PROJECT 3: Due: 12/11/2005 11:59 PM

DESIGN AND DEVELOPMENT OF A UDDI REGISTRY

This project requires you to design and develop a basic working version of a *Universal Data Discovery and Integration* registry. The registry must be designed as a web-service, and must have a set of API method that can be invoked remotely.

In preparation for this lab, go over the UDDI content covered in class, revise content on web services, and revisit the JAX-RPC examples we used for project 1. Also read up on *SAAJ* (or Soap with Attachments API for Java), since we will be implementing our UDDI registry using this API.
Resource:

`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/SAAJ.html#wp69380`

Storage Requirements:

For the purposes of this project, your CSE Oracle 9*i* accounts will be used to store registry data. You will be required to design, create and populate tables in order to store data in relation to the 4 data types:

- *businessEntity*
- *businessService*
- *bindingTemplate*
- *tModel*

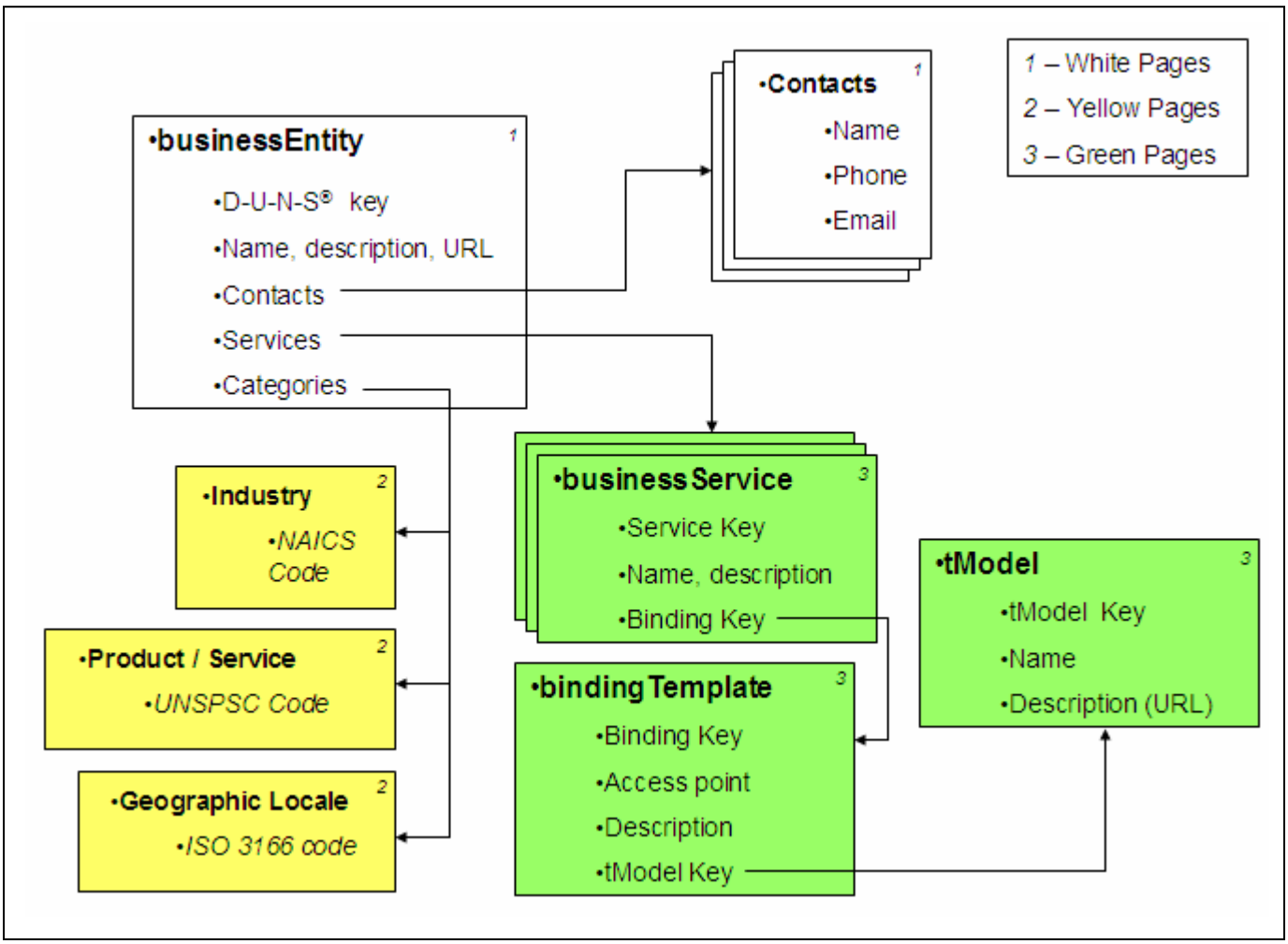
These data types can be classified further into *white, yellow,* and *green* pages as follows:

- **White Pages:** Information pertaining to the *businessEntity* is store here. You will need to store information such as:
  - A D-U-N-S® (Data Universal Numbering System) unique identifier for the business (this can be your primary key). [*ref.* `http://en.wikipedia.org/wiki/D.U.N.S.`]. You can arbitrarily assign a unique value to each new registrant.
  - Business name, description, a URL to find more information about that business
  - Contact name(s) (a business may have *more than one* contacts)

- **Yellow Pages:** This provides a *taxonomy based categorization* of businesses. This Information is also part of the *businessEntity*. We will consider 3 taxonomies
  - *Industry classification* based on *NAICS* code specifications. These specifications are extensive, so just choose a very small subset, maybe 3 or 4 categories. [*ref.* `http://en.wikipedia.org/wiki/NAICS`]
  - *Product/Service classification* based on *UNSPSC* code specifications. Again, these specifications are extensive, so select a small subset for your project. [*ref.* `http://en.wikipedia.org/wiki/UNSPSC`]
  - *Classification on Geographic Location* using the *ISO 3166* structure. Again, choose a subset for your project. [*ref.* `http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2`]

- **Green Pages:** This provides a description of the services offered by business entities (*businessService*), information on how to access those services (*bindingTemplate*), and technical information on how to access that service (*tModel*).
    - *businessService*: this data structure should provide details such as name and description of the service offered by the various businesses in the *businessEntity* data structure. Remember that a business may offer one or more services
    - *bindingTemplate:* this data structure provides details regarding the method in which a service may be invoked. Information this data structure holds includes a description of the business service, and an *access point* (i.e. an *end point URL* in the case of *JAX-RPC* services). A *bindingTemplate* is also associated with a *tModel*
    - *tModel:* this data structure provides technical information regarding how that specific service may be accessed. It will hold information such as a *model key, name,* and *description* of that model which may just be a URL of a website where more information can be gathered about that particular technology.

The diagram below shows the relationship between the various data structures:



API Requirements

Your UDDI registry must be implemented as a web service, using *SAAJ*. The following remote methods must be made available to other businesses or service that will access your *UDDI registry service* as *clients.*

1. **Inquiry API**
   a. *Find* **Information**
      i. `find_business`
      ii. `find_service`
      iii. `find_binding`
      iv. `find_tModel`
   b. **Obtain** *detailed* **information**
      i. `get_businessDetail`
      ii. `get_serviceDetail`
      iii. `get_bindingDetil`
      iv. `get_tModelDetail`

2. **Publisher API**
   c. *Save* **Information**
      i. `save_business`
      ii. `save_service`
      iii. `save_binding`
      iv. `save_tModel`
   d. *Delete* **Information**
      i. `delete_business`
      ii. `delete_service`
      iii. `delete_binding`
      iv. `delete_tModel`

Web-based client

Implement a single web-based interface. This interface should be able to invoke the various API calls perform the following fuctions:

1. Register (or delete) businesses (along with their contacts, services, and binding templates), and *tModels*.
2. Search for specific businesses and their associated services based on keywords, NAICS, UNSPCS or location codes, obtain, and display such information.
3. Obtain information on how to access specific services.
4. Obtain information on various tModels.

Note

You are not required to implement the four *detailed information* API calls, and may implement only the first of the four *delete information* API calls.

Submission Details:

Will be provided as the deadline nears…