CarrierWave: Object Model

Draft 0.1

September 17, 2002

Chris K. Wensel

1.0 Objects

1.1 Semantic Model

1.1.1 Server Semantic Model

1.1.1.1 Base Types

1.1.1.1.1 Imageable

The common interface all business types implement if they are to be shared in the client model. A class implementing Imageable will result in an Image sub-class on the client side.

1.1.1.1.2 ImageableAlias

The common interface all 'orthogonal' or 'tag' business interfaces implement if they are to be shared in the client model. An interface extending ImageableAlias will result in an ImageAlias sub-interface on the client side. Only mutators specified by @image tags will be included on this interface.

1.1.1.2 Specialized

1.1.1.2.1 ImageableIdentifiable

This interface designates any server type implementing it as having a durable identity and any lifecycle events are delegated to the `PersistenceRoot` class to be managed by any registered persistence system.

1.1.1.2.2 ImageableFinder

1.1.1.2.3 ImageableObjectFinder

1.1.1.2.4 ImageableCollectionFinder

1.1.1.2.5 ImageableAction

1.1.2 Client Semantic Model

1.1.2.1 Base Types

1.1.2.1.1 Image

1.1.2.2 Specialized

1.1.2.2.1 Finder

1.1.2.2.2 ObjectFinder

1.1.2.2.3 CollectionFinder

1.1.2.2.4 Action

1.1.3 Support Types

1.1.3.1 Icon

1.1.3.2 ImageGraph

1.1.3.3 GraphPlan

1.1.3.4 GraphNode

1.2 Imageable Meta-Tags

1.2.1 Contained in Class Comments

1.2.1.1 Class Scope

1.2.1.1.1 image-implements [interface name]

1.2.1.1.2 image-include-constructors

1.2.1.1.3 image-common-type [imageable type]

Used on ImageableAlias sub-interfaces only.

1.2.1.2 Field

1.2.1.2.1 image-field [field name] [class type]

1.2.1.2.2 image-contained-type [field name] [class type]

1.2.1.2.3 image-readonly [field name]

1.2.1.2.4 image-derived [field name]

1.2.1.3 Examples

```
/**
 * @image-include-constructors
 * @image-field someStrings java.util.List
 * @image-contained-type someStrings java.lang.String
 * @image-readonly someStrings
 */
public class BusinessObject implements ImageableIdentifiable
```

1.2.2 Contained in Field Comments

1.2.2.1 Field Scope

1.2.2.1.1 image-contained-type [class type]

1.2.2.1.2 image-readonly

1.2.2.1.3 image-derived

1.2.2.2 Examples

```
/**
 * @image-contained-type java.lang.String
 * @image-readonly
 */
private List someStrings;
```

2.0 Functions

2.1 Type Mapping

2.1.1 Class

2.1.1.1 Image to Imageable

Provided as convenience since every Image subtype knows whom its server side alter ego is.

2.1.1.2 Image interface to Imageable

Provides complete coverage allowing all type to be mapped to an Imageable type.

2.1.1.3 Imageable to Image

Used during graph node tree generation.

2.1.1.4 ImageableAlias to all Imageable types and common Imageable type

Used for creating a GraphNodeSet at a node typed by as the ImageableAlias. Only useful when creating static graphs sourced by type.

2.2 GraphNode tree generation

2.2.1 Instance

2.2.1.1 By Imageable and optionally depth

2.2.1.2 By Image and optionally depth

2.2.2 Static

2.2.2.1 By Imageable/Image type and depth

3.0 Appendix A – Future Additions

3.1 Semantic Types

3.1.1 Shared

3.1.1.1 ImageableFrame and Frame (ImageableFramed and Framed)

3.1.1.1.1 Allows namespace partitioning between deployments.

3.1.1.1.2 Note: somehow this is confusing considering runtime and dev time. Need to elaborate the role of a Frame or if its two things.

3.1.1.1.3 Former javadoc

ImageableFrame instances are objects that partition applications into name-spaces.
They are not intended for grouping or as collections in a single application, but allow multiple applications that share a common code base to co-exist in a single process space. That is, object names are scoped by the frame instance they belong to so that there will be no name collisions between applications deployed on a particular server instance.

3.1.2 Server

3.1.2.1 ImageableComposite and ImageableElement

3.1.2.1.1 Allows a small graph to be collapsed into a single Image type

3.1.2.1.2 Delegates compose, recompose, and decompose requests to ImageableComposite instance

3.1.2.2 ImageableObserver

3.1.2.2.1 Notification of “updates” and etc.

3.1.2.2.2 Former javadoc

The Imageable types also implementing the ImageableObserver interface will receive notification that the object graph it is a member of has completed a “modify”.

For example, Imageable types can be used as method objects that initiate an action on the server side when all their children have been modified.

Currently the “modify” completed event is the only applicable event. This interface can be modified to handle other event types by adding/renaming appropriate event methods.

4.0 Appendix B – Design Decisions