제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

Object Pool 패턴을 이용한 WIPI기반 MVC 모델의 개선

김철민*, 서성채, 유진호, 김병기
전남대학교 전산학과
{cmkim, scseo, jhyou, bgkim}@chonnam.ac.kr

Improvement of WIPI-based MVC Model using Object Pool Pattern

Chul-Min Kim*, Seong-Chae Seo, Jin-Ho You, Byung-Ki Kim
Dept. of Computer Science, Chonnam National University

요 약

무선 단말기와 같은 제한된 환경에서의 애플리케이션 개발은 대부분 사용자 인터페이스 부분이 차지하고 있다. 사용자 인터페이스의 효과적인 관리를 위한 기법이 연구되어 왔으며 기존의 MVC 모델이 사용되고 있다. MVC 모델은 애플리케이션 개발 시 사용자 인터페이스의 효과적인 관리를 위한 방법을 제공한다. 그러나 제한된 무선 단말기 상에서의 MVC 모델의 적용은 사용자 인터페이스 뷰의 독립성을 위한 과도한 객체 생성으로 작업 프로세스와 메모리 공간의 효율성을 떨어뜨리는 문제점을 가지고 있다.

본 논문은 기존 MVC 모델에 Object Pool 패턴을 활용하여 UI 컴포넌트의 재사용이 가능한 개선된 UP-MVC모델을 제안한다. UP-MVC 모델은 무선 단말에서 메모리 리소스 사용의 문제점을 개선하여 모바일 애플리케이션 성능을 높일 수 있다.

1. 서론

무선단말기 보급 증가와 WIPI (Wireless Internet Platform for Interoperability) 모바일 표준 플랫폼 규격의 등장으로 게임, 멀티미디어, 비즈니스 등에 적용되는 다양한 모바일 애플리케이션의 개발이 진행되고 있다[1]. 하지만 무선 단말기 환경에서 애플리케이션을 개발할 때에는 여러 가지 한계가 따른다. 작은 메모리와 CPU 속도, 제한된 전력 등이 주원인이며, 이러한 한계 때문에 서비스 개발의 대부분을 사용자 인터페이스 부분이 차지하고 있다[4]. 사용자 인터페이스 중심으로 모바일 애플리케이션 개발을 위한 기법이 요구되고 있으며, 기존 환경에서 적용되던 개발 기법들을 모바일 환경에서도 적용시키려는 연구가 진행되고 있다[2][3][5].

이러한 개발 기법 중 MVC 모델은 사용자 인터페이스에 대한 책임을 분담하는 방법으로서 유용한 구조 모델로 인식되고 있으며, 특히 엔터프라이즈 서버와 모바일 클라이언트가 연동하는 End-to-End 애플리케이션을 개발하는데 효과적인 모델로 활용되고 있다[4][6]. 그러나 MVC 모델은 제한적인 무선 단말기 상에서 적용하기에는 몇 가지 문제점을 가지고 있다[4]. 이에 본 논문에서는 모바일 환경에서 MVC 모델을 적용할 때 문제점을 제시하고, 문제점을 개선하는 모델로서 Object Pool 메커니즘을 활용한 개선된 MVC모델로서 UP-MVC 모델을 제안한다.

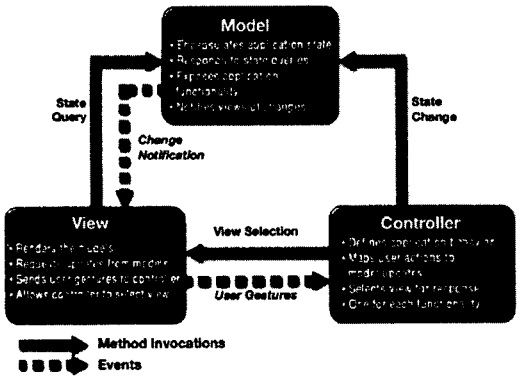
본 논문의 구성은 다음과 같다. 먼저 2장의 관련 연구에서 MVC 모델과 Object Pool 패턴을 다루고, 3장에서는 모바일 환경에서의 MVC 모델의 문제점을 제시한다. 4장에서는 문제점을 해결할 수 있는 모델로 UP-MVC 모델을 제시하고 적용하여 성능 평가를 수행한다. 마지막 5장에서는 결론을 내린다.

