# Notes on Kuali Student and Fluid

## Introduction

The user interface design project of Kuali Student consists of

1. Defining Portal-portlet interactions. Ie what parts of KS are presented in the Portal?
2. Identifying existing best practices around standard widgets. This information should be gleaned directly from existing best practices manuals.
3. Defining a set of visual stereotypes
4. Internationalization
5. Code templates for the visual stereotypes. When these templates are completed they will include::
    a. Sample code that can be executed
    b. Documentation
    c. Diagrams

## Kuali Student and Fluid

### User experience design guides

Kuali Student teams should study existing publications on user experience.  This applies to both style guides and more  general works on user  experience.  We should consider choosing some standard works.  For example:

1. ***Microsoft Windows User Experience*** (Microsoft Professional Editions) (Paperback)
2. ***Designing Interfaces: Patterns for Effective Interaction Design***  by Jenifer Tidwell

There should  be standard behaviours for common widgets:

- Buttons (text and graphics)
- Grids
- Tabs
- Progress bars
- Text boxes
- Error messages (pop-ups or in-line text?)
- Date pickers
- Toolbars
- Etc etc

And  recommended ways of deciding which widgets to use.

**Question: what can the Fluid project put together or recommend?**

*Leo Fernig, September 20th, 2007*

**Kuali Student and Fluid**

## Integrating the user experience with the environment of the millennial generation

We should explore ways of integrating the KS user experience with the connected network of the millennial generation:

1. Facebook
2. Chat
3. Texting
4. gmail/hotmail/yahoo

Examples of integration:

1. Timetable to Facebook
2. Texting for items other than emergencies

**Question: is this within the scope of the Fluid project?**

## Exposing Kuali Student in the campus portal

General considerations surrounding the question of exposing parts of KS in the campus portal:

1. Portlets
2. Window states
3. Layout options:
    - Customization
    - Personalization

**Question: is there a systematic way of approaching this?**

## Internationalization

Kuali Student is committed to internationalization in 2 senses:

1. Supporting the rules and processes of any educational jurisdiction
2. Supporting a multi-language user interface

**Are there standard ways of supporting a multi-language user interface?**

## Visualization tools

**Are there areas in Kuali Student where visualization tools would be helpful?**

*Leo Fernig, September 20th, 2007*

## Visual stereotypes

Visual stereotypes cover typical screen layouts. Screens in this sense are visual frameworks for managing a complex piece of work:

1. Registering in a course
2. Deciding on a degree program
3. Paying fees

The screen is a container for a variety of widgets (whose style and behaviour has already been defined in the style sheets: see above). When a development team is developing code for a business domain, the business analyst should be able to select the stereotypes that are best suited for that business function. Here are some examples of visual stereotypes:

1. The workbench stereotype
2. The wizard stereotype
3. The form stereotype

The workbench stereotype is similar to the Eclipse platform and consists of three areas:

1. An explorer
2. A workspace
3. A diagnostic area

The explorer gives the user a bird's eye view of a domain. For example:

1. Subject areas in the curriculum domain (Anthropology, Biology, History etc etc)
2. Sessions/terms for academic records
3. Available queries for the search screen
4. Etc etc

The explorer can have tabs that allow different perspectives for the bird's eye view (tree, list, grid etc)

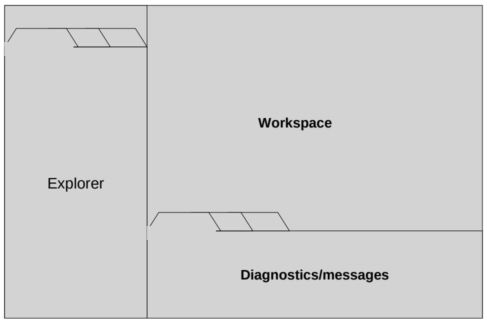
The workspace is where the actual interactive work happens. For example:

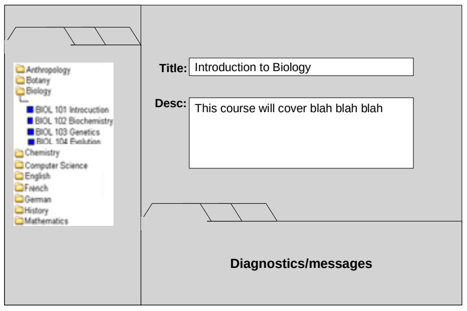
1. Adding courses to a timetable
2. Defining a new course
3. Updating grades for a class
4. Etc etc

