Document Title:	caGrid 1.0 Portal Design Document
Status:	Draft for community review
Author(s):	Colin Freas, Paul Kennedy, Chad La Joie
Publish Date:	
Change Log:	

Table of Contents

General Architecture

The portal is composed of a front end user interface and the backend that feeds it data. The front end is constructed with the Tapestry web application framework. Tapestry is a component based framework that uses the MVC pattern. The portal's components will be fed data from backend objects and can rely on the existence of certain resources, like the caGrid Index Service or a persistence mechanism, in their functioning.

The backend will make use of the open source Spring framework for dependency injection, hibernate and an embedded SQL database for data persistence and caching. The backend will query the caGrid 1.0 index service using WSDL advertised functions.

Component Composition Overview

The portal's components can be classified as one of three types.

Site-wide components provide a specific utility throughout the site. For instance, the “command bar” component and the header and footer components. The functioning of site-wide components is not context dependent: they are essentially static renderings, possibly relying on small bits of configuration from files.

Another set of are more general purpose. These can be customized or extended for a specific context. The RSS feed component, the service or center list component, and the service or center detail component all fall into this category.

The third group of components are special purpose components. These are the single use and page specific. For instance, the only place the mapping component is used is on the home page. The search component is similarly specific.

Employed Technologies

Tapestry 4.0 is used for the front end interface.

GeoTools, an open-source GIS library is used by the front end to render a map of the United States along with specific points, currently representing only cancer centers.

caGrid 1.0 will be based on the Globus Toolkit version 4.x, as specified by other elements of the caGrid development team.

Spring will be used for dependency injection and generalized interface to the Hibernate ORM framework. Hibernate will use Derby, an embedded SQL database.

User Interface

The user interface is based on the Tapestry web application framework. Tapestry is a component based architecture. Components are placed into *pages*, special components which contain other components. Pages are then displayed, typically in a web-browser, and users interact with the application as with any other web site.

The information displayed on the interface is gleaned from interactions with the business and layer layers.

Tapestry Components

Site wide components appear on all, or nearly all, pages within the application.

- Header and footer components

  These will include images, text, and links. Links will potentially point offsite, for instance to the NCICB or caBIG home page.

- Command bar component

  This is one of the primary mechanisms for navigating the site, and includes links to the major activities available in the application.

General purpose components are configurable for various contexts, and can appear in different guises throughout an application.

- PortalObjectList

  This component iterates over a collection of PortalObjects. This collection is provided by a PortalObjectsCmd object. The PortalObjectsCmd object is referenced using the PortalObjectList's getPortalObjectsCmd method, and is stored in the portalObjects collection. Prior to reference the PortalObjectsCmd object, the setPortalObjectsCmd method of PortalObjectList is called via a dependency injection specified in an external configuration file.

  The PortalObjectList will accept a parameter instructing it to render the collection as either a list of centers or a list of services.

- PortalObjectDetail

  This component accepts a single PortalObject and a parameter instructing it to render the PortalObject as a center or a service. The PortalObject's attributes are then rendered appropriately.

- RSS feed component

  The RSS feed component, when provided with a DOM object, will render the content as HTML using a compiled XSLT transformation.

Special purpose components may or may not be configurable, but they display the same information regardless of context. The distinction between these and general purpose components is somewhat arbitrary, though useful.

- Map component

  The map component is used to render a map with cancer center locations and statuses on it. It displays the centers by iterating over a collection of centers provided by a call to PortalObjectsCmd.getCenters(). The command object of the map component will be set up using dependency injection.

- Search component

  The search component will create a CommonServiceMetadata object and a string to set search criteria. The search component also contains accessor methods for the SearchObjectsCmd object. This command object provides access to the doAdvancedSearch and doBasicSearch methods. doAdvancedSearch accepts the string and the CommonServiceMetadata object as parameters, while doBasicSearch accepts only the string.

  The doBasicSearch and doAdvancedSearch methods will each return a collection of CommonServiceMetadata. That collection will be iterated over to create a rendering of search results.

Pages

All pages will have the header, footer, and command bar components.

- Map page

  This page will utilize the the RSS feed component in conjunction with a portal configuration parameter designating the grid level RSS feed source. It also contains the map component.

- Center list page

  This page uses the list component configured to display centers.

- Service list page

  This page uses the list component configured to display services.

