RAD Service

RADS

Best Practices on Application Building

https://github.com/radservice/rad-community/fork

Last Revised on Oct 2021

Prerequisites

1. Understand all major components of RADS.

Content

1. Application Building Best Practices
2. Performance

Chapter 1

Application Building Best Practices

Application Building Best Practices

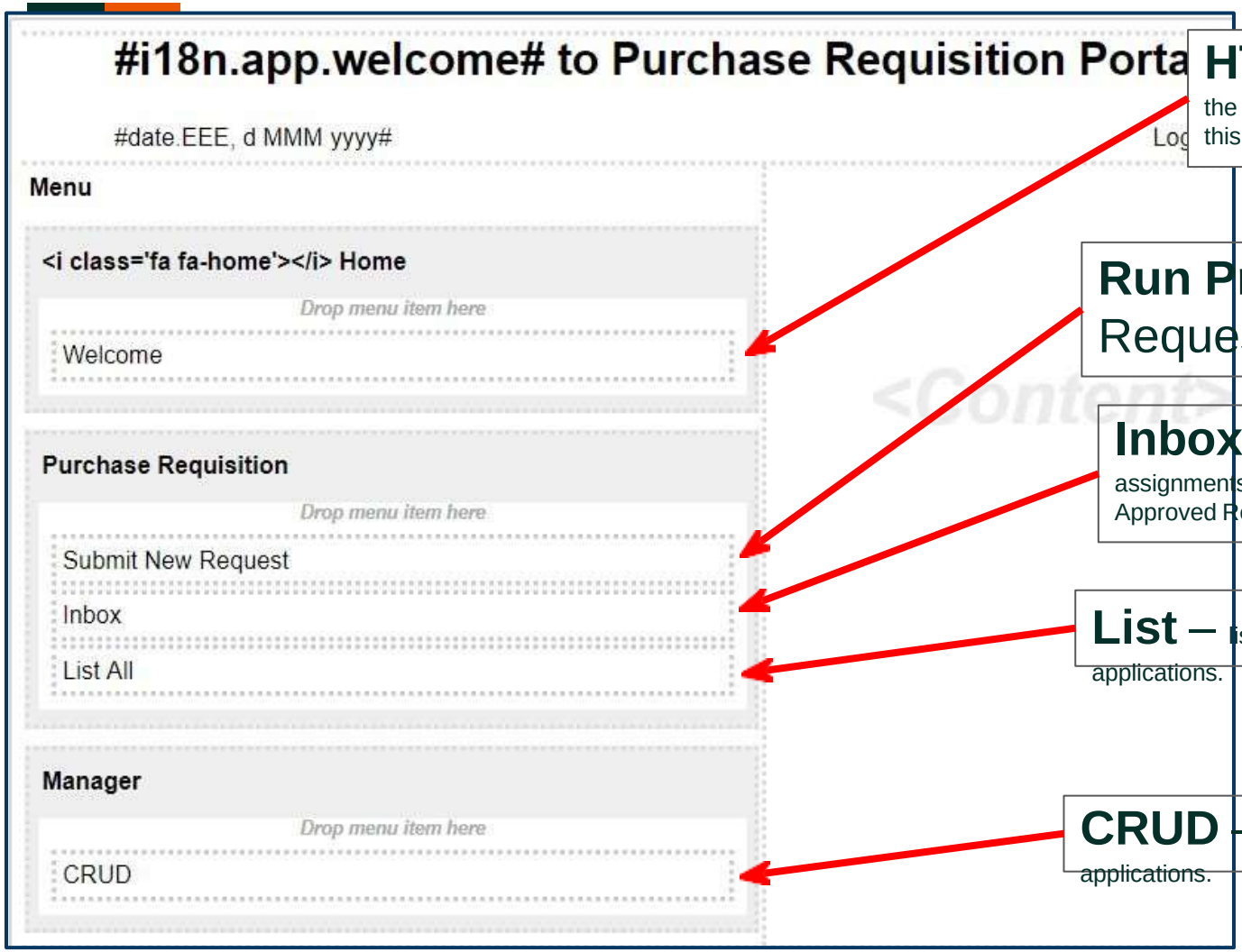
1. Draw a Sitemap
2. Segregate Userview by Roles
3. Iterative Build Process
4. Presence Indicator
5. Naming Convention
6. Taggings
7. Beanshell Coding
8. App Versioning
9. Process or Data
10. Defaults
11. Notes
12. Comments
13. Chrome Groups

Draw a Sitemap

- Before you start designing your app, try to visualize on all these components would work together.
  - Process
  - Form
  - List
  - Userview

  by using the Userview as the centerpiece of the planning.

Draw a Sitemap - Purchase Requisition App



HTML Page – to serve as the Landing Page End User should see this when they come to the site

Run Process – to start the Request Requisition process

Inbox – where everyone will access their assignments (i.e. Approve Request, Acknowledge Approved Request)

List – list down current user's applications.

CRUD – manager view to manipulate applications.

Segregate Userview by Roles

If you multiple roles and a lot of userview menu elements, it is best to segregate the userview by roles.

Benefits:

- Clear cut Permission Management
- Performance (CRUD / Inbox Count, etc)

Iterative Build Process

- Build as you go.
- Establish the most basic layer first.
- Modularize.
- Design, configure and test individual module independently when possible.
- Increase complexity between each iteration.
- Teamwork is good but….
  - Must work in the same server/instance.
  - Must work on separate item/sections at any point of time.
  - Communication and expectation must be set clearly between each team member.
  - Assign a App champion to oversee and delegate tasks.

