WebBee

Abstract

WebBee is a framework providing building blocks for rapid Java application development for both web and desktop applications including mobile ones. One of main goal of WebBee is reducing learning curve of another product and pretty much stay with basic knowledge of Java, SQL, HTML, JavaScript, CSS and general common software development practices. So WebBee offers very limited additional APIs, scripting languages, and configuration files to learn. WebBee conceptually utilizes just few application design patterns and make a developer to think in the patterns. WebBee also stays on decoupling business and presentation layers giving very flexible ways of connections.. WebBee doesn’t dictate a specific way of creation of an application; it gives much freedom of selecting desired approaches, using 3rd party libraries and etc. It isn’t also overloaded with 3rd party libraries and just defines ways of integration and a use them. WebBee differentiator from other frameworks is giving common building blocks which can be easily customized to fit in a shape of a developed application. However a developer is still free to create own building blocks or work on base level without utilizing building block concepts. It is like using Lego blocks however with possibility to put custom elements making final application look as custom developed.

Design

WebBee is a collection of several frameworks and ready to use solutions build on them. WebBee is SOA friendly and generally encourage of using SOA for applications. WebBee is offering multipurpose container, registry, JDO services, servlet framework with MVC and front controller, template engine, resource manager, object pool, cache and some other utilities and services. . It provides extremely light weight Ajax framework with server push capabilities in addition, however doesn’t limit to use any 3rd party Ajax wramework. WebBee doesn’t replace any EJB container and actually targets POJO. WebBee doesn’t dictate any patterns for adding and initializing services like inversion of control. WebBee stays on reducing amount configuration files and utilizing declarative programming based on Java annotations. WebBee is transparent to use any container based services. When WebBee is combined with TJWS web container, Java recompilation and restart of application so fast, that config based approach of customization can hardly compete with code changes and apply.

Getting started

WebBee itself is wrapped as ready to use web application. It requires only one simple table created on backend. The table has to have three columns id, username and password. Id can be defined as auto increment column.

insert into users (login,id,password) values ('username',3,'password')

A data source to this table has to be configured with name ‘jdbc/mysql/webbee-app’ in application server. There is no restriction to use MySQL database server, and name of data source can be changed accordingly application server requirements as context parameter with name model\_datasource in web.xml

Download

Please use the link

WebBee Web application development technique

Every development platform specifies own rules of development applications. Webbee isn’t an exception. So this guide covers major steps required for successful and rapid building of web applications. Webbee building blocks are based on Aldan 3 framework. So application development looks very similar. Every application is built on Front Controller and MVC design patterns. Although all views may have separate models, application itself has to define master application data model. Models staying behind of particular views are defined as Java objects with JDO annotations. So development process itself has very good natural separation of views, models and controllers. It helps in collaborative work on application, when every developer can work on own part without interfering with others.

Building application canvas

WebBee recommends certain code source and deployment organizations. Actually every modern development platform dictates certain project file organization, like Android or iOS/XCODE. The following directory structure is recommended for WebBee projects

Project home

bild - build scripts

out – binaries and deployment archives

doc – project documentation, can be additionally structured

src – project sources

res – project resources

cfg – config and deployment descriptors

css – CSS files

html – static HTML resources

image – images, can have additional directories

text – text resources, can have subdirectories by locale name

view – templates and other view related resources, can be structured

java – Java source tree

model – database scripts

js – JavaScript code

Deployment directory structure

WebBee uses standard war/ear deployment structure with possibility of detaching not protected resources as static HTML, CSS, images and scripts and serviced by different server for better scalability.

war context name contains HTML on first level

css – CSS files

js – JavaScript files

image – Images

WEB-INF web.xml

classes - project Java classes tree

lib – jar libraries

config – config files

res – root resource directory

view – view related resources eg templates, additional directories structure can be inside

text – text resources, additional locale specific structure can be inside

Preparing directory structures, build and config files

web.xml

Webbee requires just few elements in web.xml

context-param model\_datasource defines JNDI data source name, unless it isn’t used.

Listener several listeners can be specified, however app model class is required to be specified as one of listeners.

Aldan 3 servelt inherited WebBee servlet as com.beegman.webbee.base.WebApp has to be specified as main and only one servelt of application. The former can be replaced by some inherited one even for more customization. There is a possibility of use multiple servelts, however it won’t be covered by this guide.

smtp\_configcontext parameter has to be set for smtp properties file location.

Application config file

The file has format of standard Java properties file. Application model or just base block gives access to the property file, so entire application can read or modify the file. However modifications are not persistent. Name of config file usually matching application context name. The file location is specified as application servelt parameter properties. The config can be requested from application model using method getBaseConfig. Since app model can get initialized prior of servlet, this call can return null until servlet didn’t finish initialization. Services based on this config file have to aware of that. Any config property can be requested from templates, so it is a way of customization, for example config property copyright can refer to copyright note HTML insert location. Read more information about some properties of the file below.

Building tool

7Bee building tool and corresponding scripts supplied with WebBee can be used for population initial directory structures and follow build abilities. Scripts have to be modified to cover any application specific resources all 3rd parties’ libraries.