- Search page

  This page uses the search component, which allows you to toggle between advanced and basic search mode.

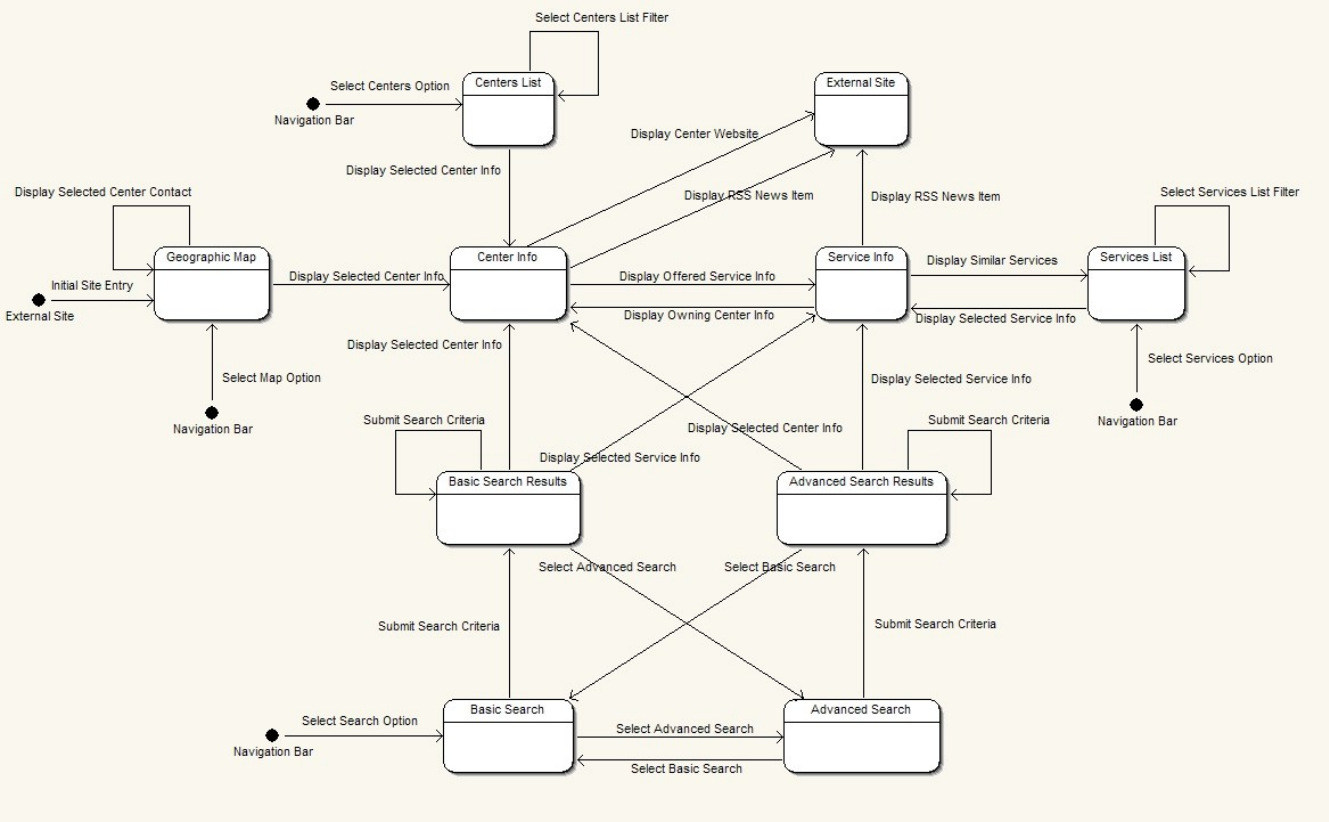
- Center detail page

  This page consists of the RSS component using the center's RSS feed URI, and the detail component, pointing to the particular center.

- Service detail page

  This page contains the RSS component using the service's RSS feed URI, the detail component rendering a specific service's details, and a component to render the uptime graph for the service.

