DJANGO VERSION MIGRATION AND PACKAGE UPGRADE GUIDELINES

Prepared by	:	ANAND RAJ B
Date	:	02-04-2024
Version	:	1.0.0
Organisation	:	Venzo technologies pvt ltd
Contact	:	anandrajb@venzotechnologies.com

***** This document helps in version migration for django projects and I believe in pushing the boundaries of what's possible, reaching new heights, and setting standards . Feel free to adjust the document based on your needs .

1 . INTRODUCTION

The objective of this document is to provide a comprehensive set of guidelines for migrating or upgrading existing Django projects to newer versions or transitioning to new projects entirely.

Whether you're working on a **legacy codebase** or starting a greenfield project, adhering to these guidelines will ensure a smooth migration and release process and considering minimised disruptions and maximising compatibility with the latest Django releases.

By addressing potential compatibility issues proactively leverage the latest features , improvements and best practices .

1.1 Scope

The ultimate aim is to establish a consistent approach to project migration, covering best practices for the following areas:

- Project setup and configuration
- Security considerations
- Code maintenance
- Handling deprecated features and refactoring
- Upgrade planning and execution
- Engagement and knowledge sharing

1 . PREPARING FOR VERSION MIGRATION

Before embarking on the process of migrating or upgrading your Django project, it's crucial to take the necessary preparatory steps. This section outlines several key points to consider:

1. Access Current Project Status
    a. Review existing code base , dependencies and third party libraries status .
    b. Identify any custom **modifications** or extensions that have been implemented.
    c. Know what **python version** the code base is developed and look for possibilities to change or improve it .
2. Evaluate Target Version
    a. Make sure to choose the target version as a **LTS** (Long term support) Version . Check here
    b. Identify potential compatibility issues and plan for necessary code **refactoring** (if needed) .
3. Create DB migration Plan
    a. Define a nice comprehensive migration plan
    b. Check for any third party fields added and possibility for upgrading it .
    c. Adhire and stick a **Rollback mechanism** method .
4. Setup and Backup
    a. Create separate development setup environment variables with testing DB on migration .
    b. **Backup** the existing code and database to prevent data loss . check here
5. Communicate with stakeholders
    a. Inform to stakeholder or managers , clients , end-users about the upgrading migrations .
    b. Ensure to send a scheduled **maintenance mode** mail to the end-users during the new version deployment time.

2 . UPGRADING AND UPDATING PACKAGES

Here is some well defined steps to follow on the upgrading process

1. Update django
    a. Add necessary schemas and indexes
    b. Follow with LTS based migration
    c. Before doing DB migrations make sure to do a **fake migration** check here .
2. Handling deprecations and upgrading third party dependencies .
    a. If the existing package is deprecated for the new version of django , make sure to look for some other alternative package with high download and good release history below within **6 - 10 months** .
    b. Update any incompatible packages to the latest compatible versions .
3. Update python (if necessary)
    a. Check if the target django version requires a new version of python .
    b. Make sure to take a look into the **release notes** of the particular version that you are migrating . check here
4. Document changes
    a. Document all package upgrades, version changes, and any compatibility issues or workarounds implemented.
    b. Make sure to freeze the pre migration and post migrations requirements in a **requirements.txt** .

3 . PRE AND POST MIGRATION CHECKS

PRE MIGRATION	POST MIGRATION
Review release Notes • Compare the existing version with target version • Check for deprecation notices and release notes	NA
Deprecations : Identify deprecated functionality. • Address any deprecated features • **makemessages** to identify deprecated features in your codebase.	Verify and check for performance and scalability . • Check for new deprecated warnings . • Test thoroughly to ensure deprecated features are replaced .
Dependencies : Ensure dependency compatibility with target version	Verify dependencies work as expected after migrations , or look for alternatives .
Database migrations : Run DB migrations before migrating to the new version and ensure to take a local copy .	Verify DB consistency after each migration step . • recommended to start with fake migrations .
Documentation Updates : Update the project documentation to reflect with the target version	Verify documentation to ensure accuracy .
Monitoring : Monitor application closely for existing issues and errors .	Address issues promptly to maintain better application stability

Codebase Update : Review for compatibility changes . ● Update syntax based on target version ● Update django settings file	Test code base to ensure functionality remains intact ● Verify settings to ensure correct configuration
Deployment : Note all the deployment configuration commands for either ASGI or WSGI based server .	Deploy to a testing or pre-prod environment and perform testing , monitor logs and report errors. ● Conduct load testing for high traffic handling (if needed)
Document : Document all the changes done and share to the team for future reference and continuous improvements .	Do a pre-deployment check and celebrate migrations .

4 . SECURITY CONSIDERATIONS

- Review and update authorization mechanisms , ensure to update the project's **security key** .
- Keep the dependencies up-to-date .
- Subscribe to security advisors and vulnerability checking tools.
- Implement or Utilise validations and output **encoding** .
- For enhanced security posture do a periodic audits (if needed)
- Advice with stakeholders and Ensure compliance with data protection policies . (if needed)

7 . CONCLUSION

As we conclude our journey through the Django Project Version Migration Guidelines, let us reflect on the progress made and the knowledge gained. By embracing the spirit of innovation, resilience, and continuous improvement, we have equipped ourselves with a comprehensive roadmap to navigate the challenges of migrating or upgrading Django projects successfully.

Throughout this guide, we have delved into preparing for version migrations, upgrading packages and dependencies, handling deprecations, and implementing security best practices.

Sincerely,

Name : ANAND RAJ B
Role : SDE - L3
Date : 02-04-2024