعی داشته باشید و به کسی که تازه وارد سازمان شده است خط دهید و بفهمانید که از کجا اطلاعات پروژه را چک کند.

به عنوان مثال تصور کنید که یک پروژه سازمان Out Source شده و هیچ Bug و ... در Azure برای آن ثبت نشده است و از طرف دیگر هیچ شناسنامه و سندی هم ندارد. حال برای این مواقع دو کار زیر پیشنهاد می‌شود:

1. اگر قرار است به شخص جدیدی این پروژه را تحویل دهید برای آن که این روند ادامه پیدا نکند، شما باید ساختار را تغییر دهید، سازمان باید Agile کار کند و محصول یک Backlog‌یی پیدا کند.
2. از طرف دیگر هم در Knowledge Management برای آن شناسنامه‌ای درست کنید و توضیحاتی (ویس، ویدیو، متن و ...) تهیه نمایید. در قدیم از Share Point به این منظور استفاده می نمودند ولی Microsoft آن را از Azure جدا نمود.

در اصل Wiki‌ها یک ساختار درختی دارند، برای نمونه به سایت Azure DevOps Docs بروید و نمونه‌ای را مشاهده نمایید. درواقع محصول ما یک محصول نرم‌افزاری است که در Wiki در مورد آن صحبت می‌کنیم.

از Wiki استفاده می‌نمایند که مشخص کنند که به عنوان مثال یک توسعه‌دهنده چه وظایفی برعهده دارد؟ چگونه باید کار کند؟ در چه شرایطی کار شما را قبول می‌کنند؟ و ...

Docs GitLab نمونه جالبی برای Wiki می‌باشد، آن را بررسی نمایید.

به ازای هر Team Project تنها یک Wiki وجود دارد و Wiki متعلق به یک محصول خاص یا یک تیم اسکرام نمی‌باشد. پس به ازای هر Team Project یک Wiki خواهیم داشت.

مبحث دیگری که اهمیت دارد دانستن این مورد است که در پشت ‌Wikiها یک Git Repository قرار دارد و درنتیجه تمام قوانین Git Repository را شامل می‌شود و هر تغییری که در آن ایجاد می‌کنید مانند آن است که یک Commit و Push را انجام داده‌اید و این دو با هم اتفاق می‌افتند.

کلیک روی ... => کلیک روی Clone Wiki => مشخص شدن مسیر Git Repository

این Git Repository کنار بقیه Git Repository‌ها قابل مشاهده نیست و Hidden می‌باشد مگر آن که آدرس آن را در مرورگر وارد نمایید. Wiki حتی می‌تواند Branch داشته باشد. برای مرتب‌سازی در اولین قدم ما به یک ساختار نیازمندیم.

Wiki به صورت Mark Down کار می‌کند که معمولا در فایل های README.md این ساختار قابل مشاهده است.

برای تعیین ساختار یک سازمان در Wiki می‌توان مواردی مثل بحث تحقیق و توسعه، واحد فروش، نرم افزارهای مختلف، آموزش و ... را دسته بندی نمود.



ایجاد WIKI

به ازای هر پروژه ای که دارید یک صفحه در Wiki ایجاد کنید و به صورت کلی سناریو و گام‌های عملیاتی و مراحل مختلف پروژه را شرح دهید و یک ساختار درختی برای درک بهتر پروژه ایجاد نمایید و به Work Item‌های مربوطه رفرنس دهید.

در Wiki می‌توانید از تمام Tag‌های HTML و استایل‌های inline استفاده نمایید. لازم است نحوه کار با Mark Down را بدانید. مثلا نحوه ایجاد TOC و ... .

اگر Team Project به ازای سازمان باشد، صفحه اصلی را به شرح کلی پروژه می‌پردازیم و اگر به ازای یک پروژه یا Software Dept باشد، توضیحات آن را می‌نویسیم و پس از آن یک Over View مشخص کرده و در آن Subject‌ها (مثلا آموزش و ...) را مشخص می‌کنیم.

در ادامه ساختار درختی را برای Wiki ایجاد نمایید. حال اگر وارد Git Repository آن Wiki شوید، شکل گرفتن ساختار درختی آن Wiki را مشاهده خواهید نمود. ولی بعضی از آیتم‌ها هم فولدر و هم فایل .md را دارا می‌باشند و بعضی دیگر فقط فایل .md را دارا می‌باشند و این مورد به این دلیل است که آیتم‌هایی که Child دارند پس حتما می‌بایست فولدری برای نگهداری Child آن‌ها وجود داشته باشد.

View Revision تعداد نسخه‌ها و تغییراتی که در هر صفحه وجود دارد را نشان می‌دهد. درنتیجه شما تمام نسخه‌های Wiki را دارید.

****

* **#:** برای Work Item‌ها می باشد.
* **@:** برای مشخص کردن اشخاص استفاده می‌شود. پس از آن یک ایمیل برای فرد ارسال می‌شود.
* **!:** برای Pull Request‌هایی که هنوز در Wiki اعمال نشده‌اند.
* به طور خلاصه در Wiki محصول را شرح داده و جزئیات را بیان می‌نماییم و به Work Item مربوطه Assign می‌کنیم.



بررسی مفهوم Query Result

حال کمی فراتر می‌رویم و در Wiki روی ... کلیک کرده و Query Result ایجاد می‌نماییم. درواقع Query Result نتایج یک Query را به ما نشان می‌دهد.



معمولا می توان Wiki را در سه لایه ایجاد نمود

1. **لایه اول:** که به شما خط می دهد و به صورت کلی در مورد محصول صحبت می کند ( بیان Details از وظایف این لایه نمی باشد.)
2. **لایه دوم (OnBoardin):** این لایه کمی وارد Detail می‌شود و در مورد جزئیات محصول صحبت می‌کند.
3. **لایه سوم:** در این لایه Option‌ها و ماژول‌های استفاده شده در محصول را تعریف می‌نماییم و درمورد آن‌ها توضیح می‌دهیم.