Page Flow Diagram



Application Code

Design Overview

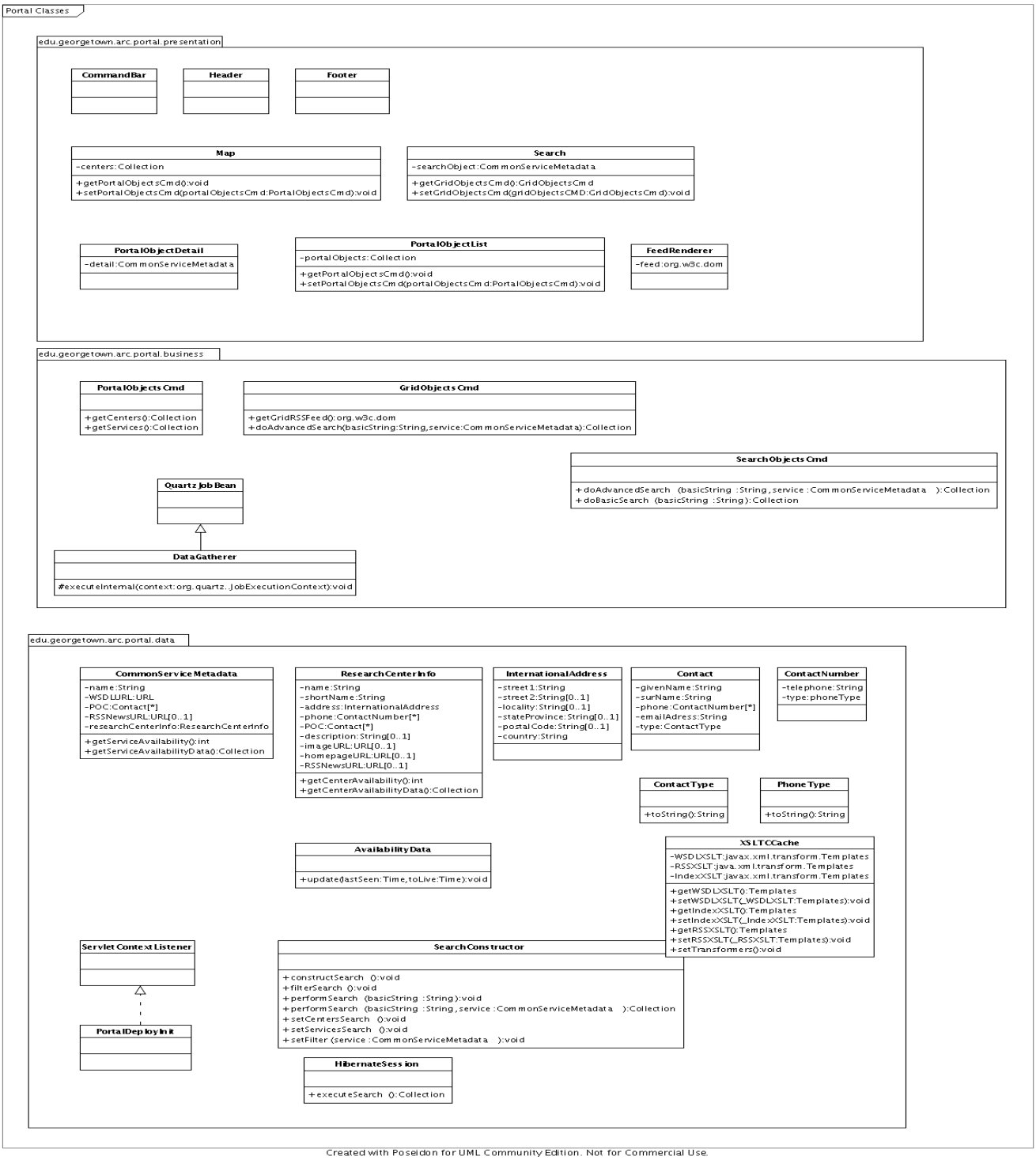
Spring will handle the connections between the portal presentation layer and the interface to the business layer through dependency injection. Presentation layer objects will be provided references to the command objects which form the interface between the presentation layer and the business logic layer.

Spring will handle dependency injection in the business layer, by providing all business layer objects with references to data layer objects.

At the data layer, Spring will also be used to configure Hibernate's SessionFactory as well as locate and load all necessary ORM XML files. The database connection will be abstracted through the standard Data Access Object (DAO) pattern. These pieces comprise the persistence layer for the portal.

