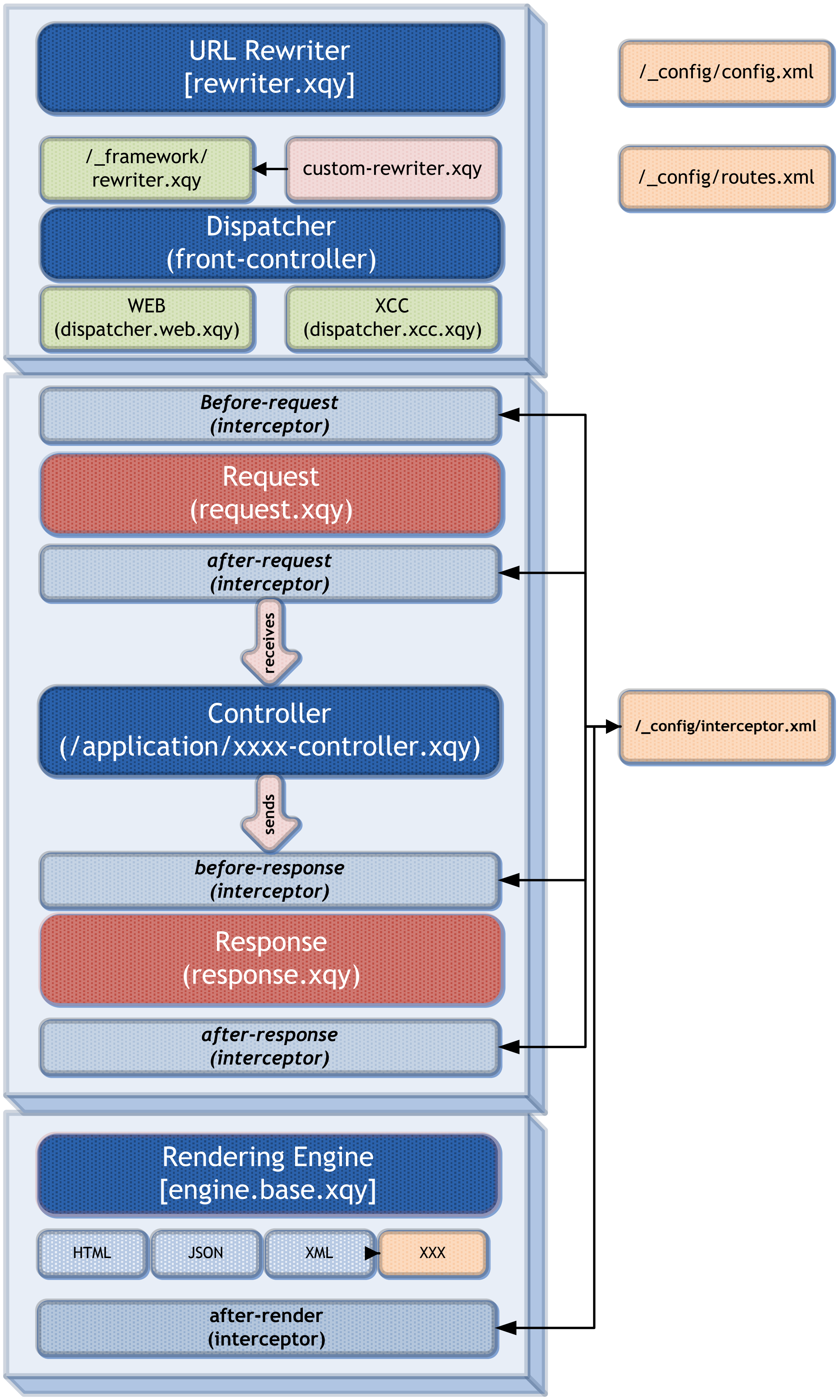
