Tags
Find a Tag
analysis application
availability_section best_practices
cad change_management
changes compare
compare_revisions
customizations customize
database db2 exchange
find_available_times gantt_chart
gantt_scheduler group
memory_footprint modifications
modify object_label
object_revision
operating_system oracle

You are in: IBM TRIRIGA > UX Framework > UX Tips & Tricks > How to vulcanize your UX application

How to vulcanize your UX application

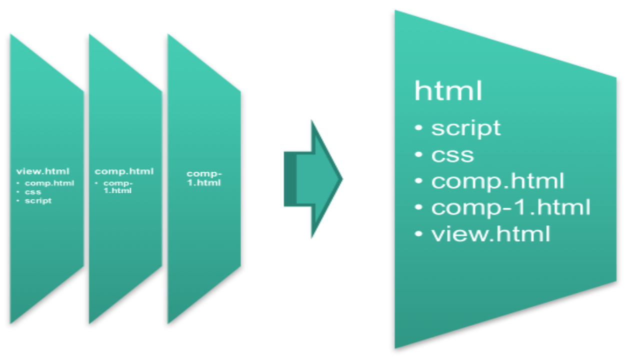
1 Like | Updated March 8, 2018 by kenny.to7 | Tags: *None*

Page Actions

There are new updates to this process starting version 3.5.3 and 10.5.3. See:

- #5 of Prerequisites for the updated UX metadata configuration
- #2 in Steps for the updated tri-vulcanize command
- #3 in Steps for the note about copying images which is no longer applicable

A typical TRIRIGA UX application will initially load hundreds of resource files which translate to the same number of HTTP requests from the server to the browser. This remains true in the succeeding requests of these resources even if they are already cached by the browser. The browser still makes the same number of HTTP requests to the server then waits for the server response to either load it from the cache or from the latest version from the server. This great number of HTTP requests is costly and can affect the performance of your application. One way of cutting down this cost is to concatenate these web resources into a single file through the **Vulcanize** build tool created by the Google Polymer team.



The TRIRIGA team has developed a tool called **tri-vulcanize** that specifically vulcanizes TRIRIGA UX component files. It pulls the component files from the TRIRIGA server based on the given view to vulcanize, then uses the Google Vulcanize build tool to concatenate these files into a single file.

Prerequisites:

1. Upgrade TRIRIGA Platform version 3.5.2 to be able to vulcanize UX applications.
2. Upgrade TRIRIGA Application version 10.5.2. to start getting the vulcanized Perceptive applications.
3. **** Download the tools here ****
4. For version **3.5.2 and 10.5.2**, follow the steps here in setting up the UX metadata: Setup the UX metadata for 3.5.2 and 10.5.2 for customized and new UX applications.
5. For version **3.5.3 and 10.5.3** and later, follow the steps here in setting up the UX metadata:
   How to vulcanize your UX application starting 3.5.3, 10.5.3 and later for customized and new UX applications.

When do I vulcanize my UX Application?

You will need to vulcanize your UX application whenever there are changes to your view files or platform component files. These usually happen when a developer has updated your view files or from application OMs and platform upgrades. It is best to re-vulcanize your UX applications when any of these changes occurred, even if you are not sure that there were UX component and application files getting updated in the upgrade.

Steps

1. **WebViewSync** - This is optional but adds convenience when pushing the vulcanized file to the view record in your TRIRIGA server. We highly recommend to use this tool so that after the vulcanized file is generated, it will be picked by the sync command and automatically pushes the file to the server. If you are not using this tool in your regular UX view development, you may skip this and use the same method or process you used when updating the file in your view like a manual upload to the UX view metadata record.
   - **ADDVIEW or PULL** - You will need the **vulcanized view** to be added or be visible by the WebViewSync tool so you need to run the ***addview*** command if it is your first time to setup the vulcanized view in your local workspace or ***pull*** command if you have already added it previously so that the vulcanized file in your local workspace is updated with the latest version from the server. The latter prevents you from getting conflict errors when the vulcanized file in your local workspace is outdated.

     ```
     java -jar WebViewSync_3.5.2.jar addview -v triview-group-move
     or
     java -jar WebViewSync_3.5.2.jar pull -v triview-group-move
     ```

   - **SYNC** - Run the *sync* command as you would regularly when you are updating any of your view files and you want those updates to be pushed to the TRIRIGA server. The vulcanized file is just like any other UX files in your view record.

     ```
     java -jar WebViewSync_3.5.2.jar sync -a
     ```

2. **tri-vulcanize** - run the *tri-vulcanize* command against the files of the view you want to vulcanized i.e. your development view files.

   NOTE: For UX applications delivered in version 3.5.3 and 10.5.3 and for UX applications maintaining only one set of UX metadata records, use the **tri-vulcanize** command in here: How to vulcanize your UX application starting 3.5.3, 10.5.3 and later

   ```
   tri-vulcanize --user vinid --password password --url http://hostname:port
   --view triview-group-move --component triview-group-move-dev
   --output triview-group-move/triview-group-move.html
   ```

   Where:

--user	The user name of your TRIRIGA user.

--password	The password of the user.
--url	The website URL of your TRIRIGA server where to pull the files of your view from. Do not add slash (/) at the end of the URL.
--view	The exposed name of the development view record you want to vulcanize.
--component	The root component name of the development view record you want to vulcanize.
--output	The folder or location where to generate the vulcanized file and its filename. You need to provide the location of the folder of your vulcanized view in your local workspace.

Some things to consider when running this command:

- Do not run this command inside an active view folder in your command line. The *tri-vulcanize* command will create a temporary build folder (with name like *components-<some numbers>*) which stores the files pulled from the server then remove it after the vulcanized file has been successfully generated. If the *sync* command of WebViewSync tool is running in the background against the view, the temporary build folder will be pushed to the server unnecessarily, so run the *tri-vulcanize* command outside of any active view folder.
- When the vulcanized file was not successfully generated due to errors, the temporary build folder will not be deleted, this is so that you can check the files it pulled from your server to help you diagnose the problem. You will need to manually remove this folder after resolving the issue.
- You can ignore the warnings, [warn], thrown in the terminal but watch out for errors, ERROR.
- If you see unknown errors, try upgrading your NPM and tri-vulcanize tools to the latest versions.

3. **Copy images -** Manually copy all images (if available) from your development view record to the vulcanized view record so that the vulcanized file can access those images at runtime. In your terminal, go to the folder of your development view, check if there is *images* folder, if there is then run the command below and provide the location of the folder of your vulcanized view plus the images folder.

   NOTE: This is no longer applicable if you have upgraded to version 3.5.3 and 10.5.3 and if your UX application only maintaining **one set** of UX metadata records.

```
<your development view folder>~$ cp -R images/ ../triview-group-move/images
```

4. **Push** - Push the vulcanized file to the TRIRIGA server

```
java -jar WebViewSync_3.5.2.jar push -f -v triview-group-move
```

Comments (0) | Versions (62) | Attachments (8) | About

There are no comments.

Add a comment

Feed for this page | Feed for these comments