Application model

Any Webbee based application h as to provide own implementation of application model class extending com.beegman.webbee.model.AppModel. The implementation class has to be also specified in web.xml as listener, see above. The following methods should be overridden for purpose:

getAppName – returns name of application which is used for generation many unique ids.

getServletName – it is very important to keep this name in sync with app servlet name specified in web.xml, for example if web.xml defines servlet as: <servlet-name>webapp</servlet-name>, then the method has to return “webapp”.

initServices – has to be overridden if more application services need to be initialized and started. Base implementation runs only AsyncUpdater service, see more details below. So do not forget to call super implementation if the service is needed for your application. When any new services added, another method has to be overridden, it is deactivateServices, see more details below.

deactivateServices – override this method if new services were instantiated in initServices and need to be destroyed or deactivated at end of application or when application is temporary unloaded. However do not design your services in way if calling the method was skipped for some reason, some system resources will be consumed forever.

More methods can be overridden to customize application behavior, for example tabs or quick access links. Although the former ones can be configured just modifying resources.

A developer can add more methods in base app model as needed. Since the model is available for all parts of application it can be used for data exchange.

Managing user sign on

Application model getAuth has to be overridden to return instance of Auth object used for user authentication with possibly following authorization.

Application config parameter public\_home specify a page to redirect when session expired.

Multi languages support and resources

User locale is always available in page services, and all resource loading get verified against current user locale. So all application text constants have to be stored in resource files for easier localization if applied.

Standard resources

There are several standard resource files which suppose to be supplied for normal app functioning.

commonlabels.properties – contains labels shared by all views

navigationtabs.properties – contains navigation tabs definitions

fastaccessbuttons.properties – contains fast access buttons definitions

View resources

View resources are templates, JSP or other renderer type resources. WebBee encourages using only template based view resources.

Resource naming convention

All resources related to a particular page service have to carry page service name with corresponding to resource type extension. Text resources have extension .properties. View related resources have extension helping to pick correct view renderer approach. Since base WebBee view renderer is template engine, template resource extension can be configured. Parameter viewext in application properties file.

General application UI layout

Webbee softly dictate certain UI layout based on common practice of creation web application UI. So entire screen consists of the following areas

Top tool/info bar with fast access buttons and user name stationary

Navigation / menu bar

Breadcrumbs bar (optional)

Left navigation panel (optional)

Work area

Status bar

Standard UI layout is defined in canvas template. The template can be changed to provide any other layout or customize existing one.

navigationtabs.properties

Used for defining tabs labels and links. Property name is used for ordering tabs in alphabetic order. Value is comma separated values with page service name, label, and description correspondingly. Commas have to be HTML encoded. Only first value is required.

fastaccessbuttons.properties

Used for defining fast access buttons appearing in top line right side. Syntax is the same as for tabs.

Standard templates(views) wrappers

canvas.nnn

This wrapper defines general layout of application UI. If it isn’t planned to customize it, then include to Webbee standard application layout can be provided there, like

@%'canvas.htmt'@

Insert/copyright.htmt

Standard copyright HTML template fragment. Name and location of this file is defined in application property file as property copyright. A developer can customize as location as content of the file.

Insert/headcss.htmt

Any custom css can be added in this template, as direct style definition, as included CSS file.

Insert/headextra.htmt

Extra head content can be added to this file, ofr example JS files for 3rd party Ajax libraries.

Insert/headjs.htmt

Another way to include extra JS files. Can be specified in this template.

Enabling server side push

A web push handler URL has to be specified and also requester loop started. This code can be added in initFields handler as below:

<script>

function initFields() {

handlerUrl = Status/ajax/Asyncupdate';

// run it async

updateUI();

}

</script>

Since Asyncupdate' is available in BaseBlock, a developer rarely needs to override it. However getAsyncupdateViewName may need be overridden especially if different than ‘htmt’ template extension is used. Another solution can be defining asyncupdate\_json with project template extension providing inside same content as asyncupdate\_json.htmt. Or make content of the file directly pulling JSON template as below:

@%'insert/eventqueue.json'@

Another method which can be considered to be overridden is getUIID. This method return session id by default, but sometimes you may need something else especially for application where session isn’t supported. updateUI initiates push content loop. The former function is provided in ui.js, so assure this script is added once in involved templates as below

<script src="@contextpath@/js/ui.js" language="Javascript"></script>

Beefing with data objects

All WebBee data objects have to be inherited from SimpleCoordinator, it gives access to application model from data object and can be used for requesting additional information from common data model required for normal functionality of data object. Fields of data object have to be defined as public and JDO and form presentation annotations can be added to provide as automatic persistence of data object as automatic form rendering built of the data object.

Page service class

Any user interaction with application happens through page service class. The class gives access to a page model which is rendered by view, and the class can also process control events and update corresponding model as needed. All page service classes have to be stored in the same package and name of this package has to be defined in PresentServicePackage config parameter. Such classes can be dispatched by Front Controller. Page service class must be inherited from BaseBlock or from corresponding functional block.

Adding views

A view resource or some other active renderer component as JSP has to be added with corresponding page service name in corresponding to renderer type location. Normally views are stored in template resource folder which location is defined by TEMPLATEROOT configuration parameter. Controls are part of page service classes

Adding background processes

All background processes have to be wrapped in services and instantiated and started in initServices of application model class.

Final wrap

A final application will require most of resources from WebBee base application, so is recommended to copy all these resources blindly. However to avoid security risk, it I s recommended to redefine as default view extension as page services package name. So if application requires some template resources from WebBee base packaging, then such resources can be renamed with required extensions, or resource delegation approach (as include) should be used. Former provide better application upgradability to next versions of WebBee.

Webbee base blocks set

Signon

Inherit page service class which is responsible for signing on a user in system.

Useprofiler – view/edit user profile

Userregistration – user registration service

Userlogin – user login service

Tabular – service for tabular report presenatation

Form – service for forms

Message – service for delivering messages