**Can members of the FLUID project work with KS (functional and technical folk) on defining a set of stereotypes and building them with technologies that the FLUID project has recommended?**

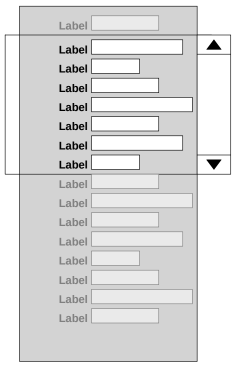
## Kuali Student and Fluid

### The workbench stereotype

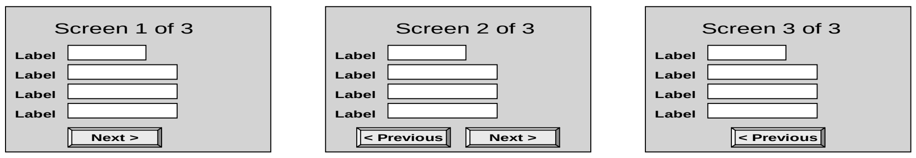




### The form stereotype



### The wizard stereotype

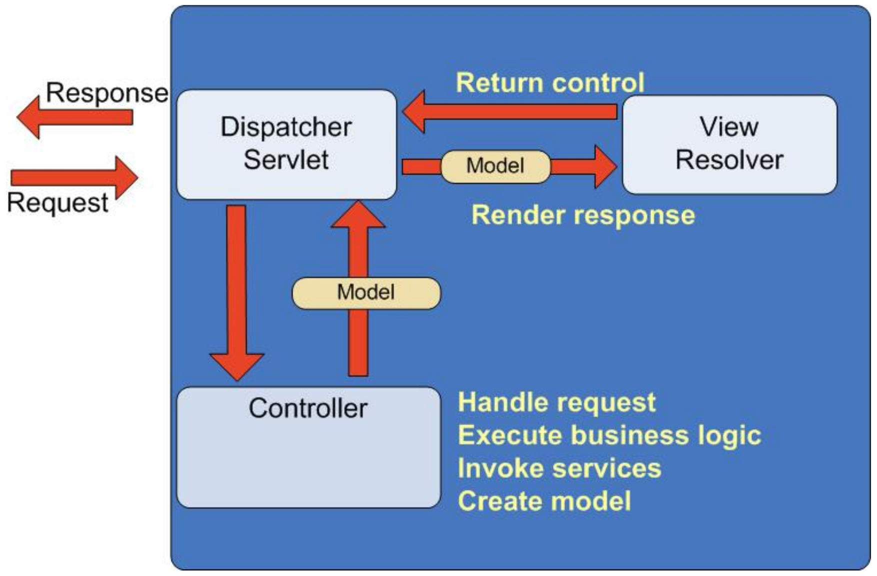


*Leo Fernig, September 20th, 2007*

# Kuali Student and Fluid

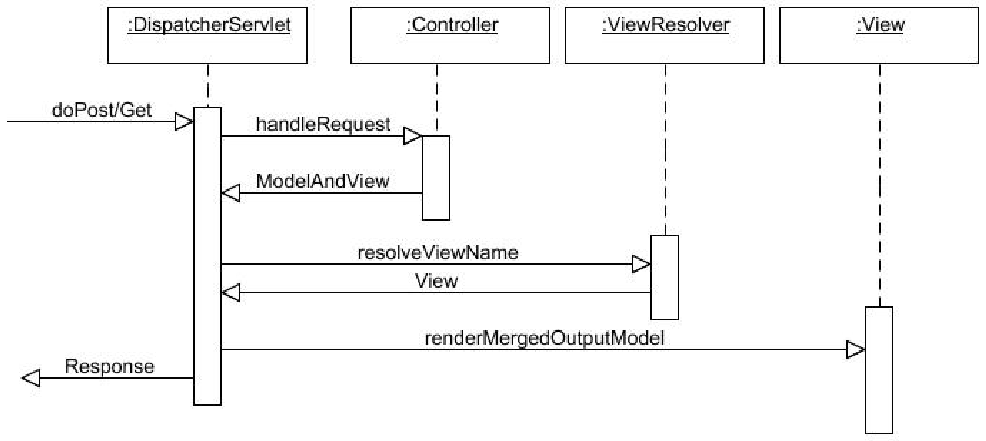
## Code templates for the visual stereotypes

The diagram below illustrates the MVC pattern.



