:shell

samplelaunchone-01-adminoperadores-perfil-datos.sh

argv[0]: "../../../lib/common-test/launchers/launchbyname.js"
arv1[1]: "@testspath/springnut/adminoperadores-springnut/suite-01-adminoperadores-perfil-datos-springnut-test.json"

:node

launchbyname:m
require
launcharguments:m
require
commandline:m
.testName
.testSpec
fCmdLineArguments()
.argv
process:g

fDefineLauncherOne_andRunIfFirst(testname, testSpec, launcharguments)

launcher-one:m
require
firsttorun:m
flsFirstToRun
<clbk
exitRecord:callback
FirstLauncherOneFactory
LauncherOneFactory
LauncherOne_Constructor
launcher-one:i
pF_LauncherOne(clbk)
pF_LauncherManyOrOne(clbk)
<clbk
launchOkFailRecord:callback
fBoundTestStepFunctions()
<seriesToRun
pF_pRunSeries(seriesToRun,clbk)
pInitFinderToRunLauncherSeries(theSeriesToRun)
OverallRunLauncherSeriesCallback:callback
pTryToRunNextInLauncherSeries(OverallRunLauncherSeriesCallback)
fFindLauncherSeriesToRunNext()
pRunNextInLauncherSeries(OverallRunLauncherSeriesCallback)
<clbk
OverallRunLauncherSeriesCallback:callback

pSL_loadcmpandtestspaths
pSL_initReleasingChecker
pSL_initCleanUpChecker
pSL_readConfig
pSL_parseConfig
pSL_createTest
pSL_runTest
pSL_releasingCheck
pSL_cleanupCheck

pSL_loadcmpandtestspaths(clbk)
cmpandtestspathsloader:m
require() pgOverrideModuleVariations()
overrider:m
require()
custom:m
require()
overrides:m
CmpAndTestsPathsLoader_Constructor
notSupplyingCmppathsnameOrTestspathsname
v_CmpAndTestsPathsLoader:CmpAndTestsPathsLoader
pF_CmpAndTestsPathsLoader(clbk)
pRunCmpAndTestsPathsLoaderSeries(clbk)
shall exec async w/clbk steps:
pSL_load_cmppaths
pSL_instantiate_cmploader
pSL_load_testspaths
pSL_instantiate_specloader
pSL_load_cmppaths(clbk)
_v_CmpPathsName = this.CMPPATHSNAME
cmpandtestspathsloader VARIATION
#root/cmppath.json
require()
pgOverrideModuleVariations()
if _v_CmpPathsName starts with
ROOTPATHSTEPSYMBOL = "#root"
then replace with
cmpandtestspathsloader
CMPANDTESTSPATHSPATHTOROOT = "../../.."
PathsLoader_Constructor(_v_CmpPathsName)
:PathsLoader
pLoadPathsNamed(_v_CmpPathsName, clbk)
pLoadXNamed(thePathsName,clbk)
fLoadXNamed(thePathsName)
fReplaceInNameToLoad(thePathsName)
if thePathsName starts with
ROOTPATHSTEPSYMBOL = "#root"
then replace with
xloader
XLOADER_PATH_TO_ROOT = "../../.."
require()
_v_NameToLoadReplaced
_v_CmpPaths
contents of file at #root/cmppath.json

pSL_instantiate_cmploader(clbk)
fgGlobalCmpLoader
cmploader:m
gCmpLoaderSingleton
CmpLoader_Constructor
gCmpLoaderSingleton:CmpLoader
pClearResLoaderPaths
fAddResAndPackagePaths(DEFAULTCMPPATHSFORPACKAGEPATHS == empty)
_v_CmpLoader
fAddResAndPackagePaths(_v_CmpPaths)
contents of file at #root/cmppath.json

pSL_load_testspaths(clbk)
like pSL_load_cmppaths with
_v_TestsPathsName = TESTSPATHSNAME = "#root/testspath.json" VARIATION
_v_TestsPaths
contents of file at #root/testspath.json

