Chapter 5

SCALABILITY AND AVAILABILITY
### Chapter 5 - Scalability and Availability

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Name of the Sub-Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>The importance of Scalability and Availability</td>
<td>79</td>
</tr>
<tr>
<td>5.2</td>
<td>Design Patterns for Scalability and Availability</td>
<td>81</td>
</tr>
<tr>
<td>5.3</td>
<td>The approach for Scalability and Availability</td>
<td>82</td>
</tr>
<tr>
<td>5.4</td>
<td>Refactoring Algorithm for scalability and availability</td>
<td>87</td>
</tr>
<tr>
<td>5.5</td>
<td>Case Studies</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>5.5.1 Hospital Management System</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>5.5.2 Library Management System</td>
<td>92</td>
</tr>
<tr>
<td>5.6</td>
<td>Summary</td>
<td>95</td>
</tr>
</tbody>
</table>
CHAPTER 5

Scalability and Availability

This chapter presents the research study made on reliability of web applications in terms of scalability and availability. Towards this end the architectural pattern XWADF introduced in chapter 3 is enhanced by incorporating the design patterns identified for promoting scalability and availability of web applications into the architectural pattern.

Then the enhanced architectural pattern is applied to the two case study applications HMS and LMS. The scalability is measured in terms of access time and throughput as these attributes improve scalability. Scalability and availability are the reliability attributes that enhance the popularity of web applications. The architectural pattern is evaluated using scalability metrics.

The results revealed that the reliability of the web applications has improved significantly when XWADF is adapted.

5.1 The importance of Scalability and Availability

In simple words, scalability is the ability to grow, the ability to serve increased number of requests or clients. In the context of web application design, it is essential to keep reliability [151] in mind. In this chapter we focus on two reliability attributes such as scalability
and availability. These two are highly desirable features that improve the quality and performance of web applications.

According to Sanderson [152] a web application is scalable when each user gets same quality irrespective of the number of users concurrently accessing the application. Poor usability [153] and poor scalability results in web applications when designers are not aware of web design patterns [154]. Web frameworks were also written without considering scalability usability and simplicity [155].

Enterprise web applications that generate content dynamically and data grids throw scalability challenges [156]. There are many good reasons to use design patterns as they can get rid of reinventing the wheel by promoting reuse. Moreover they are proven, and expressive. Model View Controller (MVC) [157] is one of the well-known architecture which renders many advantages to web applications including maintainability [158], availability and scalability [159].

The modern web applications use three tier or n-tier architecture. The generic model is [160] illustrated in Figure 5.1.

![Figure 5.1: Architecture of modern web sites](image-url)
As can be seen in Figure 5.1, the architecture reflects the complete flow including web resources running in web server and client, application and database servers. The entire flow together actually forms a functional web applications. However, in this chapter, our focus is to promote scalability and availability through best practices or design patterns that are the underlying proven blueprints within our architectural pattern XWADF.

5.2. Design Patterns for scalability and availability

Flyweight

When there is almost similar nature of creating high number of objects, Flyweight is used. When there is high number of objects then it consumes high memory.

So it requires flyweight design pattern which reduces the load on memory by sharing objects. It is achieved by using two types of properties intrinsic and extrinsic.

Mediator

Mediator design pattern is a behavioral design pattern which is widely used. Mediator introduces a layer in between two objects so that the interaction between that two objects taken place. The intermediate mediator object communicates between the two objects, and also helps in interaction for a set of different objects.

Factory Method

One of the creational patterns is a factory method. It is used to instantiate an object from a set of classes. The factory method will
have the super class, so that you can program for the interface and not for the implementation.

**Singleton**

There are only two points in the definition of a singleton design pattern,

1. Only one instance of a class is allowed.
2. That single instance accessed globally.

**Proposed Design Patterns**

So the design patterns identified to enhance web application’s scalability and availability include Flyweight, Mediator, Factory, and Singleton. Influenced by the functionality of these patterns we proposed the two design patterns that contain the required functionality to improve reliability of web applications.

The patterns we proposed include Object State Pattern and Interaction Pattern which will provide seamless interaction between objects involved in the request processing of a web application.

**5.3. The Approach of Scalability and Availability**

Design patterns which are already part of the XWADF contribute to performance of web application. When web application performance improves, it adds to the scalability of the application at design level. It is to be noticed that our research is not into server side measures to make web applications scalable.

However, we throw light on the design of web applications for improving performance using an architectural pattern with many
underlying design patterns. Our previous chapter 3 was particular about improving Response Time and throughput of web applications for improving performance of web applications. However, the Response Time and throughput attributes can be used to measure scalability of web applications.

Hence we have scalability in terms of response time and similarly scalability in terms of throughput. There is a small variation when comparing response time and throughput with scalability in terms of response time and throughput.

We are embedding the proposed design patterns into our architectural pattern by enhancing XWADF. The improved architectural pattern is as shown in Figure 5.2.