Iterative Build Process

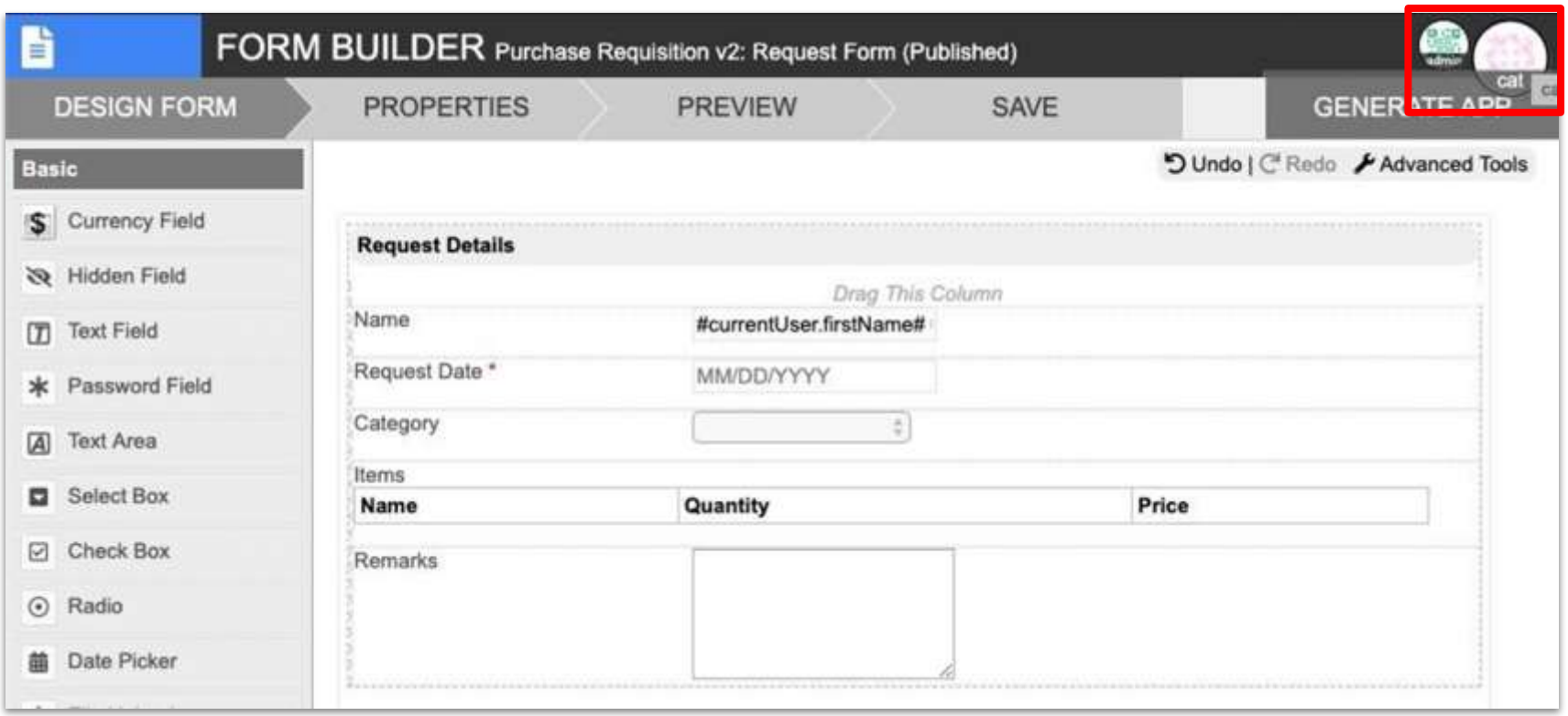
Incrementally go through each step thoroughly before going to the next…

1. Process Design
   - Verify that all possible routes flow as intended to.
2. Form Design
   - Build and test with and without being part of the process.
   - Define table names with ERD in mind.
3. Form with Process
   - With form and process individually tested and verified, then only add Form layer on top of process layer.
4. Datalist
   - With all Form and Process verified and tested, then only design the Datalist for reporting purpose.
5. Userview
   - Should come last depending on the flow of design you have. With everything tested, one can now add in permission control across all the layers.

Once a cycle is completed, consider versioning it to save your work before moving to next new cycle.

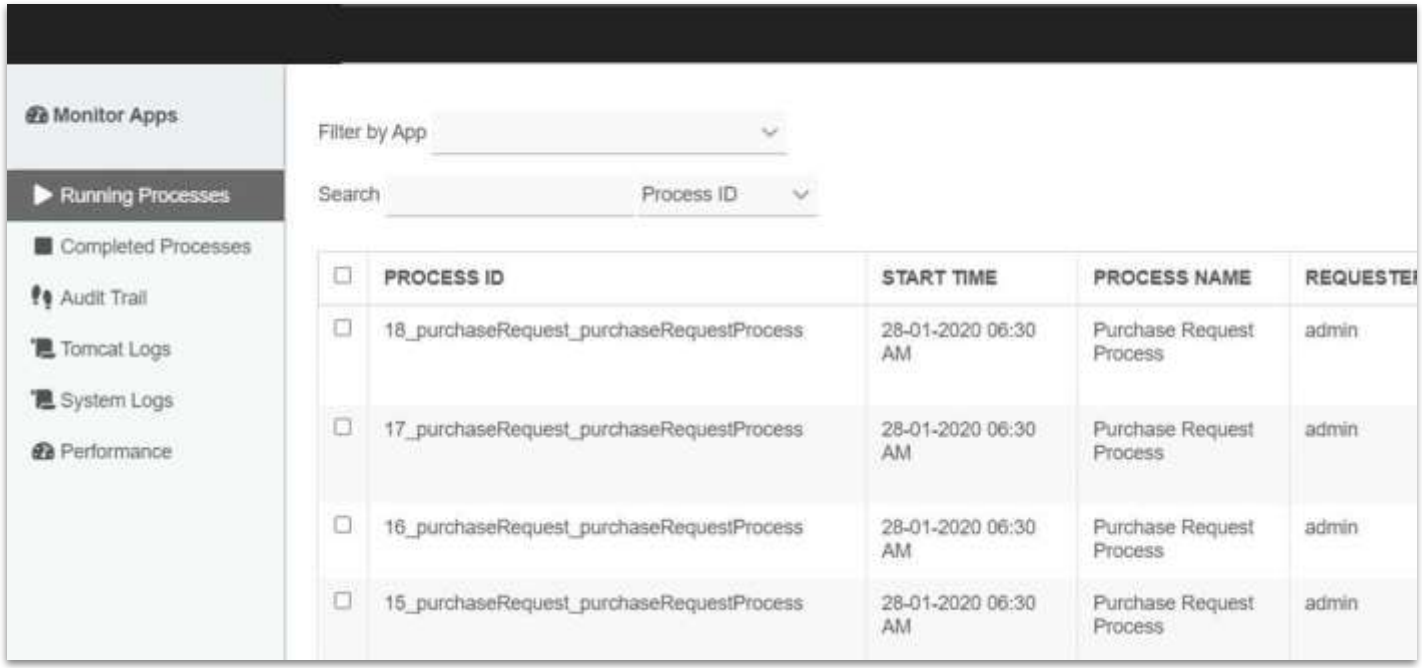
Presence Indicator

- Real-time presence will check to see if someone else is viewing or editing a RADS Component while you have it open - Helps avoid conflicts and promote collaboration.



Naming Convention

- Use **camelCase** when appropriate. Particularly:-
  - App ID
  - Process ID
  - Activity ID



Because RADS uses underscore (_) as the separator in the naming of Process Instance ID.

Naming Convention

- Use **camelCase** for the rest as well to maintain consistency.
  - Form ID
  - Datalist ID
  - Userview ID
  - Workflow Variable

Naming Convention

- Use **snake_case** when dealing with **database** related fields. Particularly:-
  - Form Element ID.
  - Form Table Name.

- Define a **prefix** for **Form Table Name**
  - All apps' form data are stored in the same database.
  - Prevent other App from writing into other App's set of tables.

