The Livespace Bus

Part 2 – API

Under The Hood

- Entities are modelled as Java objects
  - We call these *databeans*
- Published by putting them into a *container*
- *Server* containers publish entities
  - Like putting them in a database
- *Client* containers find them
  - Like a live query

Under The Hood

- Clients have a replicated copy of discovered entities
- This copy is can be changed
- Changes on client and server are asynchronous
  - Clients do not block waiting on a server
- Will expand on this in next section

DataBeans

- Extends JavaBean concept
  - Also resembles KVO on Mac OS X
- Key features of JavaBeans/DataBeans
  - Discoverable properties
  - You can listen for property changes

JavaBean Example

```java
public class PersonBean
{
  private String name;
  private int age;
  private List<Address> addresses;
  ...

  public String getName ()
  {
    return name;
  }

  public void setName (String newName)
  {
    String oldName = name;

    name = newName;

    firePropertyChanged ("name", oldName, newName);
  }
  ...
}
```

Differences from JavaBeans

- Can monitor nested beans – property events percolate down
- Supports dynamic properties
- Encapsulates collections – Set's, Map's, etc are DataBeans* too
- No setter/getter/property event boilerplate
- All properties available via `setValue ()` and `getValue ()`

* Almost: this is actually a convenient lie

DataBeans

- A data bean contains *data only*
- No behaviour, no methods
- Enables sharing across VM, language & machine boundaries

DataBean Example

```java
public class PersonDataBean extends SimpleDataObject
{
  public String name;
  public int age;
  public List<Address> addresses;

  ...
}

PersonDataBean person = new PersonDataBean ();

person.setValue ("name", "foobar");
System.out.println ("Name is: " + person.getValue ("name"));
System.out.println ("Age is: " + person.age);
```

DataBean Events

```
person.addPropertyListener (new PropertyListener ()
{
  public void propertyChanged (PropertyEvent e)
  {
    System.out.println ("change: " + e);
  }
});

person.setValue ("age", 42);
person.addresses.get (1).setValue ("street", "Sesame Street");
```

Prints:

```
change: path: "name" old: 32 new: 42
change: path: "addresses/1/street" old: "" new: "Sesame Street"
```

End Of Part 2

Exercise – DataBeans

- Setting up environment
- Simple use of a DataBean