pSL_instantiate_specloader(clbk)
fgGlobalSpecLoader
specloader:m
require() pgOverrideModuleVariations()
overrider:m
require()
custom:m
require()
overrides:m
gSpecLoaderSingleton
SpecLoader_Constructor
gSpecLoaderSingleton:SpecLoader
like pSL_instantiate_cmploader with
specloader.fgGlobalSpecLoader
gSpecLoaderSingleton
DEFAULTSPECPATHSFORPACKAGEPATHS
_v_SpecLoader
_v_TestsPaths contents of file at #root/testspath.json
pClearResLoaderPaths
fAddResAndPackagePaths(DEFAULTSPECPATHSFORPACKAGEPATHS == empty)
_v_SpecLoader
fAddResAndPackagePaths(_v_TestsPaths)
contents of file at #root/testspath.json
* fAddResPath()
_v_ResAndPackagePaths.push({ thePackagePath, theResPath}

pSL_readConfig(clbk)
pSL_readConfig_async(clbk)
specloader:m
fgGlobalSpecLoader()
<- gSpecLoaderSingleton
gSpecLoaderSingleton:SpecLoader
pLoadSpecNamed(_v_SuppliedSpecsSource,clbk)
pLoadResNamed(theSpecName,clbk)
fFullLoadPathsForResNamed(theResName)
same impl
fLibPathPrefix() <- SPECPATHPREFIX
SPECPATHPREFIX "@testspath"
prepend ROOTPATHSTEPSYMBOL = "#root" if only 2 steps in test path
fLibPathsForResNameSteps(theResName)
... pLibPathsForResNameStepsExceptSub_into
pLibPathsForResNameSteps_own_into
returns [theResName,
* all paths with replaced step[0] by matching resPath from
_v_ResAndPackagePaths
which were initialized by
CmpAndTestsPathsLoader pSL_load_testspaths
with content from #root/testspath.json
pLoadResWithFullLoadPaths(someFullLoadPaths,clbk)
defaults to asynchronous loop calling
pLoadResWithFullPath_async_local
with callback and exception handler
attempts to load from each full load path
fLoadResWithFullLoadPath_single
require(theFullLoadPath)
QJO LOCAL
IF SYNCHRONOUS (not the default)
fLoadResWithFullLoadPaths(someFullLoadPaths)
for each full load path attempt :
fLoadResWithFullLoadPath_single
require(theFullLoadPath)
_v_ConfigSpec

pSL_parseConfig(clbk)
specparser_config:m
SpecParserConfig_Constructor(_v_ConfigSpec)
aSpecParser:SpecParserConfig
fParse()
_._v_ParseSuccess
this._v_Configuration = ._v_ConfPopulated
_v_ConfPopulated:Configuration
fResolveImports("PARSE")
pResolveExportsNow("PARSE")
pMatchChecksNow("PARSE")
fFirstCheckFailedToMatchNow("PARSE")
if aFirstCheckFailed
clbk(LAUEVKIND_CHECKFAILED)

pSL_createTest(clbk)
scheduleroot_test:m
._v_RootTest = ScheduleRootTest_Constructor(_v_Configuration)
_v_RootTest:ScheduleRootTes

pSL_runTest(clbk)
._v_RootTest.pF_ScheduleRoot(clbk)
pS_ScheduleOne(_v_Configuration, clbk)
pS_Schedule_withConstructorAndMethodName(theConfiguration, clbk,
theM_scheduleone_test.ScheduleOneTest_Constructor, "pF_ScheduleOne
scheduleone_test:m
ScheduleOneTest_Constructor(theConfiguration)
:ScheduleOneTest
pF_ScheduleOne()
fOwnTestMethodToSchedule
fFieldResolutionsByName(
_v_ConstructorModuleName,
_v_ConstructorName,
_v_MethodName)
fLoadCmpNamed(ConstructorModuleName)
<- aModule
QJO SYNC
fLoadResNamed(theCmpName)
fFullLoadPathsForResNamed(theResName)
fLoadResWithFullLoadPaths(someFullLoadPaths)
for each full load path attempt :
fLoadResWithFullLoadPath_single
require(theFullLoadPath)
QJO SYNC LOCAL
ConstructorModuleName:Module
aModule[_v_ConstructorModuleName]
<- aConstructor
new aConstructor
<- aTest
aTest:ConstructorModuleName
pAddSubordinateTest(aTest)
aModule[_v_MethodName]
<- aMethod
aMethod:function
aMethod.bind(aTest)
call(clbk)