Naming Convention

- Not to be confused with Form Table Name on the previous slide, **Form Name** should be named with process prefix and numbering if it is bounded to a process.
- Example:-

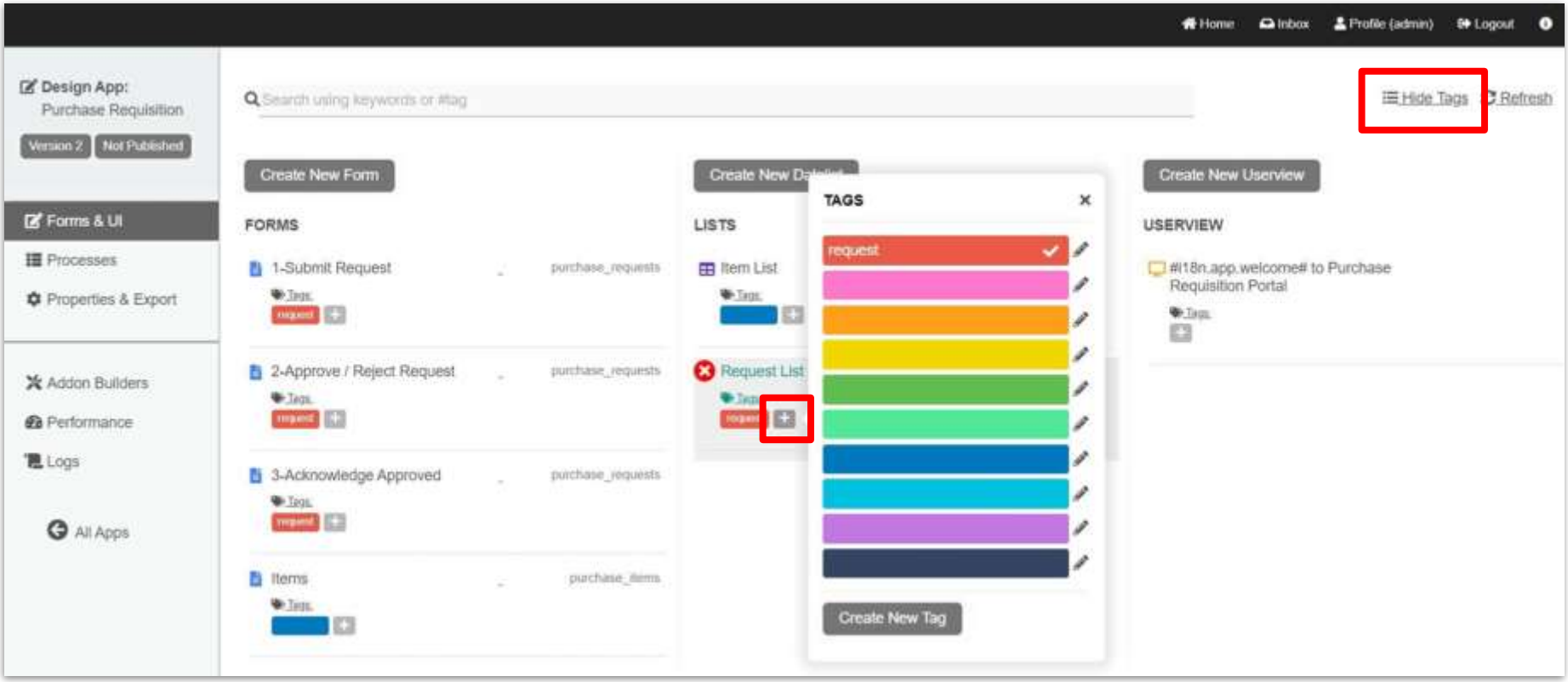
Form Name
R-1-Submit Request
R-2-Approve Request
R-3-Acknowledge Approved Request
QR-1-Submit Quick Request
QR-2-Acknowledge Approved Quick Request

Taggings

- Taggings and Labels are an extra layer of visual data and organization to the RADS components.
- View organized tags at a glance.
- Can also be used in the search filter.

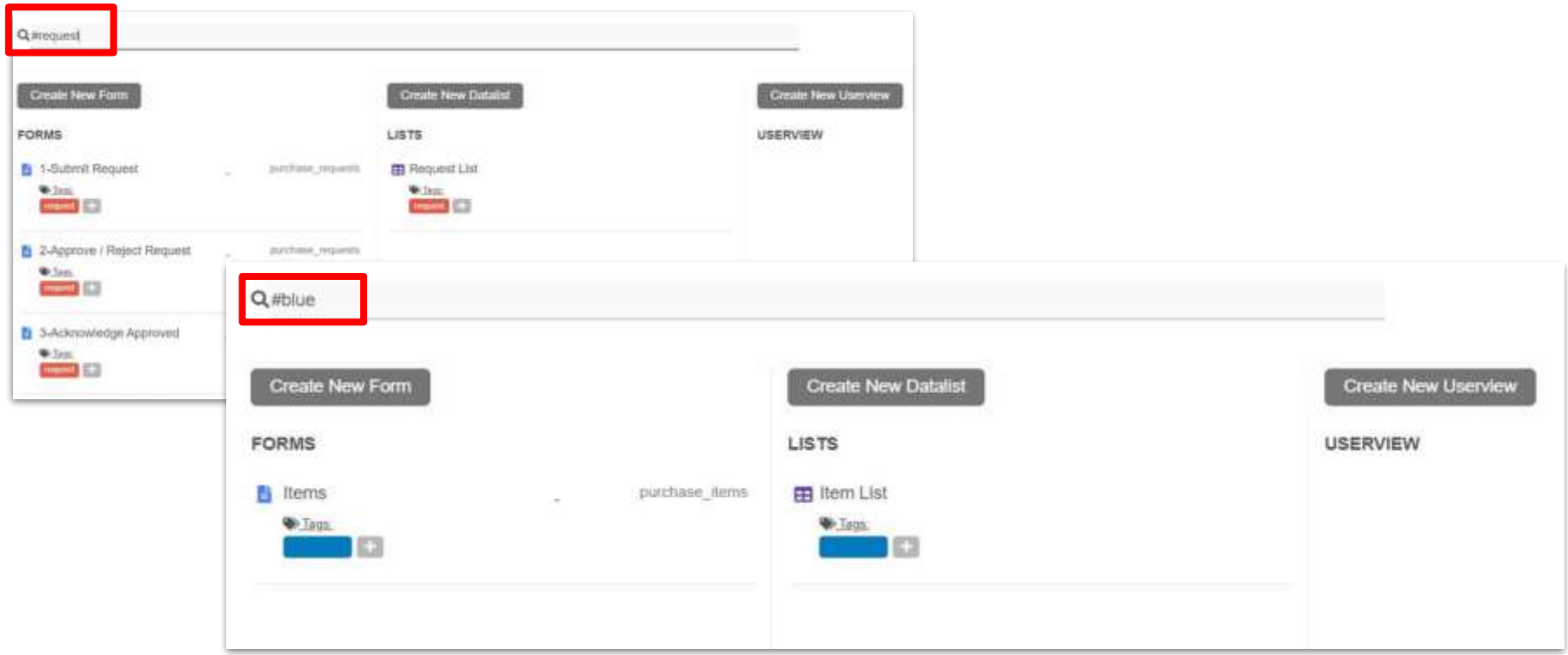
How to Add Tag?