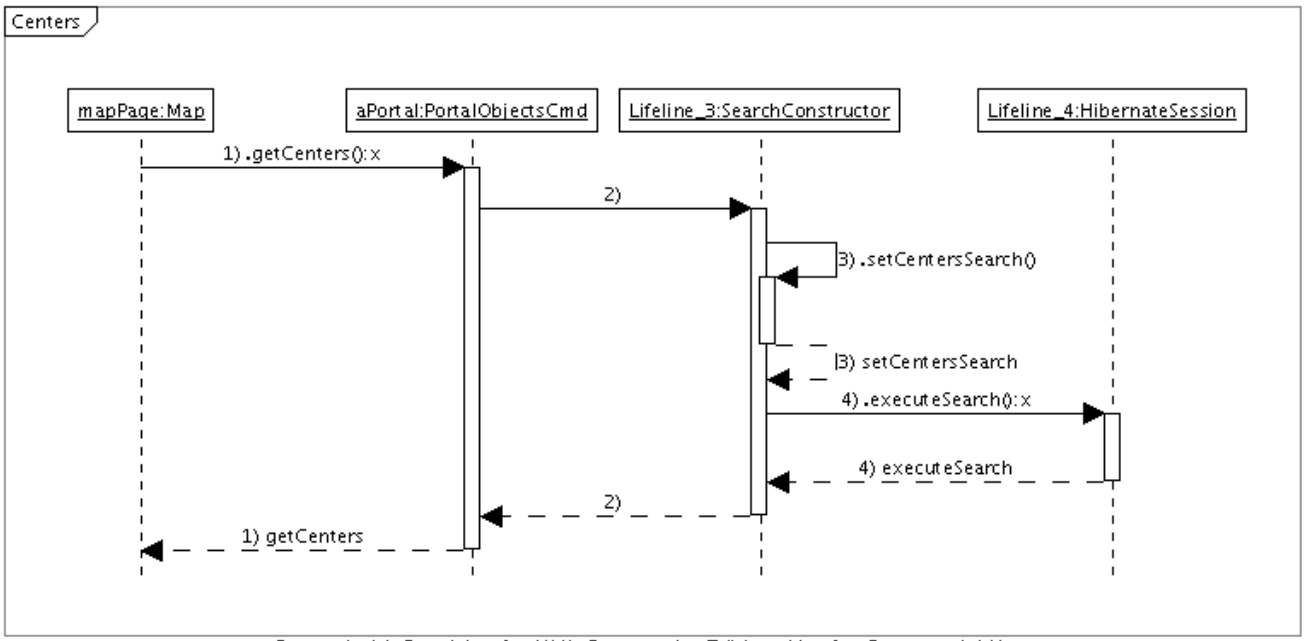
Finally, at application deployment and initialization, the embedded database will be initialized and any XSLT files (for transformations of WSDL, RSS, or IndexService content) will be compiled to XSLT. A QuartzJobBean will also be initialized to provide for periodic querying of the caGrid1.0 index service to feed data to the data layer. A second QuartzJobBean will be initialized to handle RSS caching and persistence.

