MSpace ComponentModel 0.1

Néstor Salceda

16 de abril de 2005

1. Propiedades y requisitos que cada componente software debe cumplir:

- Cumplimiento del paradigma de la programación orientada a componentes software.
- Separación clara entre modelo, vista y controlador.
- Independencia entre componentes.
- ¿Variación del comportamiento de un componente según una definición exterior, sin modificar el código fuente?
- Estandarización de la nomenclatura y de la jerarquia de clases interna de un componente.

2. Propiedades de la framework para la versión 0.1:

- Localización de los componentes por un nombre e identificador único.
- Ejecución de los casos de uso de un componente dinámicamente.
- Posiblidad de redirección a la vista:
  - Indicar manejo de la vista.
  - Indicar que método rebe ser ejecutado en la vista como respuesta a un caso de uso.
  - Indicar qué instancia de la vista se debe utilizar.
- Posibilidad de llamar a métodos no bloqueantes.
- Concentración en el manejo de excepciones.

3. Explicación de cada una de las propiedades:

3.1. Localización de componentes por nombre:

Para tener acceso a los servicios que nos brinda un componente se debe conseguir una instancia de dicho componente, para conseguir esta instancia se le pasará el nombre (identificador) de un componente. Una vez conseguida la instancia se provee una interface común para el acceso a los servicios que ofrece el componente.

3.2. Ejecución de los métodos dinámicamente:

Resolución de los métodos a ejecutar en tiempo de ejecución, pudiendo variar el comportamiento de la ejecución de los casos de uso; según los parámetros pasados.

3.3. Redirección a la vista:

Cuando ejecutamos un caso de uso del componente, se puede optar por redirigir a la vista la salida de ejecución de dicho caso de uso. Al igual que a veces no tiene porqué ser necesario el redirigirlo a una vista. Y también se debe poder especificar a qué vista queremos que se le redirija.

3.4. Métodos no bloqueantes:

Se trata de no esperar a que un método finalice su ejecución para poder proseguir con la siguiente ejecución. Un ejemplo claro es la ejecución de un método en segundo plano.

3.5. Manejo de excepciones:

A través de un controlador de excepciones, todas las excepciones producidas de la ejecución de un método serán redirigidas a un controlador de excepciones y desde ahí exisitirá un método común de procesado de las excepciones.

4. Posibles propiedades de la framework en versiones futuras:

- Acceso a los casos de uso mediante roles.
- Ejecución de los métodos y las respuestas de manera transaccional.
- Métodos virtuales.
- Preejecución, postejecución e intercepción de métodos.