DocNo: 001.E.1.1

**Grape**

**Software Architecture**

**Document**

**Version 1.0**

**By**:

Group Undefined

2015-04

**Group Member**:

Hunter Lin

Birdy

Listen

Morning

Syachi

**Document Language**:

English

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2015.4.12 | 1.0 | Initialization of the report | Hunter Lin |
| Final Date | 2.0 |  |  |

Contents

**[1.](#_Toc417418435)****[Introduction](#_Toc417418435)** [5](#_Toc417418435)

[1.1. Purpose 5](#_Toc417418436)

[1.2. Scope 5](#_Toc417418437)

[1.3. Reference 5](#_Toc417418438)

**[2.](#_Toc417418439)****[Architectural Representation](#_Toc417418439)** [5](#_Toc417418439)

**[3.](#_Toc417418440)****[Architectural Goals and Constraints](#_Toc417418440)** [5](#_Toc417418440)

**[4.](#_Toc417418441)****[Use-Case View](#_Toc417418441)** [5](#_Toc417418441)

[4.1 Overview 5](#_Toc417418442)

[4.2 Architecturally Significant use cases 5](#_Toc417418443)

**[5.](#_Toc417418444)****[Logical View](#_Toc417418444)** [5](#_Toc417418444)

[5.1. Overview 5](#_Toc417418445)

[5.2. Front-end Interaction Mechanisms 6](#_Toc417418446)

[5.2.1. Front Controller 6](#_Toc417418447)

[5.2.2. Command Delegator 6](#_Toc417418448)

[5.2.3. Service Locator 6](#_Toc417418449)

[5.2.4. Security Handler 6](#_Toc417418450)

[5.3. Data Operation Mechanisms 6](#_Toc417418451)

[5.3.1. Persistency 6](#_Toc417418452)

[5.3.2. Session Facade 6](#_Toc417418453)

[5.4. Architecturally Significant Use Case Realization 6](#_Toc417418454)

[5.5. Architecturally Significant Model Elements 6](#_Toc417418455)

[5.6. Architecturally Significant Classes 7](#_Toc417418456)

**[6.](#_Toc417418457)****[Process View](#_Toc417418457)** [7](#_Toc417418457)

**[7.](#_Toc417418458)****[Deployment View](#_Toc417418458)** [7](#_Toc417418458)

**[8.](#_Toc417418459)****[Implementation View](#_Toc417418459)** [7](#_Toc417418459)

**[9.](#_Toc417418460)****[Size and Performance](#_Toc417418460)** [7](#_Toc417418460)

**[10.](#_Toc417418461)****[System Size](#_Toc417418461)** [7](#_Toc417418461)

**蓝色部分：Syachi**

**黄色部分：Hunter Lin**

**红色部分：Listen**

**紫色部分：Morning**

**绿色部分：Birdy**

1. **Introduction**

## 1.1. Purpose

## 1.2. Scope

## 1.3. Reference

1. **Architectural Representation**
2. **Architectural Goals and Constraints**
3. **Use-Case View**

## 4.1 Overview

## 4.2 Architecturally Significant use cases

1. **Logical View**

This section depicts firstly some important mechanisms in design model, most of which are generated by Design Patterns. Secondly, we describe the architecturally significant parts of the design model, such as the decomposition into subsystems and packages, and the logical structure of our system. We’ll start from the overview of the architecture, giving a direct and general view of the contents, then the presentation of the important structure, behavioral elements and other evaluations.

## 5.1. Overview

There are three dominant structures in the application design model:

1. Logical decomposition of the system into three layers.
2. The structure of the use case realizations derived from design patterns. Note that these mechanisms include some of the pre-defined solutions to facilitate our further implementations.

The high-level diagram of above is showed below:

You can see many mechanisms in the design model. Some of these mechanisms are derived from design patterns. In fact, the mechanisms we depict here can be of great use to any developer who intends to create a system with group operations. We use mechanisms to provide pre-designed solutions to some common problems that have to be addressed repeatedly in the application and to unify the designs of every part. That would significantly reduce our workloads.

In our grape system, two kinds of mechanisms exist:

1. Front-end Interaction with Other Components:
   1. Front Controller
   2. Command Delegator
   3. Service Locator
   4. Security Handler
2. Data Access and Operation
   1. Persistency
   2. Session Façade

These two kinds of mechanisms will be addressed below. In the following two sections, we will organize each mechanism in a strict and clear order. First, a class diagram and a sequence diagram will be displayed. Then, we will introduce how the mechanism works and the situation we apply it to our Grape system. Finally, we will address the reason why we choose this mechanism, and the advantages of using this mechanism.