- In **Design App**, click on **Show Tags** and click on **+** sign to add color tags and add labels to the color tags.



Use Search Filter With #tag

- Use Search filter with prefix # to search the relevant tag.
- If no text were defined in the tag, you can search filter by color name.



Beanshell Coding

- Do not use **System.out.println** as it will not be displayed in **Monitor > System Logs** or **RADS.log**
- Use *LogUtil* method instead (see **LogUtil.java**)

Examples:

LogUtil.debug(className,message)

LogUtil.error(className,throwableException,message)

LogUtil.info(className,message)

LogUtil.warn(className,message)

Reference:

https://docs.rads.purwana.net/Managing+Log+Files

Version Control - App Versioning

- Between each **development** cycle, it is recommended to create a new App version before moving on to the next iteration.
- Plan your release to the production.
- **Beware**:
  - Each “New Version” and “Import App” would increase the App Version count in your environment.
  - Production server’s app version **!=** Dev server’s app version.

Version Control

App Version	Process Version
1	1
1	2
2	3
2	4

+1 Update **process version**

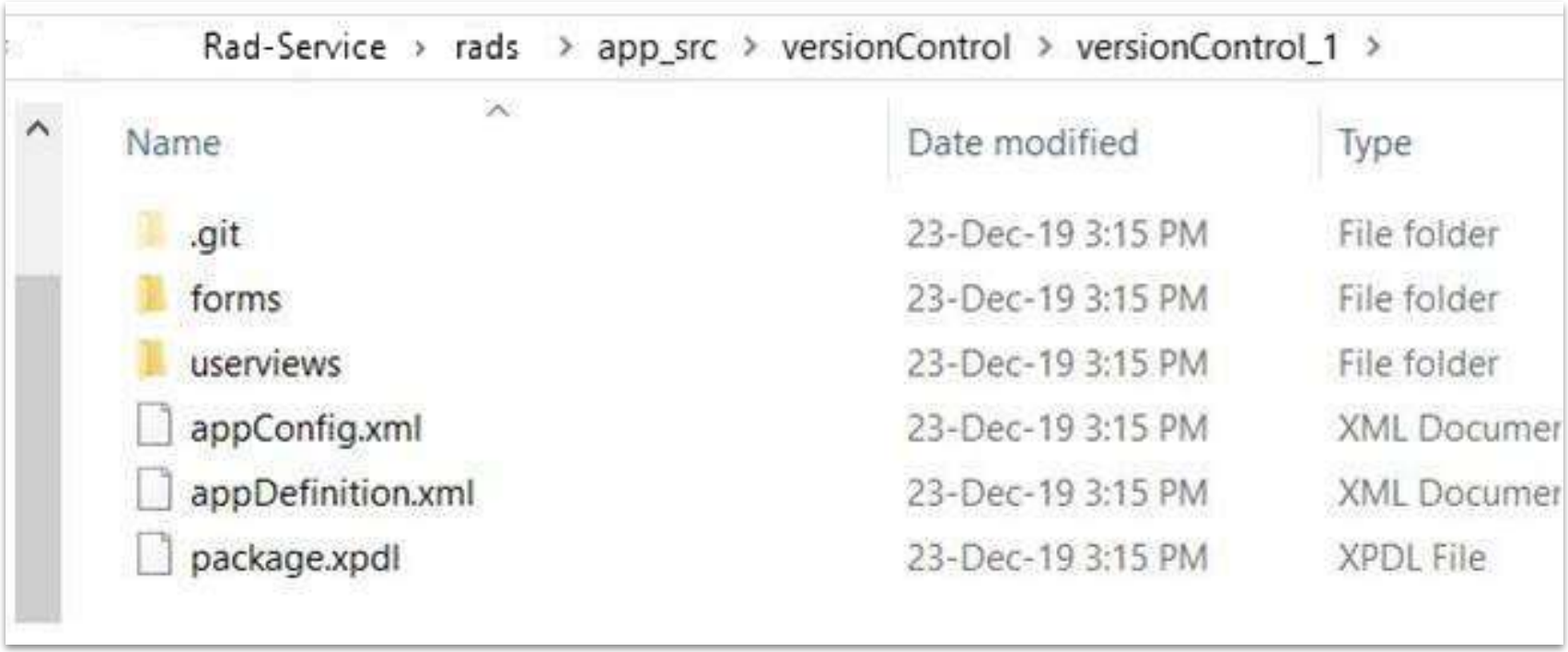
+1 Update **application version**

+1

+1 Update **process version**

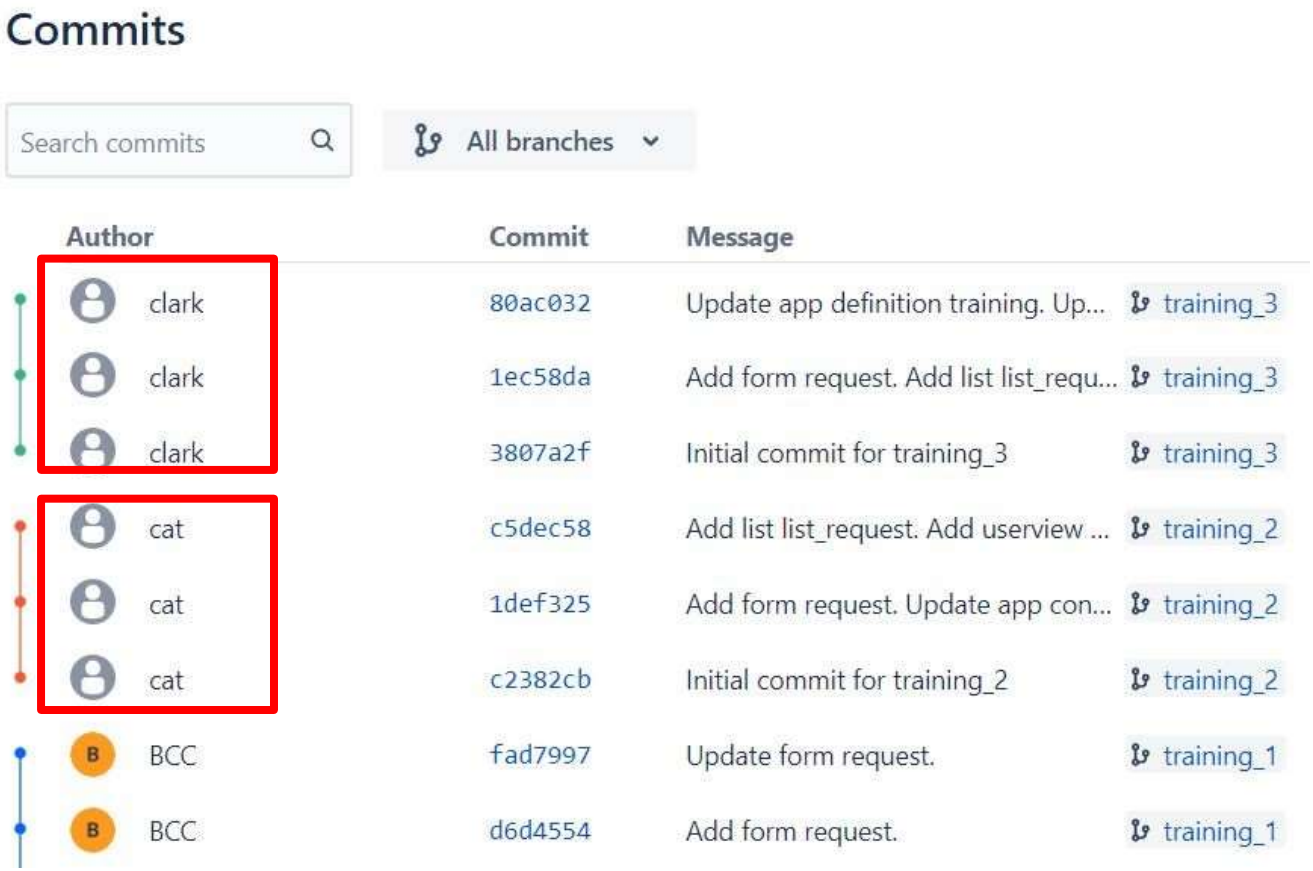
Version Control - GIT

- RADS DX has in-built Git version-control system.