2. 관련연구

2. 1. MVC 모델

MVC(Model-View-Controller) 모델은 객체지향 사용자 인터페이스 분해를 위해 잘 알려진 방법으로 컴포넌트의 기반 설계 방식으로서 애플리케이션을 모델(Model), 뷰(View), 컨트롤러(Controller)로 나눈다. MVC 모델은 사용자와 상호 작용하는 소프트웨어를 설계하는 과정에서 다음과 같은 Model-View-Controller의 세 가지 요소 형태로 분리할 수 있다[2]. MVC 모델의 구조와 각 요소 간

의 관계는 [그림1]과 같다.



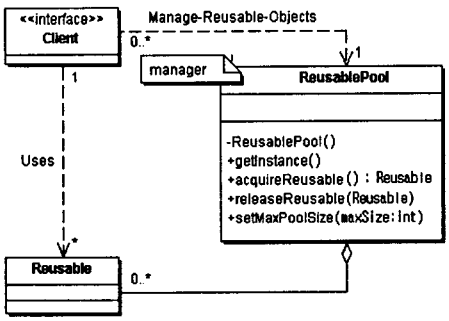
[그림1] MVC 모델 구성 및 관계

MVC 모델을 개발환경에 적용하는 경우 각각의 소프트웨어를 컴포넌트 형태로 개발할 수 있다. 따라서 빠른 속도로 변화하는 소프트웨어 개발 환경에서 유지보수와 확장성이 뛰어난 장점이 있다. 그러나 뷰의 독립성을 유지하기위해 수반되는 과도한 객체 생성과 UI 컴포넌트의 용도에 따른 부분적인 이용은 불필요한 자원의 낭비와 실행 속도의 저하를 가져오는 단점을 가지고 있다[4].

2. 2. Object Pool 패턴

객체 재사용 기법으로 흔히 객체 풀(Object Pool) 기법을 사용한다. 이는 임의의 객체를 미리 생성해 두고 벡터 클래스를 사용해서 관리하는 기법이다. 해당 객체를 사용하기 위해서는 풀의 벡터에 있는 객체를 가져오고 사용한 후에 객체 풀에 반납한다. 자주 사용하는 인스턴스 객체를 매번 사용되어질 때마다 생성하지 않고 미리 생성해 두었다가 사용한 후 다시 제거하지 않고 재사용 한다면, 클래스 인스턴스 생성에 소비되는 시간을 감소시킬 수 있으며 메모리 로딩 시간을 절약할 수 있다[7][8][9].

[그림2]은 객체 풀 패턴에서 클래스 역할 관계를 보여주는 UML 클래스 다이어그램이며 다음은 각 클래스의 역할을 설명하였다[7].



[그림2] Object Pool Pattern

- Reusable : 이 클래스 인스턴스는 제한된 시간동안 다른 객체와 서로 협력한다. 제한된 시간 후 협력이 끝나면 다시 ReusablePool에 반환된다.
- Client : 이 클래스 인스턴스는 Reusable 객체를 이용한다.
- ReusablePool : 이 클래스 인스턴스는 클라이언트 객체가 사용하는 Reusable 객체를 관리하는 역할을 담당한다.

3. 모바일 환경에서의 MVC 모델 적용에 따른 문제점

모바일 단말기 환경에서 작은 메모리와 CPU속도는 기존의 설계 패턴을 적용하는데 많은 문제점을 수반한다. 특히 MVC 모델을 적용했을 때에는 뷰의 독립성을 유지하기 위하여 다음 뷰를 확보하기 위해 많은 객체를 생성하게 되며, UI 컴포넌트의 용도에 따른 부분적인 이용이 이루어지게 된다. 결과 다음과 같은 문제점을 발생시킨다.

