UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

Resumo Artigo I
Desenvolvimento de Aplicações Web utilizando o MVC Design Pattern

Autor
Omar Omar

Orientador
Frank Siqueira

Florianópolis, Outubro de 2004

Abstracrt

Este artigo introduz o conceito de *Partição de aplicações web flexível*, um modelo de programação e de infra-estrutura de implementação, baseado no modelo Model/View/Controller (MVC), que aborda a dificuldade encontrada na aplicação deste modelo ao desenvolvimento de aplicações web, visto que em função das tecnologias correntes os desenvolvedores sentem-se impelidos a realizar a partição da aplicação já na fase de design, enquanto que o modelo MVC é independente de partição.

1. Introdução

1.1 O Model/View/Controller Design Pattern

O MVC já é bem conhecido e muito útil para arquitetar sistemas de software interativos. A idéia principal é separar as interfaces de usuário da camada de dados representada pela interface de usuário.

Neste modelo, a camada *View* exibe informações ao usuário, e, em conjunto com a camada *Controller*, que processa a interação do usuário. O *Model* é a porção da aplicação que contém tanto a informação representada pela camada de *View* e a lógica que modifica esta informação como resposta à interações do usuário.

Algumas vantagens deste design pattern:

- O visual da aplicação pode ser drasticamente modificado sem modificar estruturas de dados e lógica de negócio;
- A aplicação pode facilmente manter diferentes interfaces, como multilingüe, o diferentes conjuntos de permissões de usuário;

1.2. Aplicações web e o modelo MVC

As aplicações web, como sistemas interativos de software, podem se beneficiar do design pattern MVC. Contudo, o problema envolvido nesta abordagem surge do fato de as aplicações web serem intrinsecamente particionadas entre o cliente e o servidor.

Naturalmente, os desenvolvedores podem pré-particionar as aplicações, decidindo qual método rodará no servidor e qual no cliente. Uma vez feita a decisão, o modelo MVC poderá ser aplicado tanto à porção cliente da aplicação quanto à porção do servidor. Contudo, tal decisão não é trivial, visto que tais decisões dependem de requisitos de aplicação que freqüentemente mudam conforme a evolução do desenvolvimento, o que torna a partição correta praticamente impossível.

Portanto, aplicar o modelo MVC a um ambiente onde as decisões de partição não são fixas é uma tarefa extremamente difícil e complicada.

Ademais, as decisões de tecnologias a serem empregadas no projeto também determinam em grande parte como a aplicação será particionada.

Em função deste cenário, o artigo introduz o conceito de Partição de aplicações web flexível, para que se possa aplicar o MVC de forma mais natural às aplicações web.

3. Flexible Web-Application Partitioning(fwap)

A independência de particionamento torna o desenvolvimento muito mais flexível, pois possibilita aos desenvolvedores alternativas para encarar as mudanças tecnológicas e de características de projeto da aplicação, e até mesmo mudanças de ambiente e infra-estrutura como congestionamentos na rede.

O problema desta perspectiva é como manter uma independência de partição visto que as aplicações web dependem da localização.

A proposta da abordagem fwap possibilita a adoção da perspectiva de independência de particionamento, onde as aplicações são desenvolvidas e testadas num único espaço de endereço, mas que podem ser implementadas para várias arquiteturas cliente/servidor sem modificar o código fonte da aplicação.

Para conseguir a independência, fwap direciona seus esforços a uma série de problemas difíceis, que envolvam tecnologias de particionamento independente para os componentes individuas de Model, View e Controller do modelo mvc. Além da integração destes componentes em um modelo de programação consistente.

3.1. Arquiteturas *fwap*

- *Single-mvc (smvc)* Funciona como a arquitetura de desenvolvimento fwap. Corresponde ao modelo clássico MVC.

- *Thin-Client* Uma arquitetura de implementação onde o Model e o Controller residem no servidor e geram Views do lado do cliente. Evita sobrecarregar o cliente.

- *Dual-mvc* Uma arquitetura de implementação onde tanto o Model quanto o Controller residem em ambos cliente e servidor. Tanto o cliente quanto o servidor pode também gerar Views para serem exibidas no cliente.

3.2. Trabalho Relacionado

3.2.1. Especificação e geração de View

Para que Views possam ser exibidas em diversas plataformas, elas devem ser geradas de uma forma independente de plataforma. Uma abordagem é utilizar XML como uma linguagem de marcação universal. Contudo, a abordagem deste artigo baseia-se na utilização de API Java para especificar a biblioteca de componentes GUI e construir as Views a partir destes componentes em tempo de execução, utilizando bibliotecas dependentes de plataforma do lado do cliente(Ex. Utilizando HTML para navegadores web).

3.2.2. Especificação e execução de Controller

Controllers devem poder ser executados sem alterações tanto no cliente quanto no servidor para se encaixar na abordagem *dual-mvc.* Quando se compilam controllers para JVM, excluem-se uma série de clientes que não possuem JVM, como PDA´s e Web-phones. Uma alternativa é utilizar Javascript ao invés de Java. Ou também utilizando o Java plug-in, que é o que o artigo sugere.

3.2.3. Frameworks de apresentação Thin-Client

O que diferencia o modelo fwap das demais tentativas de adequação do design pattern mvc é a sua independência de particionamento, visto que os demais esforços neste sentido são dependentes de partição.