The sequence of events is:

1. The Dispatcher receives a Request (eg "add a course")
2. It is dispatched to the Controller
3. The Controller will execute  any business associated with the request (eg checking for available seats, checking the students pre-requisites etc etc)
4. The Controller gathers the information (Model) that needs  to be returned to the user.
5. The Model is mapped to a view which is then returned to the user (Response)



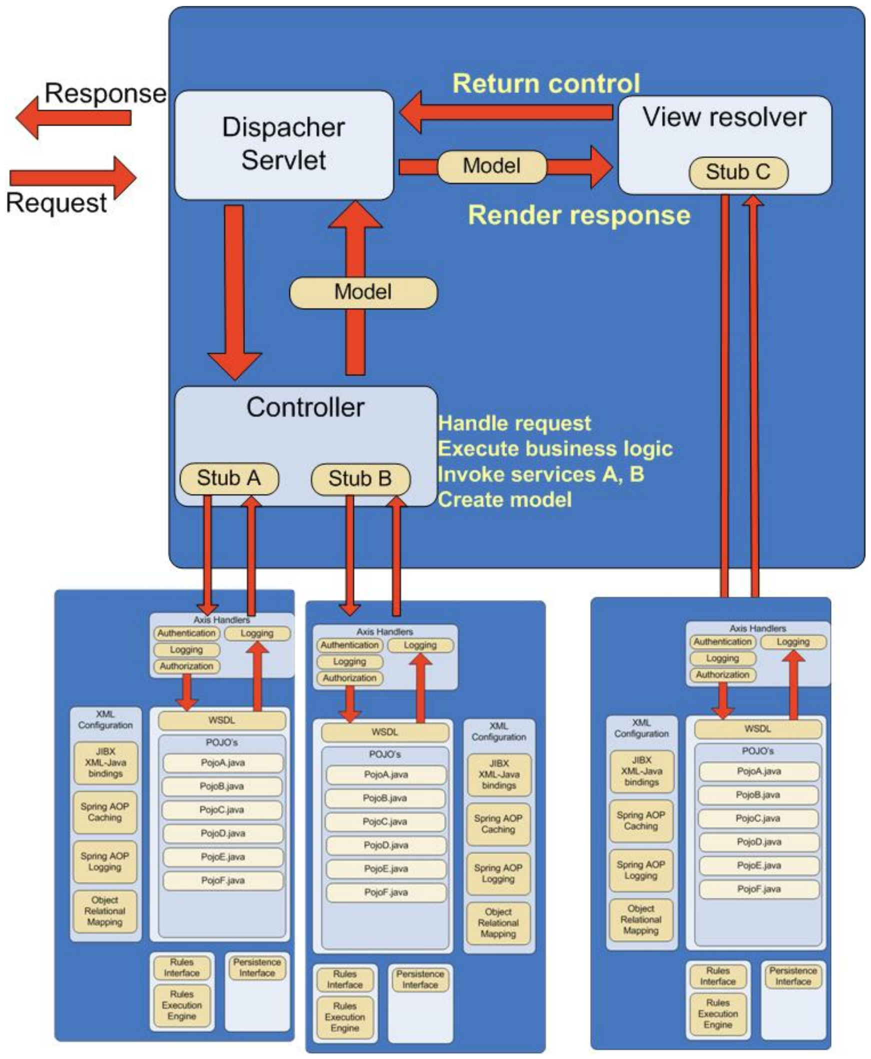
*Leo Fernig, September 20th, 2007*

## Kuali Student and Fluid

In the SOA there are two respects in which the MVC is a little different.

2) The Controller (which acts as a bridge between a user request and business functionality) will be interacting with one or more web-services (as opposed to encapsulating the business logic directly or talking to an EJB).
3) The view template may be built dynamically by invoking services that return UI components (labels, drop-down lists etc)

In the illustration below we see a Controller communicating with two web services. It invokes Service A and Service B via StubA and StubB. The view resolver itself uses a service (Service C) to get certain UI components such as labels and lists



*Leo Fernig, September 20th, 2007*

## Kuali Student and Fluid

The MVC pattern can be implemented with a framework like Spring  MVC.  There are still two problems that need to be solved by the UI stereotypes.

1) How do we integrate Ajax with an MVC implementation (and is that the right approach, or is the MVC component now on the client?)
2) How do  we aggregate UI components on the Portal?

**Can KS use the  expertise the FLUID project already has with Ajax and Spring MVC, Spring Portlet MVC  etc?**

*Leo Fernig, September 20th, 2007*