## 5.2. Front-end Interaction Mechanisms

### 5.2.1. Front Controller

Class diagram and sequence diagram:

### 5.2.2. Command Delegator

### 5.2.3. Service Locator

### 5.2.4. Security Handler

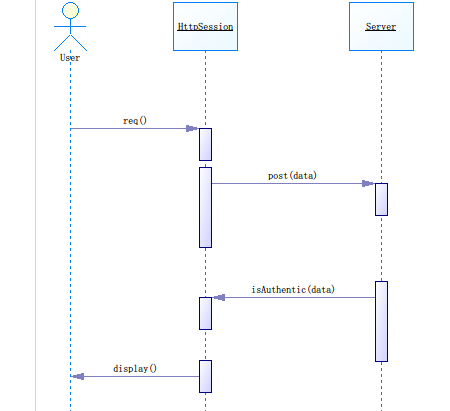
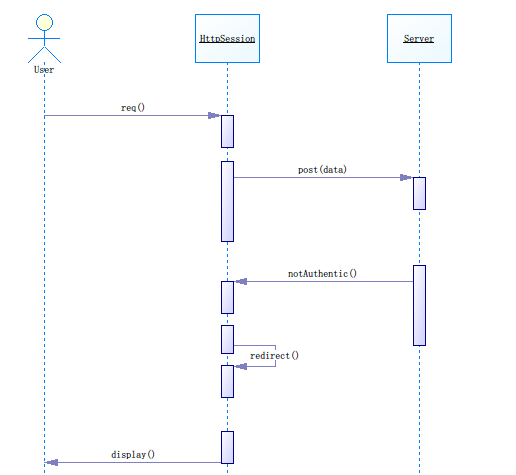
**How it works:**

When the user wants to use the system,the system will check whether he has logged in.If not,it will redirect to the page for logging in.And if the user has logged in,the system will identify his role(admin or normal user).Then corresponding function will be displayed.

**Why we use it:**

To avoid the users access the resources, which they have no authority to access.

### sequence diagram:



## 5.3. Data Operation Mechanisms

### 5.3.1. Persistency

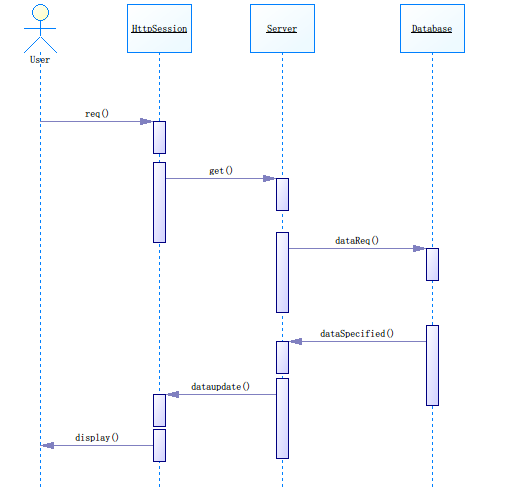
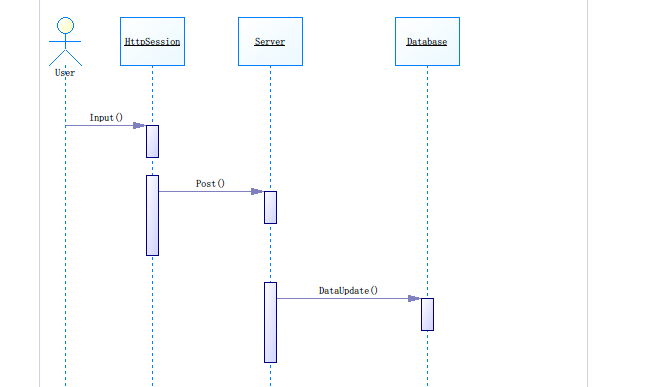
**How it works:**

In our system,we use mysqldb for python to connect MySQL database.One of the convenience is that you we can use the same SQL instruction in python by mysqldb.And it’s therefore simple to operate dynamic change of all kinds of data .

**Why we use it:**

It’s essential to keep the data permanently for further use.Obviously it’s a basic function of all websites.

### sequence diagram:



### 5.3.2. Session Facade

## 5.4. Architecturally Significant Use Case Realization

## 5.5. Architecturally Significant Model Elements

## 5.6. Architecturally Significant Classes

1. **Process View**
2. **Deployment View**
3. **Implementation View**
4. **Size and Performance**
5. **System Size**