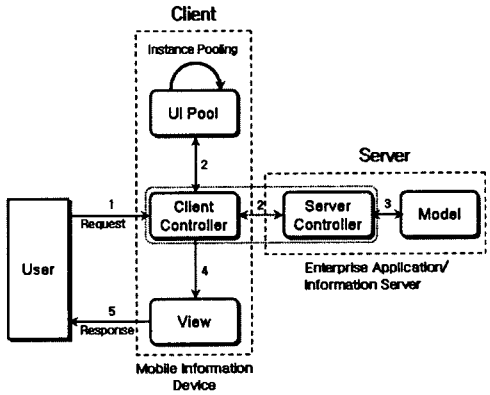
- UI 컴포넌트 인스턴스의 반복 생성으로 인한 동적 메모리 공간의 부족
- 빈번한 가비지 컬렉션 수행으로 인한 프로세스의 일시 중지 현상
- 메모리 로딩 시간의 증가

위에서 기술한 문제점은 시스템의 성능을 저하시키며 최종적으로는 사용자와의 상호작용을 방해하는 요인으로 작용한다.

본 논문에서는 이러한 문제점의 개선을 위한 구조 모델로 UP-MVC 모델을 제안한다.

4. UP-MVC 모델

본 논문에서 제안한 UP-MVC 모델은 기존 MVC 모델의 개선된 구조 모델이다. UP-MVC 모델은 엔터프라이즈 서버와 모바일 클라이언트가 연동하는 End-to-End 애플리케이션을 개발하는데 효과적인 구조 모델이다. UP-MVC 모델은 크게 Model, View, Controller 그리고 UI Pool의 네 개의 요소로 구성된다. 기존의 MVC 모델에 UI Pool을 배치하여 효과적으로 사용자 인터페이스를 관리하는 구조이다.

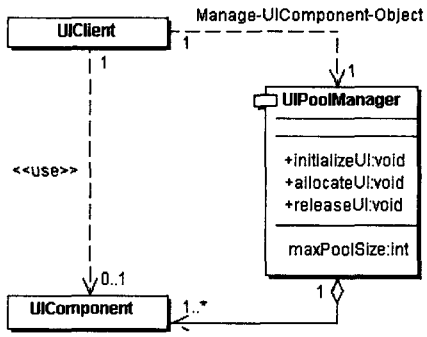


[그림3] UP-MVC 구조

UP-MVC 모델은 크게 모바일 디바이스에서 수행되는 클라이언트 부분과 서버 부분으로 나누어진다. 먼저 클라이언트 부분에는 View, Client Controller, UI Pool이 위치해 있으며, 서버에는 Server Controller와 Model이 위치해 있다. 각 요소의 역할은 아래와 같은 작업을 수행한다.

- View : 서버에 위치해 있는 모델의 정보를 컨트롤 흐름제어를 통한 적절한 결과화면의 표현을 담당한다. UI 컴포넌트와 그들 간의 결합된 형태이다.
- UI Pool : 뷰를 위해 자주 사용되는 UI 컴포넌트 요소를 매번 사용되어질 때마다 생성하지 않고 미리 생성해 두었다가 쓰고 다시 저장하여 재사용할 수 있는 메커니즘을 제공한다.
- Client Controller : 사용자의 입력 및 흐름제어를 담당하며 UI Pool과 Server Controller와의 협력을 통해 다양한 뷰를 제공한다.
- Server controller : Client Controller의 요청에 따른 비즈니스 영역의 상태 정보를 처리하여 보내준다.
- Model : 비즈니스 영역의 상태를 처리하는 로직을 담당하고 있다.

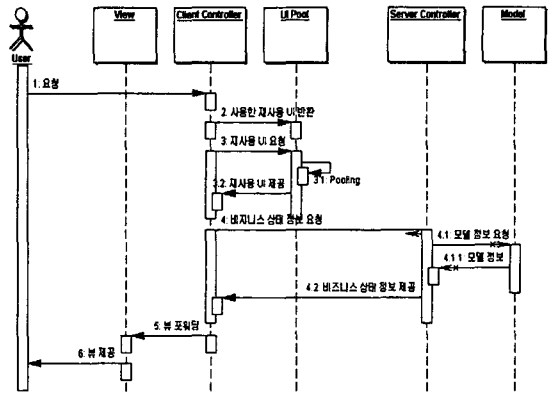
[그림4]는 UI Pool 클래스들의 역할을 보여주는 UML 클래스 다이어그램이며 다음은 각 클래스의 역할을 설명하였다.