در Wiki برای نظم کلی عکس‌ها و ... بهتر است عکس‌ها را که در پوشه‌ای به نام Attachments قرار دارند، پوشه‌بندی نمایید. اگر شما عکس‌ها را از طریق Drag and Drop داخل Wiki قرار دهید، به صورت نامرتبی قرار می‌گیرند. بنابراین آن‌ها را در Attachments پوشه‌بندی کرده و ساختار منظمی را برای آن‌ها ایجاد نمایید و درنهایت از آن آدرس برای نمایش عکس در Wiki استفاده کنید.

Wiki در یک لحظه می‌تواند Branch فعال را نشان دهد و قادر به نمایش چندین Branch با یکدیگر نمی‌باشد.

مسئله دیگری در Cross-Repo-Policy وجود دارد که باید به آن توجه داشت: یکسری از Policy‌ها را می‌توان روی کل Repository اعمال نمود که این موارد روی کل Git Repository اعمال می‌شود، درنتیجه اگر شما روی Cross-Repo-Policy یک Policy ایجاد نمایید، در آن صورت متأسفانه گاهی موارد در Wiki درگیر آن Policy هم می‌شوید که خطا می‌دهد که این Branch را به صورت دستی نمی توانید Update نمایید و باید Pull Request بزنید که این Wiki را برای شما Update کند.



مبحث REPO

ترتیبی وجود دارد که ابتدا Wiki محصول را مشخص کرده و سپس Board آن محصول را مشخص می‌نماییم و در مرحله سوم که Repo می‌باشد می‌خواهیم برای آن محصول کد بنویسیم. نکته‌ای که وجود دارد متصل کردن Repo و فرآیندهای داخل آن به بخش Board است. یعنی ما چنان چه کدی می‌نویسیم، آن کد مربوط به کدام Task است یا Branch مربوط به کدام Pbi، Technical Dept و ... می‌باشد.

Repo برای برطرف کردن نیازها شش آیتم دارد که پیچیده‌ترین آن‌ها بحث Pull Request و Branch‌ها می‌باشد.

1. **Files:** وظیفه آن نشان دادن این مورد است که چه فایل‌هایی وجود دارد و روی چه Branch‌هایی کار می‌کنیم و یا آن Branch چه فایل‌هایی دارد، History آن را نمایش می‌دهد و ... .
2. برای نامگذاری Repository‌ها سعی کنید نام آن‌ها چند سیلابه باشد. این مورد در جستجو کمک می‌نماید. به عنوان مثال:
3. **Product- اسم محصول-زیرسیستم آن**
4. **Commits**
5. **Pushes**
6. **Branches:** Branch را باید محدود استفاده نمایید و Tag‌ها را باید بیشتر استفاده کنید. درواقع به جای نگه‌داشتن Branch باید Tag ایجاد نمایید. Branch و Pull Request به هم وابسته‌اند و Pull Request چیزی جز Merge دو Branch نمی‌باشد.
7. **Tags:** مثل یک پرچم است و بسیار حائز اهمیت می‌باشد. ما از کجا بدانیم تا کدام Commit ما ورژن 1 بوده است؟ برای همین منظور از Tag‌ها استفاده می‌نماییم.
8. **Pull Requests**

دو فایل مهم که هر Repository در ابتدا باید داشته باشند:

* **فایل README.md:** شناسنامه محصول در Git Repository می‌باشد.
* **.gitignore:** مشخص می‌کند چه فایل‌هایی نباید در Git Repository نمایش داده شوند.



فرآیند انتقال Git Repository به Repo در Azure

ابتدا از Gogs یک Repository را روی سیستم خود Clone نمایید و سپس توسط Command زیر Origin آن مشخص می شود:

git remote –v

حال خیلی راحت می توان Origin آن را عوض کرد و به Azure نسبت داد. برای این منظور از Command زیر استفاده نمایید. فقط نیاز است که Git Repo در Azure خالی باشد و History نداشته باشد. در ابتدا توسط Command زیر Remoteجاری را حذف نمایید:

git remote remove origin

حال گزینه دوم را از Git Repo کپی کنید و در Command قرار دهید تا Git Repository به Azure منتقل شود. همچنین شما می‌توانید درون خود Azure این عملیات را انجام داده و Git Repository خود را Import نمایید که گاهی این مورد جواب نمی‌دهد.

Azure امکانی که دارد و هر ابزار دیگری این امکان را دارا نمی‌باشد، گزینه Fork می‌باشد. Fork درواقع مثل حالت یک کپی از پروژه جاری در Repository دیگر می‌باشد. به عنوان مثال می‌توانید از یک Git Repo (پروژه Open Source) یک Fork تهیه نمایید و روی یک قسمت آن کار کنید و نتیجه را بدون نیاز به Git Repo و ... به اشتراک بگذارید.

در Azure اگر بخشی از اسم فایل را هم بدانید و سرچ کنید به شما آن فایل را نشان می‌دهد.

همچنین می توان در Azure در فایل README و ... روی یک خط Comment ایجاد نمود. برای این منظور وقتی موس را روی متون حرکت می‌دهید یک علامت + کنار آن ایجاد می‌شود، روی آن کلیک کرده و یک شخص یا Work Item را به آن Assign نمایید.

Repo از نوع SVN را برای استفاده در Azure باید Convert نمایید. پیشنهاد می‌شود که آن را به git تبدیل کنید و Convert را هم انجام ندهید تا در آینده دچار مشکل نشوید. در Azure به طور کلی TFVC و Git وجود دارد .