Class Diagrams



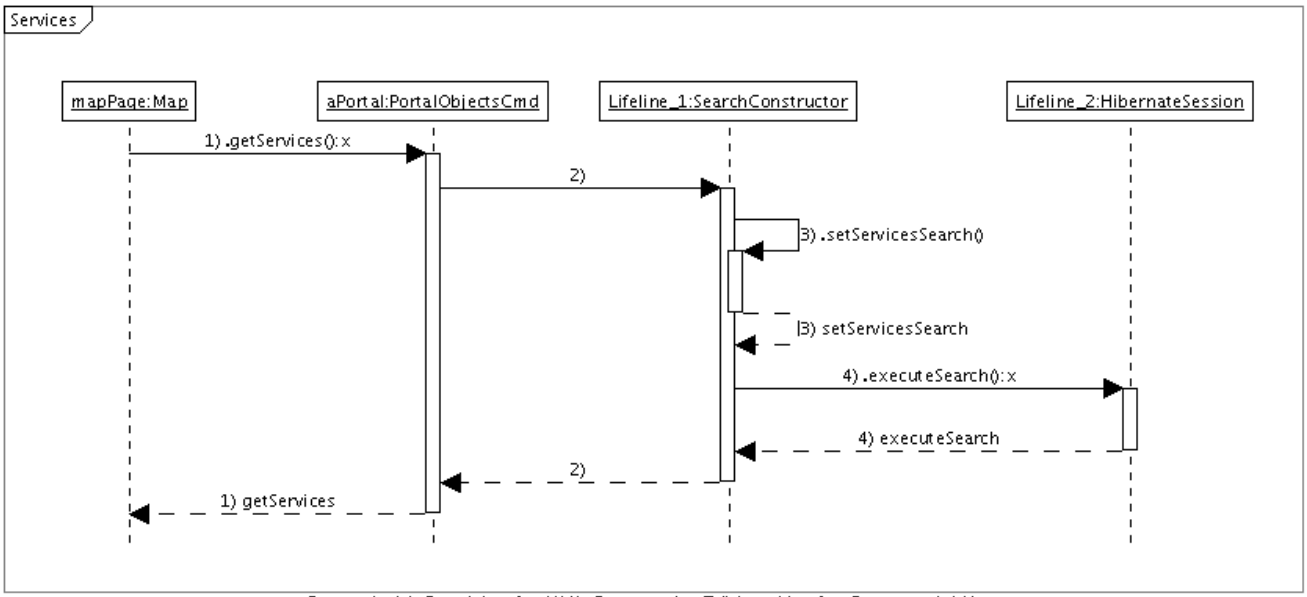
Use Case Sequence Diagrams

Centers list:



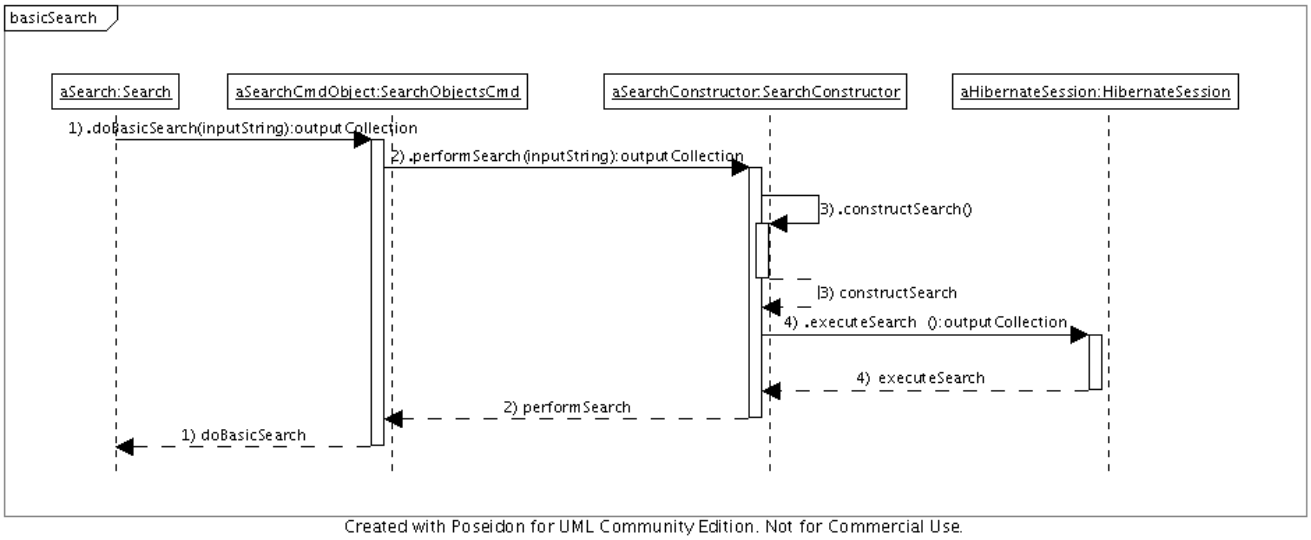
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Services list:



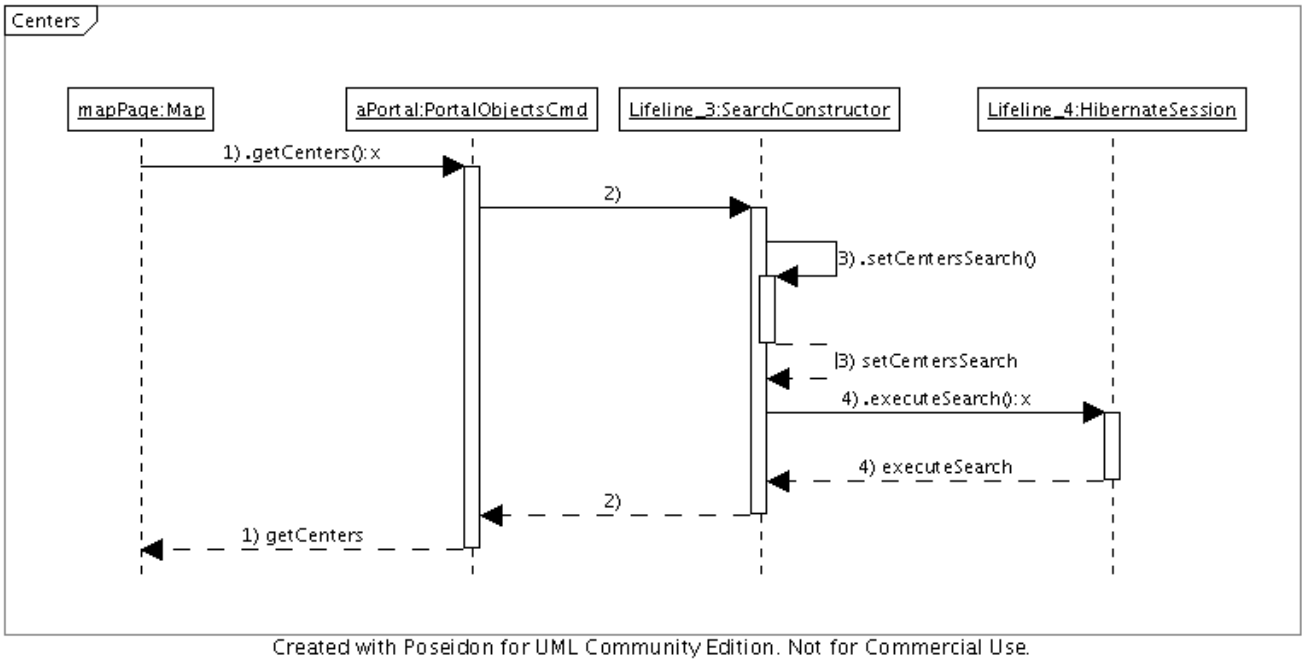
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Basic Search:



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Advanced Search:



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Database Design

The portal will use Apache's Derby embedded database. Communication with the database will be configured using the dependency injection capabilities of Spring. The mapping between the objects used in the portal and the representation in the database will either be provided as hibernate hbm.xml files or written as XDoclet in the source code. Spring and Hibernate will allow the loading of the mapping files and the resolution of objects.

A cache for service and research center information will be kept using persistent objects. The portal will aggregate “time last seen” and “time to live” for caGrid services as advertised by services and aggregated by the caGrid 1.0 index service. RSS feeds will have a cached and persistent form. This allows announcements to be available during maintenance periods and network outages.

A description of the database tables is included:

ServiceToContact	
serviceID	int
contactID	int

URL	
URLID	int
URL	varchar

CenterToContact	
centerID	int
contactID	int

CenterToContactNumber	
centerID	int
contactNumerID	int

ContactType	
contactTypeID	int
contactType	varchar

ContactNumber	
contactNumberID	int
phoneTypeID	int
telephone	varchar

CommomServiceMetadata	
serviceID	int
name	varchar
WSDLURLID	int
RSSNewURLID	int
researchCenterID	int

InternationalAddress	
addressID	int
street1	varchar
street2	varchar
locality	varchar
stateProvince	varchar
postalCode	varchar
country	varchar