[그림4] UI Pool 구조

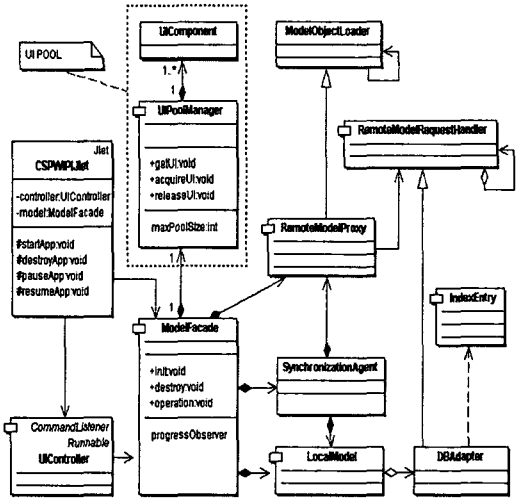
- UIClient : 재사용할 UIComponent 객체를 요청하는 컨트롤러 객체이다.
- UIPoolManager : UIComponent 객체를 담고 있는 풀이다. UIClient 객체가 UIComponent 객체를 요구시 풀 안에 대기하고 있는 UIComponent 중 해당되는 객체를 리턴해 준다.
- UIComponent : 재사용가능한 UI 컴포넌트로 지정된 객체로 풀 안에서 대기 상태에 있게 된다.

[그림5]는 UP-MVC의 각 요소 사이의 작업 처리 시나리오를 UML 시퀀스 다이어그램으로 보여주고 있다.



[그림5] UP-MVC 작업 처리 시나리오

[그림6]는 WIPI 기반에서의 UP-MVC를 적용한 자바 애플리케이션(Jlet)의 UML 클래스 다이어그램이다.



[그림6] UP-MVC 모델을 적용한 WIPI 애플리케이션 클래스 다이어그램

[그림6]에서 UIController, ModelFacade는 클라이언트 컨트롤러를 구체화시킨 클래스로서 서버와 클라이언트의 정보를 제어하여 다양한 뷰를 선택하고, 점선으로 강조된 UI Pool을 통하여 UI 컴포넌트를 재사용 함으로써 메모리 관리의 문제점을 개선할 수 있다.

따라서 UP-MVC 모델에서는 Client Controller에 의한 UI Pool의 선택적 사용과 이에 따른 UI 컴포넌트를 재사용 할 수 있으므로 불필요한 메모리의 낭비를 최소화하며 수행 속도의 향상을 가져올 수 있다.

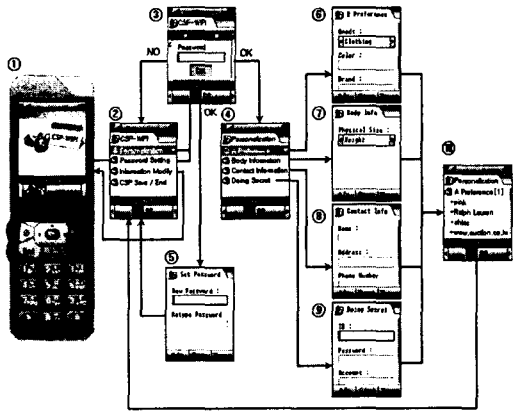
4. 3. WIPI기반에서의 UP-MVC 적용 및 성능평가

제안된 UP-MVC 모델을 통해 서버와 모바일 클라이언트가 연동하는 End-to-End 애플리케이션으로 WIPI기반 Client-Side Personalization(CSP-WIPI) 애플리케이션을

구현하였다. CSP-WIPI는 사용자가 핸드폰(cell phone)을 통해서 개인 정보를 보관하고 정보를 필요로 하는 서버에게 제공해주는 mCommerce 애플리케이션이라 할 수 있으며, WIPI 플랫폼 기반의 클라이언트 티어, J2EE 플랫폼 기반의 애플리케이션 서버 티어로 구성된다.