- You may use it to track changes made to your app.
- Find out more in Module 12.



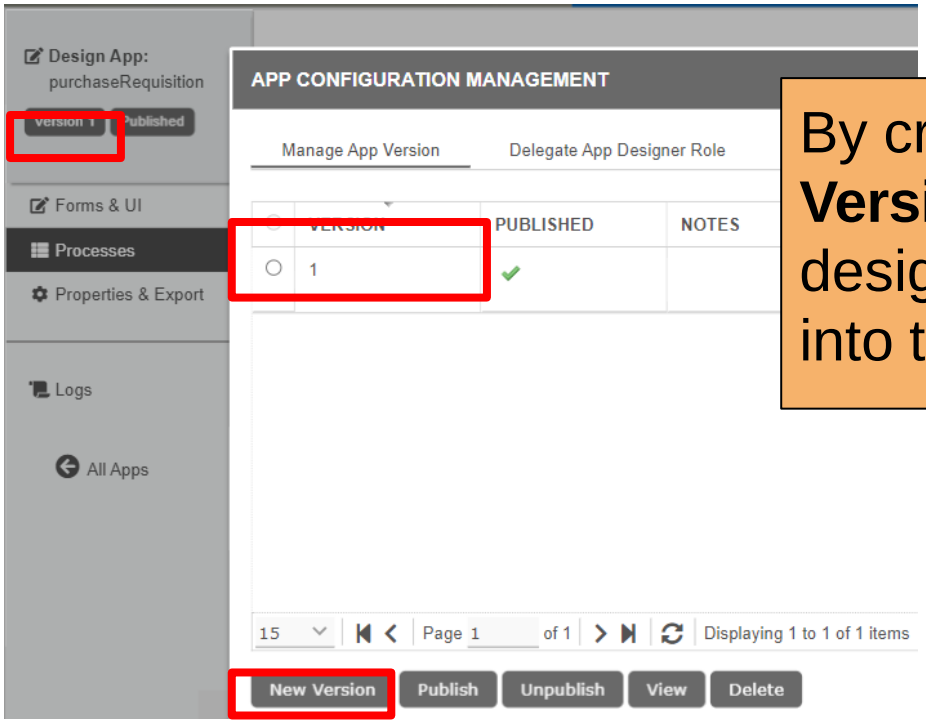
Version Control - GIT

- Ensure to use **named user** during development, it will make it more meaningful to track changes



How to Update Application Version?

1. App Control Panel > Versions > Select version > New Version.



By creating a **New Version**, the App design will be **cloned** into the new version.

Online Reference:

https://docs.rads.purwana.net/App+Versioning+and+Publishing

How to Update Application Version?

2. Import App
   - By importing the app into a RADS server, the Application Version will **increase by 1** over the existing version already in the server.

What does this means?

When you are dealing with the same app across different RADS servers, you may end up with different app version in each server but with exact same app design.

App Version Across Different Servers

- When you are dealing with the same app across different RADS servers, you may end up with different app version in each server but with exact same app design.

Development Server **Production Server**

Keeping Track of App Design Across Different Servers

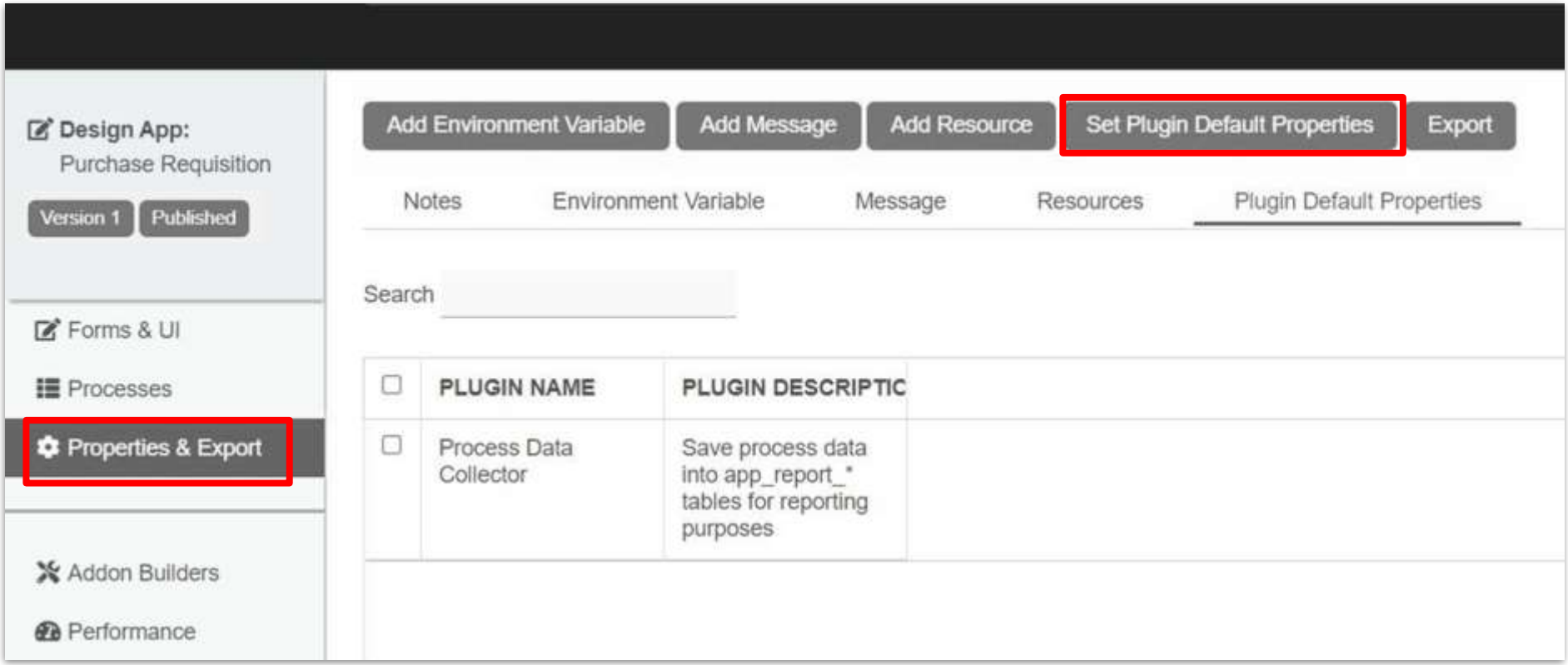
- With the nature of increment of the last app version when an app is imported in, it is imperative to keep track of the “real” app version (app design).
- Make use of **Notes** in app’s properties.