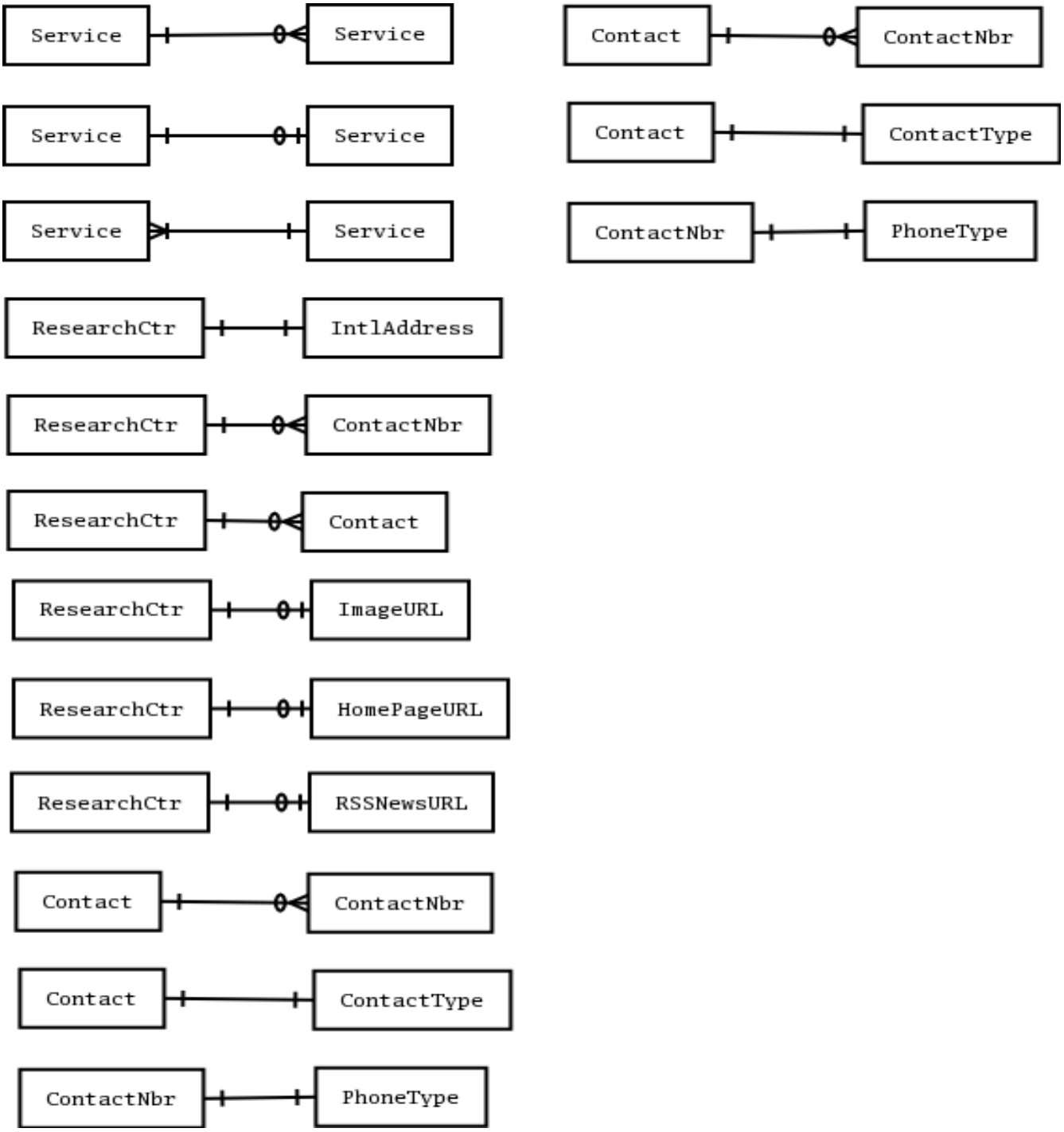
Contact	
ContactID	int
givenName	varchar
surName	varchar
phone	varchar
emailAddress	varchar
contactTypeID	int

PhoneType	
phoneTypeID	int
phoneType	varchar

ResearchCenterInfo	
centerID	int
name	varchar
shortName	varchar
addressID	int
description	varchar
imageURLID	int
homepageURLID	int
RSSNewsURLID	int

RSSNewsCache	
cacheID	int
RSSNewsURL	int
cacheObj	varchar

Database ERM Diagram


Service
Service
Service
Service
Service
Service
Contact
ContactNbr
Contact
ContactType
ContactNbr
PhoneType
ResearchCtr
IntlAddress
ResearchCtr
ContactNbr
ResearchCtr
Contact
ResearchCtr
ImageURL
ResearchCtr
HomePageURL
ResearchCtr
RSSNewsURL
Contact
ContactNbr
Contact
ContactType
ContactNbr
PhoneType

Build and Deployment

Build and Testing Process

Building of the portal codebase will be automated through the use of the ant build tool.

All classes will be tested through the use of unit testing frameworks (Junit, EasyMock, jMock, ...) and will be constructed as a separate build target in the ant build.xml. The architecture and use of Spring for dependency injection facilitate loose coupling and ease testing of components.

Deployment Process

Portal configuration items will be stored in the WEB-INF directory. The entire application will be built as a WAR file for easy deployment in a J2EE container.