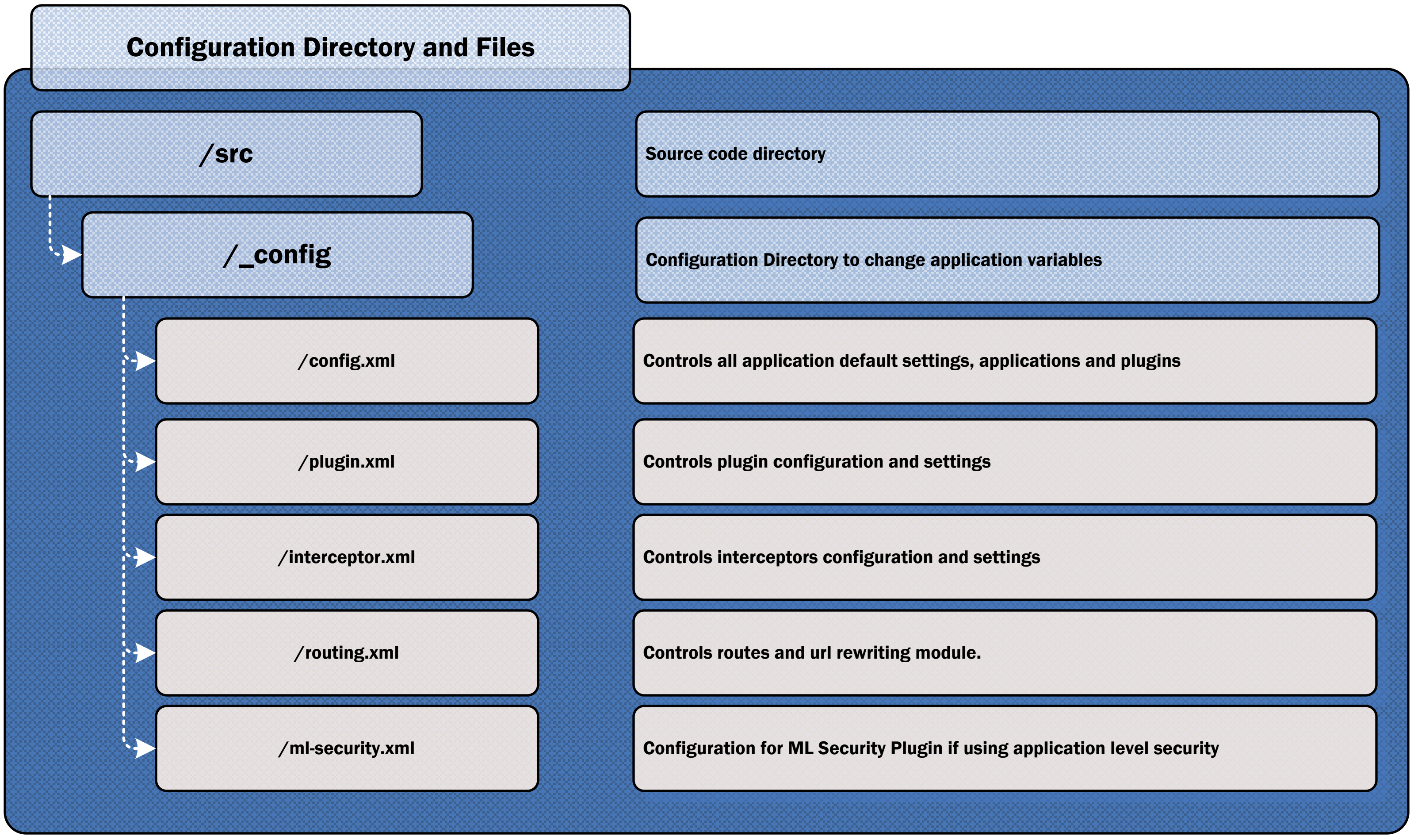
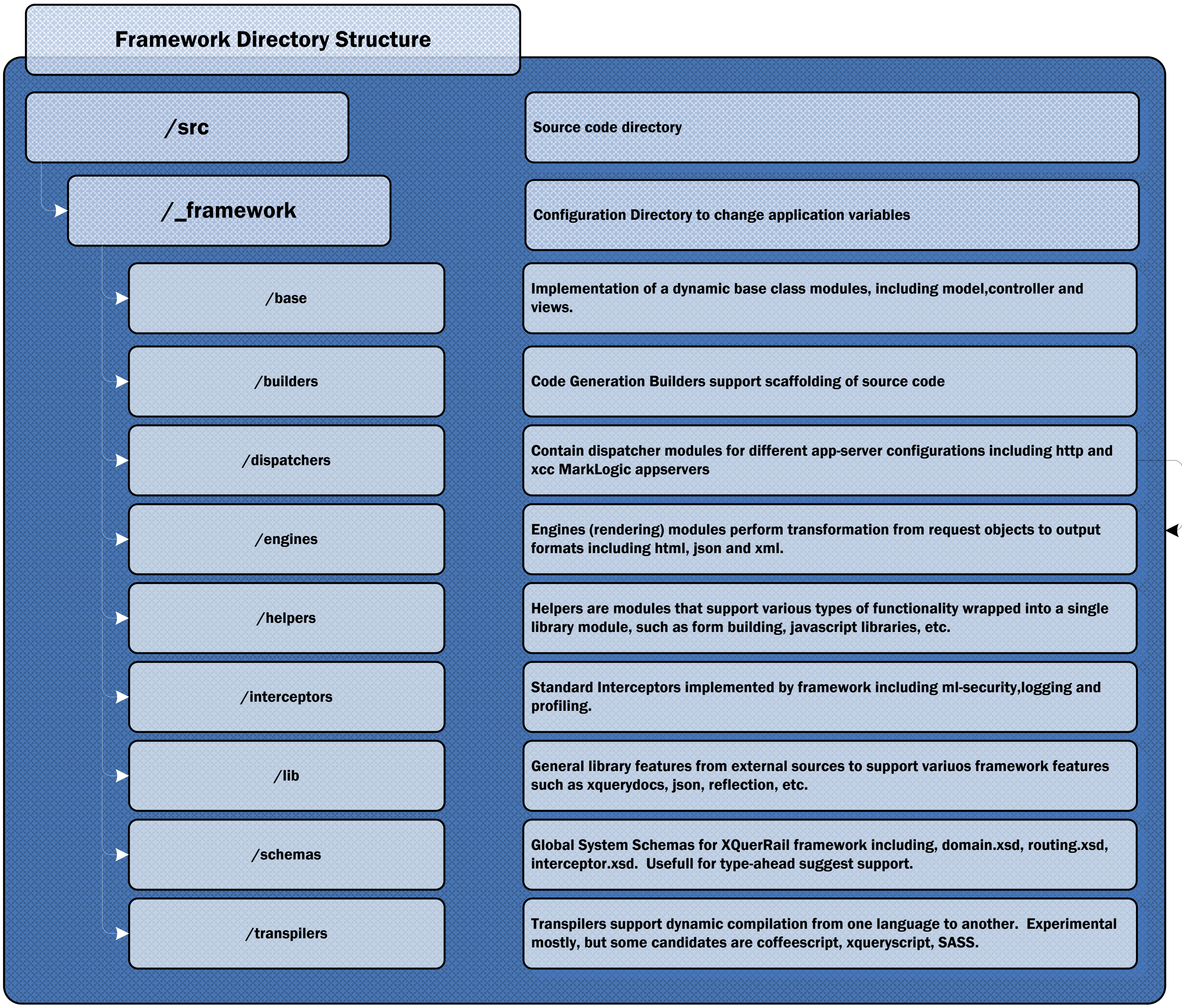
Routing/Request/Response Pipeline