Process or Data

- Do you really need a Process?
- Are you merely doing CRUD?

Defaults

- To minimize maintenance, one may consider to use…
  - Plugin Default Properties.

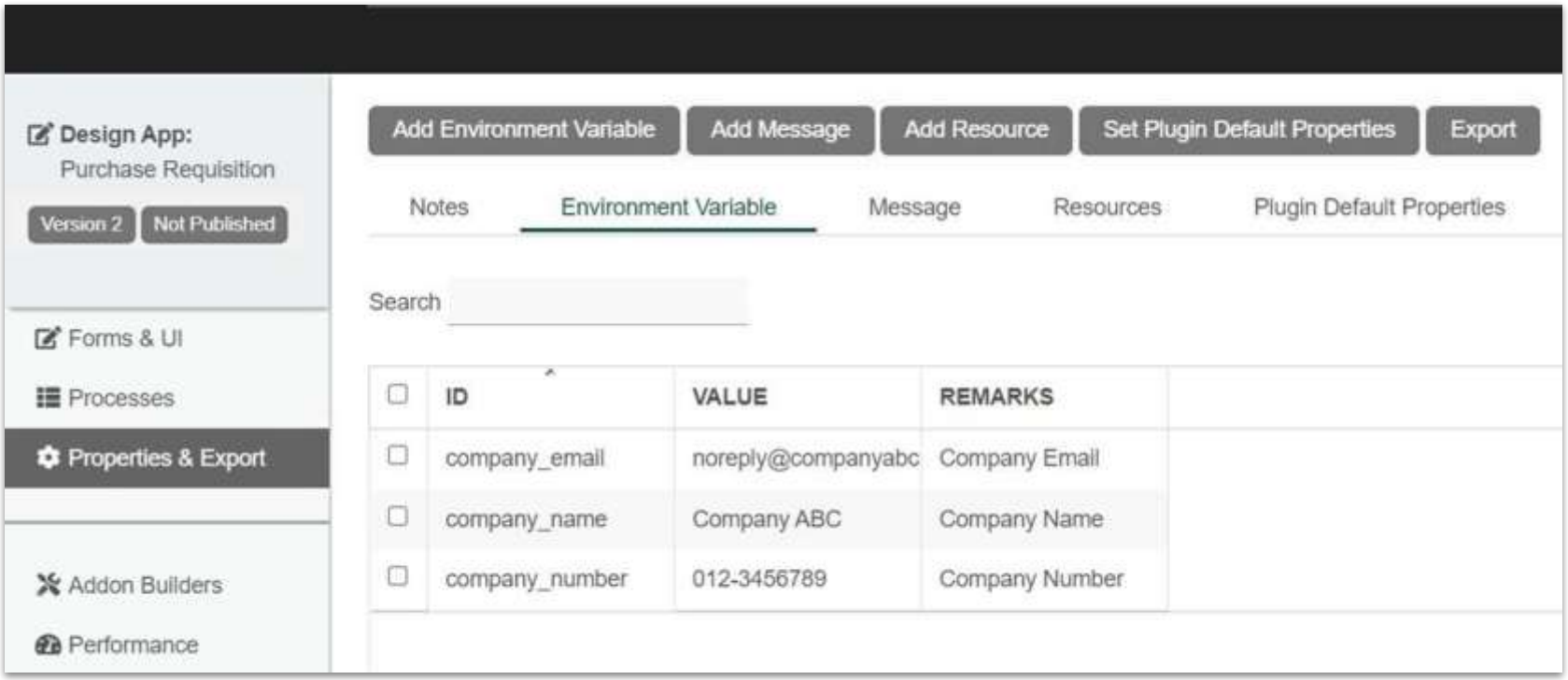


About Plugin Default Properties…

- It’s good to…. only apply/add in changes after each plugin is tested individually, to steer clear of configuration mistake(s) or bugs.

Defaults

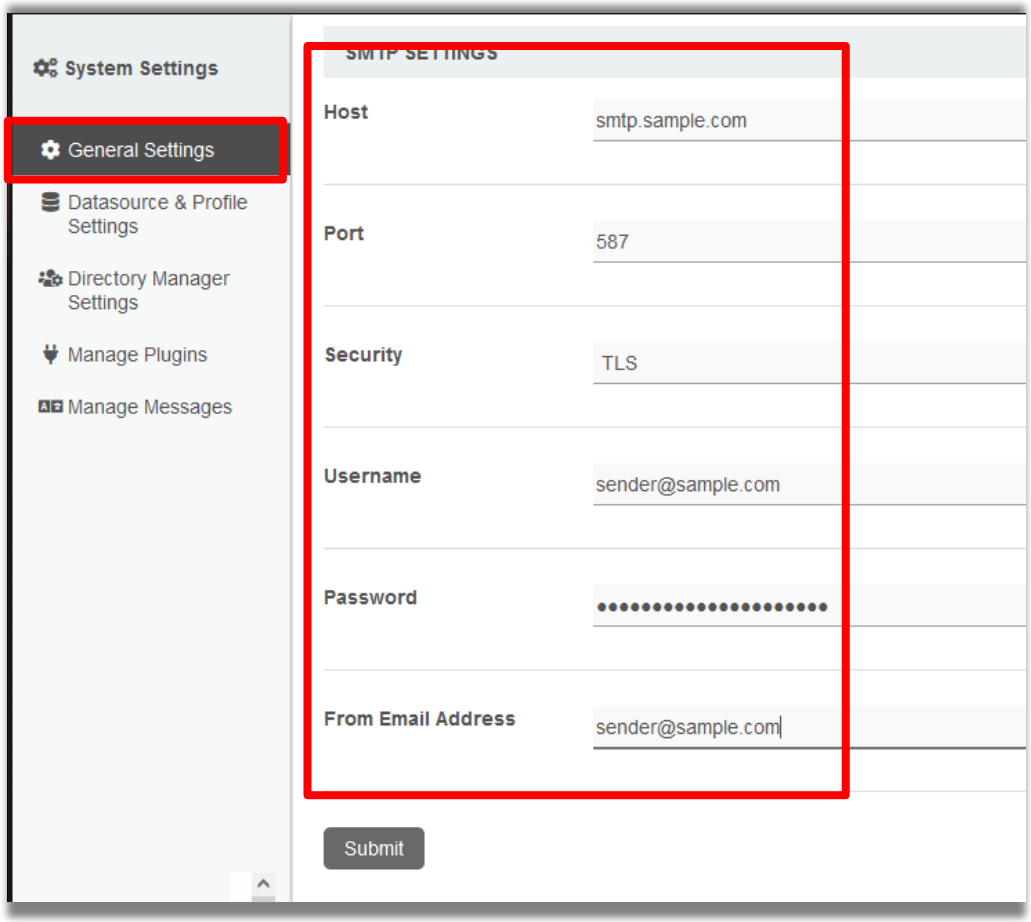
- To minimize maintenance, one may consider to use…
  - Environment Variable, to factorize frequently used values