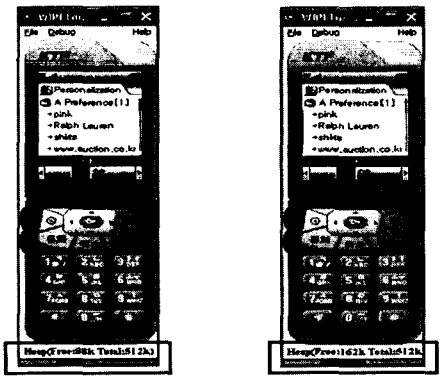
UI Pool은 사용자 인터페이스를 담당하고 있는 클라이언트 측에 배치되어 UI 컴포넌트의 재사용을 위한 메커니즘을 제공한다. 자주 사용되는 UI 컴포넌트로 Shell Component, DialogComponent, LabelCompoent, Progress Component을 생성하고 저장하여 해당 UI 컴포넌트를 필요로 하는 뷰에게 할당하고 회수한다.

[그림7]에서는 두가지 버전의 CSP-WIPI 애플리케이션이 동일한 화면 흐름과 기능을 수행하는 것을 보여주고 있다. UI Pool이 포함된 버전과 포함되지 않은 버전을 개발하여 각각의 메모리 상태를 측정함으로써 성능 평가를 실시하였다.



[그림7] CSP-WIPI 화면 흐름

[그림8]은 WIPI 에뮬레이터를 이용한 애플리케이션 수행에 따른 상태를 보여주는 결과 화면이다.



[그림8] MVC 적용과 UP-MVC적용 후 결과 화면

[그림7]에서와 같이 애플리케이션 수행 사이클 진행 결과를 에뮬레이터 화면으로 보여주고 있으며 메모리 사용 결과는 힙(Heap) 메모리 사용 표시 줄을 통해 확인할 수 있다. UP-MVC의 적용 애플리케이션의 메모리 여유 공간이 MVC의 적용 애플리케이션의 메모리 여유 공간보다 74Kbyte 정도의 메모리를 절약하였다. 따라서 UI 컴포넌트의 재사용을 통한 효율적인 메모리 관리가 이루어지고 있음을 확인할 수 있다.

5. 결론 및 향후 연구 과제

본 논문에서는 모바일 애플리케이션 개발에 있어서 구조 모델인 MVC 모델을 적용함으로써 발생하는 문제점을 제시하였다. 또한 MVC 모델의 장점인 유지보수와 확장성을 확보하고 UI 컴포넌트를 재사용할 수 있는 모델로서 UP-MVC 모델을 제시하였다. 제시된 모델은 Object Pool 메커니즘을 활용하여 모바일 단말기에서의 효과적인 리소스 관리가 이루어 질 수 있게 설계된 구조 모델로서 메모리의 효율성을 높이고, 불필요한 자원의 사용을 방지하는 효과를 제공하여 보다 효율적인 모바일 애플리케이션을 개발할 수 있게 한다.

향후 모바일 환경에서 Object Pool 메커니즘을 적용할 수 있는 구성 요소의 체계적인 분류와 연구가 필요하다.

참고문헌

[1] 이상윤, 김선자, 김홍남 "한국 무선 인터넷 표준 플랫폼(WIPI)의 표준화 현황 및 발전 전망", 정보과학회지 제22권 제1호, 2004

[2] Andrzej Dabkowski, Anna Maria Jankowska and Karl Kurbel, "Model-View-Controller Design Pattern for Mobile and Desktop-based Applications", 2003

[3] Jean Vanderdonckt, Murielle Florins, Frederic Oger, "Model-Based Design of Mobile User Interfaces", 2003

[4] Michael Juntao Yuan, "Enterprise J2ME: Developing Mobile Java Applications", Prentice Hall PTR, 2003

[5] Enrique Ortiz, Eric Giguere, C. Enrique Ortiz, "Mobile Information Device Profile for Java 2 Micro Edition", John Wiley & Sons, 2001

[6] Martyn Mallick, "Mobile and Wireless Design Essentials", 2003

[7] Wiebe de Jong, "Implement a JDBC Connection Pool via the Object Pool Pattern", http://www.developer.com/java/other/article.php/626291

[8] Mark Grand, "Patterns in Java Volume 1", John Wiley & Sons, 1998

[9] Bruce Powel Douglass, "Real-Time Design Patterns", Addison-Wesley, 2002