Framework Directory Structure

/src	Source code directory
/_config	Configuration Directory to change application variables
/_framework	Framework Source directory including all core modules used within framework
/_plugins	Plugins directory for custom or shared plugins
/_lib	Common Shared Libraries Directory (not required)
/_resources	Core Resources including Javascript, Stylesheets and Images. This directory is accessible to all applications so ensure your changes reflect global properties.


Configuration Directory and Files
/src
Source code directory
/_config
Configuration Directory to change application variables
/config.xml
Controls all application default settings, applications and plugins
/plugin.xml
Controls plugin configuration and settings
/interceptor.xml
Controls interceptors configuration and settings
/routing.xml
Controls routes and url rewriting module.
/ml-security.xml
Configuration for ML Security Plugin if using application level security



Framework Directory Structure

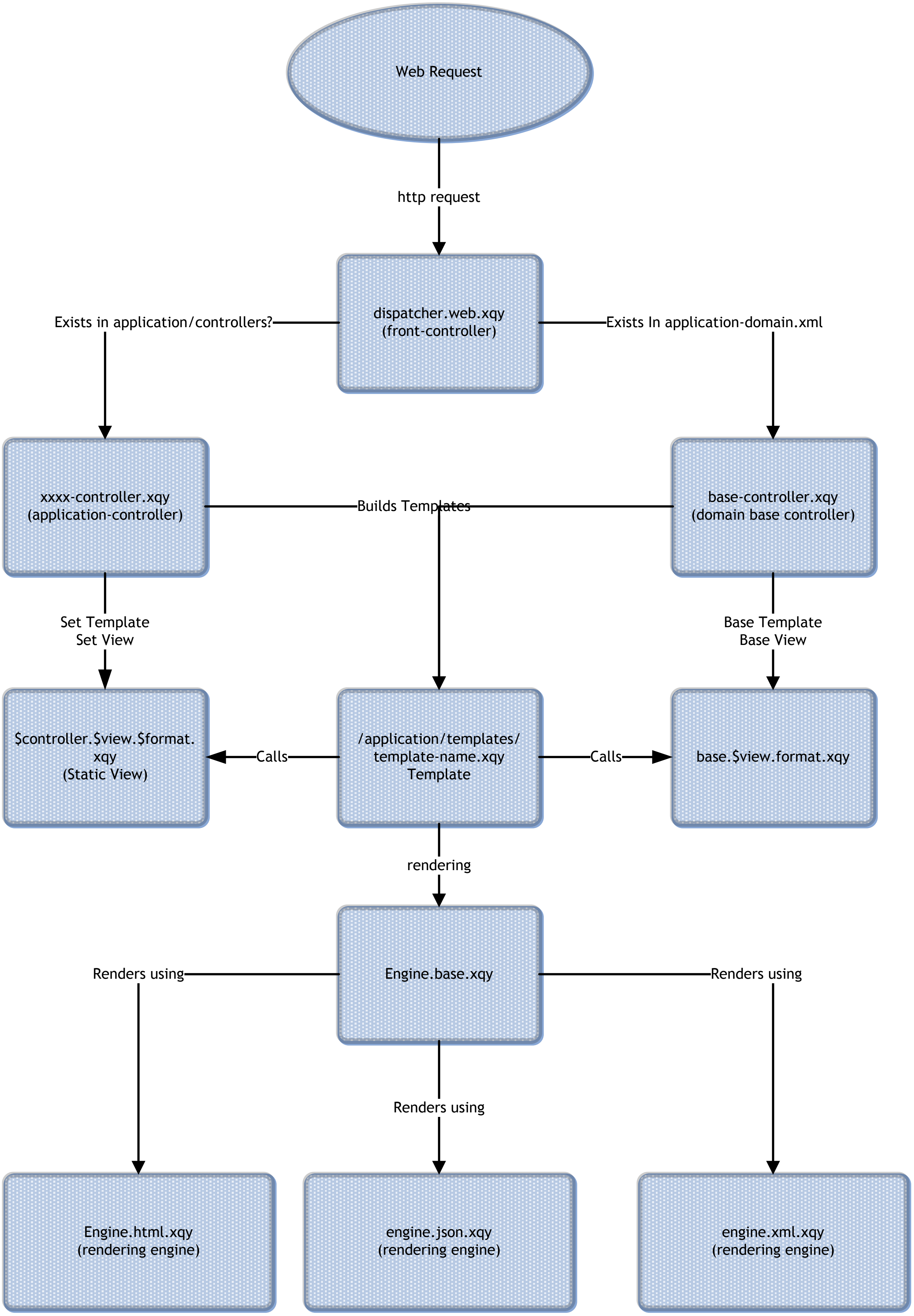
/src	Source code directory
/_framework	Configuration Directory to change application variables
/base	Implementation of a dynamic base class modules, including model,controller and views.
/builders	Code Generation Builders support scaffolding of source code
/dispatchers	Contain dispatcher modules for different app-server configurations including http and xcc MarkLogic appservers
/engines	Engines (rendering) modules perform transformation from request objects to output formats including html, json and xml.
/helpers	Helpers are modules that support various types of functionality wrapped into a single library module, such as form building, javascript libraries, etc.
/interceptors	Standard Interceptors implemented by framework including ml-security,logging and profiling.
/lib	General library features from external sources to support variuos framework features such as xquerydocs, json, reflection, etc.
/schemas	Global System Schemas for XQuerRail framework including, domain.xsd, routing.xsd, interceptor.xsd. Usefull for type-ahead suggest support.
/transpilers	Transpilers support dynamic compilation from one language to another. Experimental mostly, but some candidates are coffeescript, xqueryscript, SASS.