Refer to our hash variable KB to learn how to use environment variables

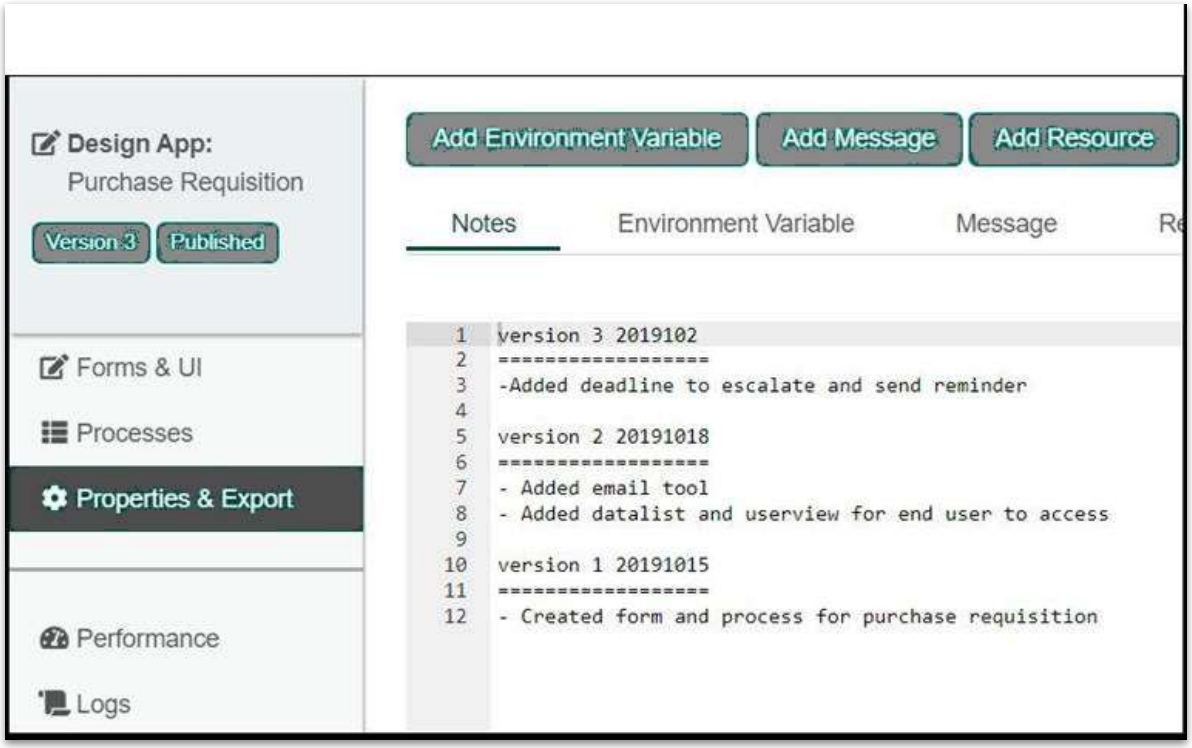
Defaults

- To minimize maintenance, one may consider to use…
  - Platform-level email settings.



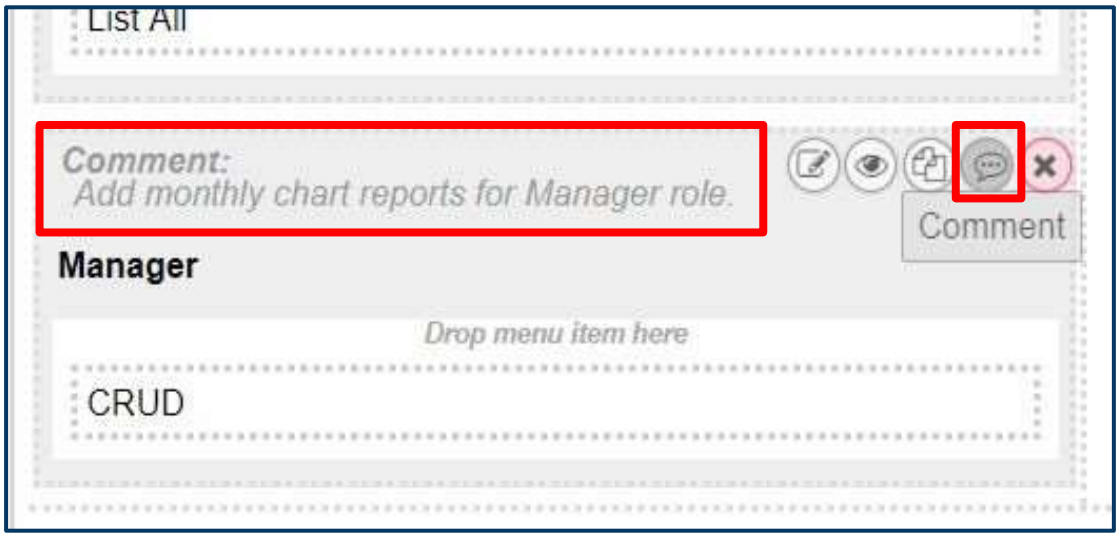
Notes

- Use this feature to jot down important notes for your apps.
- Example: keep track for app development changes.