Embora aplicações fwap possam ser implementadas na arquitetura thin-client, fwap enfatiza a possibilidade de utilizar a arquitetura dual-mvc, aumentando a performance do lado do cliente(já que alguns componentes model e/ou controller estarão no cliente). Em função disso, fwap procura abordar questões como Sincronização do Model e delegação transparente do Controller, que não são do interesse de frameworks baseados na arquitetura thin-client.

3.2.4. Sincronização do Model

Todas as aplicações que envolvem interação humana levantam a questão da sincronização da camada Model, visto que a escala de tempo de uma interação humana é muitas vezes maior que o tempo de transação de banco de dados.

Portanto, esta é uma questão que deve ser abordada por qualquer sistema que envolva acesso a banco de dados e interação humana multi-usuário, não limitando-se ao modelo fwap.

Há diversas formas de tratar este problema, há locks de arquivos, tabelas, registros e colunas, que possibilitam o trabalho multi-usuário. Outra técnica é a otimização de controle concorrente. Há ainda a abordagem Predizer e Transformar.

A arquitetura dual-mvc ainda introduz a complicação adicional da sincronização de cópias temporárias de registros do banco de dados entre o cliente e o servidor mantido com base na sessão do usuário (Abordado com maior profundidade na seção 5).

3.3. O modelo de programação fwap

A infra-estrutura fwap provê uma interface base que especifica que cada aplicação web baseado no modelo fwap está associada a um model e um view, cada qual acessado através de sua própria API.

3.3.1. View

Views fwap são compostas de um conjunto de elementos GUI, como caixas de texto, botões e radio-botões. Os componentes GUI são especificados através de interfaces. Os componentes são criados, acessados e removidos somente através da API fwap, utilizada pela invocação do método getview da aplicação fwap.

3.3.2. Model

O model de uma aplicação fwap segue os mesmos princípios de View: Os componentes são especificados através de uma interface, e os componentes são acessados somente através de uma API fwap. Os componentes fwap seguem a especificação de componentes Enterprise Java Beans.

3.3.3. Controllers

O controller é uma unidade funcional que pode ser invocada para rodar tanto do lado do cliente quanto do lado do servidor. Usa-se um descritor de implementação de controller para determinar onde este irá rodar. O que contrasta com a abordagem cliente/servidor clássica.

Deve-se especificar o conjunto de métodos que poderão ser flexivelmente particionados. O que poderá ocorrer somente se o método for parte de uma interface específica de aplicação.

Com isso pode-se dividir as responsabilidades do controller entre o cliente e o servidor, seguindo a arquitetura dual-mvc.

3.4. Desenvolvimento

O desenvolvimento de aplicações fwap consiste na disponibilização de um implementação da subclasse de interface da aplicação fwap.

O modelo fwap não prove nenhum suporte adicional para esta questão porque ela é específica de cada aplicação.

4. Implementação atual

4.1. smvc

Os componentes view da arquitetura smvc são renderizados utilizando-se componentes Java swing. E a aplicação é desenvolvida e testada como se fosse uma aplicação stand-alone swing.

4.2. thin-client

Utilizam-se componentes que geram HTML invés dos componentes swing.
Utiliza-se um servidor de componentes como o Tomcat, onde arquivos .class são disponibilizados como entradas servlets no arquivo web.xml.
- Inicia-se uma nova instancia da aplicação fwap que será associada ao cliente;
- Tal aplicação gera a View Inicial, que consistirá de um stream HTML;
- envia o HTML ao cliente.

O controller e o Model não rodam no cliente, que somente visualiza as Views geradas. Daí o nome thin-client.

4.3. dual-mvc

Duas instancias da aplicação fwap rodam:
Uma no cliente e uma no servidor.
Deve-se sincronizar a serialização de componentes das diversas camadas de View, Model e Controller entre o cliente e o servidor. E decidir quais componentes de qual camada irão rodar no cliente e quais irão rodar no servidor, e adequar a invocação dos métodos.

5. Trabalho em andamento

O trabalho em andamento visa enriquecer a biblioteca de componentes de View, visando as arquiteturas smvc e thin-client. Contudo o esforço principal está no desenvolvimento da arquitetura dual-mvc;

5.1. Sincoronização do Model

Atualmente copia-se todo o model de um local para o outro, do cliente ao servidor ou vice-versa. Um trabalho em andamento visa realizar a cópia somente das porções do model que sofreram modificação, otimizando assim a abordagem de forma significativa.

5.2. Pré-Buscas e atualizações do Model

Possibilitar aos clientes buscar diretamente os dados e atualizações, sem ter que passar obrigatória e primariamente pelo servidor, como ocorre hoje em dia.

6. Conclusão

Este artigo descreve como um design pattern MVC de particionamento independente pode ser usado no ambiente intrinsecamente dependente de particionamento de aplicações web. A utilização do modelo fwap possibilita que o código escrito para uma aplicação stand-alone seja utilizado sem modificações em várias implantações para várias plataformas cliente com qualquer esquema de particionamento.
Além disso, o modelo fwap possibilita o re-particionamento, após a conclusão dos trabalhos e em função de novas necessidades sem custo e/ou esforço adicional.