Framework Files Description

/src	
/_framework	Files under framework usually are the core features of the XQuerRail framework. The files below control various aspects of system functionality
/config.xqy	Wrapper functions for accessing configuration settings
/domain.xqy	Wrapper functions for accessing application domains.
/error.xqy	Base implementation of an error handler.
/interceptor.xqy	Wrapper functions for accessing interceptor configuration.
/request.xqy	Wrapper functions around request map. Provide standard parsing and editing of request map.
/response.xqy	Wrapper functions around response map. Provide core response features used during request/response pipeline
/rewriter.xqy	Entry Point rewriter used on app-server if using http server.
/routing.xqy	Controls routing from rewriter. This allows for rewriter module to changed for new implementation such as Moustache or other routing library.

Application Directory Structure

/{$application-name}	Application Directory. Can have directory for each application
/controllers	Main application controllers. The dispatcher will call corresponding named controller module. Controllers for application usually in $controller-name[dot|dash]controller.xqy
/xxxx-controller.xqy	Files should be formatted as ${controller-name}-controller.xqy Namespace : ${app-namespace}/controller/${controller-name} Shortcut Key :${application-name}:controller:${controller-name}:action(${params})
/domain	Controls all dynamic scaffolding for application in application-domain.xml
/application-domain.xml	Controls all dynamic scaffolding for application in application-domain.xml
/lib	Libraries directory commonly shared in your application code.
/xxxx-lib.xqy	Files should be formatted as ${library-name}-lib.xqy Namespace : ${app-namespace}/lib/${library-name} Shortcut Key :${application-name}:lib:${library-name}:{function-name}(${params})
/models	Models for CRUD and Search operations
/xxxx-model.xqy	Files should be formatted as ${model-name}-lib.xqy Namespace : ${app-namespace}/model/${model-name} Shortcut Key :${application-name}:model:${model-name}:{function-name}(${params})
/resources	Resources should be common web resources or code that you would like to include in your application. Also certain tags will look in this folder for assets such as <?controller-script?> or <controller-stylesheet?> Application Specific Resources including scripts, stylesheets and images
/scripts	Script directory used to dynamically insert <?controller-script?> tag. Ensure your file is named exactly as your controller name
/stylesheets	Stylesheet directory used to dynamically insert <?controller-stylesheet?>. Filename must match name of controller in order for it to be included.
/images	Basic image library for application
/tags	Application Tag Libraries. Tags are controlled by using specific names and creating them in your application directory as xxxx-tag.xqy. When a tag with names matches will call the tag render method.
/views	Views directory with each controllers views listed under their name. The view filename should be in the form of $action.$format.xqy
/${controller-name}	Controller Views are stored under the controller name folder and can correspond to a given action or a named view you would like to render from controller. The response:set-view() function will proper view routing.


Web Request
http request
dispatcher.web.xqy
(front-controller)
Exists in application/controllers?
Exists In application-domain.xml
xxxx-controller.xqy
(application-controller)
Builds Templates
base-controller.xqy
(domain base controller)
Set Template
Set View
Base Template
Base View
$controller.$view.$format.
xqy
(Static View)
Calls
/application/templates/
template-name.xqy
Template
Calls
base.$view.format.xqy
rendering
Engine.base.xqy
Renders using
Renders using
Renders using
Engine.html.xqy
(rendering engine)
engine.json.xqy
(rendering engine)
engine.xml.xqy
(rendering engine)