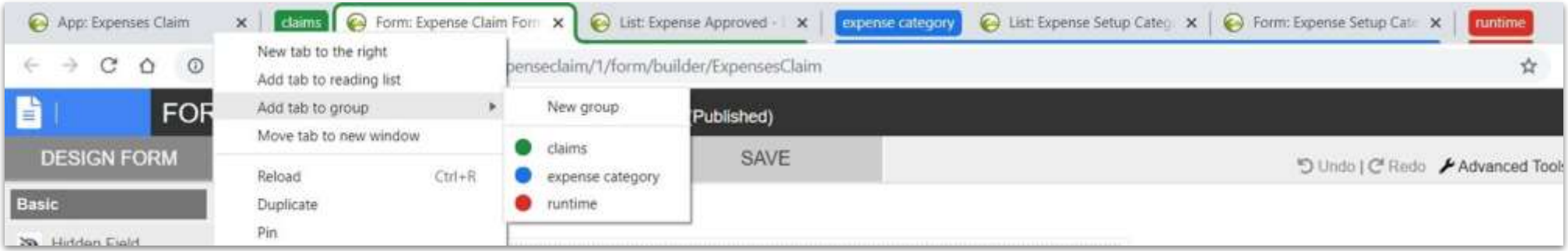
Comments

- Use this feature to comment on a form section/userview category. This can aid comprehension in a collaborative development team.
- Example: take note of the intention of a section or category/pending edits/notable info/etc.



Chrome Browser - Add Tab to Group Function

- If you use chrome browser, try use the add tab to group feature to stay organized, avoid confusion and open relevant tabs only



Chapter 2

Performance

About Performance...

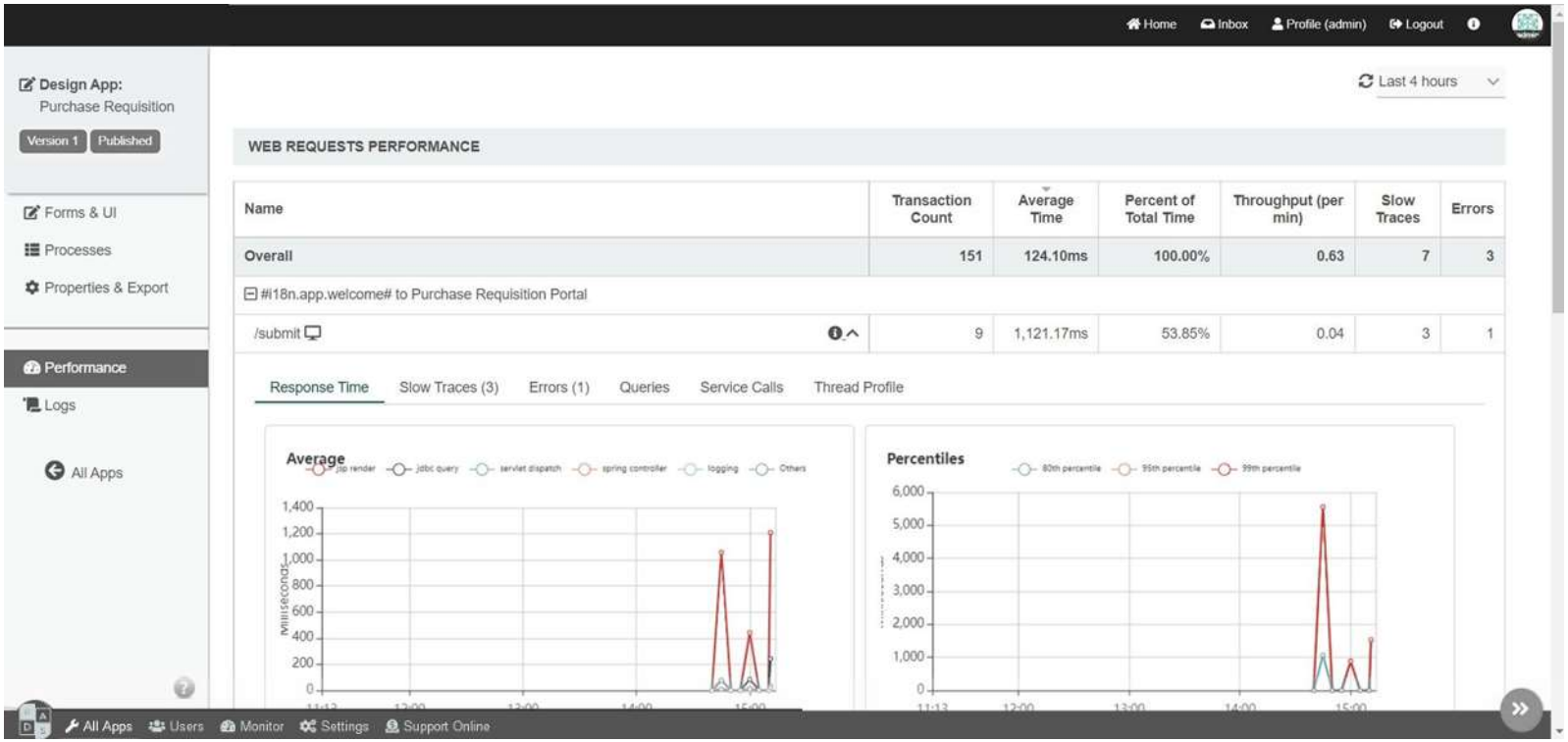
- Performance varies **wildly** from one to another depending on factors such as:-
  - Total number of users.
  - Maximum expected concurrent users.
  - Number of apps running on the platform.
  - Complexity of each of the apps.
  - Amount of data generated in each app.
  - Network infrastructure.
  - etc...

Server sizing

- Since each app differs to one another:
  - There's no one-size-fits-all
  - If its too small = poor performance
  - if its too big = waste of resources
- Run stress test / load test to measure its capability
- Check APM & Performance Analyzer (DB connection leaks) to ensure it can sustain the load and does not crash the app / database / server

Performance Monitoring

- Any resource usage (e.g. JDBC connections in Beanshell scripts or plugins, IO, etc) must be closed in a try-finally block to prevent leaks.
- Monitor using **APM** to monitor usage patterns across differents apps.



App Development DEVOPS style

- Use APM to continuously monitor slow traces and errors during development
- Continuously monitor existing pages for performance improvements and regressions.

Scripting

- **Scripting**
  - Beanshell scripts are good for **rapid** development, but for large deployments, it may be slow.
  - Consider developing/porting to a plugin.

Integration

- Any integration (e.g. SQL queries, REST calls, etc) may be slow, so optimize them.

Database Indexing

- Form Data are stored with the prefix “**app_fd_**”.
- Columns are saved in **text/varchar** to maintain flexibility.
- Add indexing as needed, depending on use case.
- MySQL/MariaDB related: Enable slow query logging to track and improve slow queries.

Caching

- Since v6, all userview elements can be cached to reduce resource-intensive server/database operations (for non time-sensitive elements only).
- BE CAREFUL when selecting caching scope.

- Visit

https://docs.rads.purwana.net/Performance+Improvement+with+Userview+Caching to find out more.

More…

- Visit https://docs.rads.purwana.net/Deployment+Best+Practices for more deployment best practices and tips.

Module Review

1. Application Building Best Practices
2. Performance

Stay Connected With RADS

- rads.purwana.net
- https://github.com/radservice/rad-community