Figure 5.2: The approach of Scalability and Availability
As can be seen in Figure 5.2, it is evident that more design patterns are included into the architecture so as to make the application highly scalable with high availability. Interestingly these patterns can be used in either model layer or controller layer based on the complexity that arises in those layers in case of complex enterprise web applications.

**The different objects in scalability and availability**

The different small objects are created using creational patterns known as factory method and singleton. From these two patterns Object State Pattern is proposed. Using singleton different objects are accessed from only one object.

The different objects shared memory efficiently using flyweight pattern. Similarly Mediator helps interaction between set of objects. From these two patterns Interaction Pattern is proposed. The following figure 5.3 shows representation of these objects.

![Diagram](image)

Figure 5.3 Different objects in Scalability and Availability
From the above figure 5.3 different objects in scalability and availability are shown. Here all objects are created. All objects share memory. These objects are fine grained objects so interaction between these objects is controlled easily using mediator pattern.

Here Scalability is taken in terms of response time and throughput. Scalability in terms of response time $S(Rt)$ is a measure that tells how long user waits to get response to a query.

That is $S(Rt) = \frac{Wt}{res}$ where $Wt$ is waiting time, and $res$ is a response.

Scalability in terms of throughput is the amount of work accomplished in a certain amount of time period.

That is $S(Tp) = \frac{W}{T}$ where $W$ is the amount of work and $T$ is a Time Period.

The simplest representation for availability is as a ratio of the expected value of the uptime of a system to the aggregate of the expected values of up and down time.

Let $N[uptime]$ $\rightarrow$ $D[uptime]$ $\rightarrow$ expected uptime value

$V[downtime] \rightarrow$ downtime value then

$$\text{Availability (A)} = \frac{N[uptime]}{D[uptime] + V[downtime]}$$

Downtime (or) outage duration: A period of time that a system fails to provide or perform its primary function.
If we define status function $X(t) = 1$ System function at time $t$ else 0 otherwise.

Therefore the availability $A(t)$ at time $t>0$ is represented by

$$A(t) = E[X(t)]$$

The different notations in the above diagram is given by

- Model $\rightarrow$ M
- Controller $\rightarrow$ C
- View $\rightarrow$ V
- Cp $\rightarrow$ Connection Pool
- P $\rightarrow$ POJO
- Da $\rightarrow$ DAO Database Access Object
- OSP $\rightarrow$ Object State Pattern
- IP $\rightarrow$ Interaction Pattern
- S $\rightarrow$ SQL Mapper
- Ca $\rightarrow$ Cache
- Dp $\rightarrow$ Delegation Pattern
- Dt $\rightarrow$ DTO Data Transfer Object (or Value Object)
- De $\rightarrow$ Decorator
- Req $\rightarrow$ Request
- Res $\rightarrow$ Response

The processing of the above XWADF architectural pattern approach is as follows.

1. $M \leftarrow Cp + P + Ca$  [First Cp, P, Ca carried in Model M]
2. $M \leftarrow S + Da + Dt$  [Next S, Da, Dt carried in Model]
3. M $\leftarrow$ OSP + IP  \[ Next OSP, IP carried in Model M\]
4. C $\leftarrow$ Dp + Dt  \[ First Dp and Dt carried in Controller C\]
5. C $\leftarrow$ OSP + IP  \[ Next OSP and IP carried in Controller C\]
6. V $\leftarrow$ De  \[Finally response is generated in View V\]

5.4 Refactoring Algorithm for Scalability and Availability

Existing web applications that do not rectify for Scalability and Availability well can be refactored to be enhanced XWADF. This algorithm guides developers to refactor successfully. This algorithm can be a basis for developing an automatic conversion tool for reliability in future. Figure 5.4 shows the proposed refactoring algorithm for scalability and availability.

**Algorithm:** Refactoring Algorithm for Scalability and Availability

<table>
<thead>
<tr>
<th>Inputs:</th>
<th>Existing Web Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outputs:</td>
<td>Enhanced XWADF for Scalability and Availability</td>
</tr>
<tr>
<td>Assumptions:</td>
<td>Java Web Application with Servlets and JSPs as Web Resources</td>
</tr>
</tbody>
</table>

**Process:**

1. START

2. If Web Application is in MVC Pattern Then
   a. Configure connection pool in web server/application server
   b. Implement a design pattern that gets connection from pool
c. Define POJO for every relational table
d. Move the Database interaction logic to DAO’s
e. Implement Object State Pattern
f. Implement Interaction Pattern
g. Implement SQL Mapper design pattern
   and use it in DAOs
h. Implement Design Pattern for Caching
i. Apply caching in Model layer
j. Use Delegation Pattern and DTO in controller
k. Identify responses to decorator in the view
l. Use decorator pattern to decorate responses

3. If Web Application is not in MVC Pattern Then
   a. Move Presentation logic to view layer
   b. Move business logic layer to model layer
   c. Go back to step 1 to refactor as specified.

4. STOP

Figure 5.4: Refactoring algorithm for Scalability and Availability

The refactoring algorithm guides developers to refactor their existing web applications to be compliant with our enhanced architectural framework XWADF. We have taken two case studies to demonstrate the effectiveness of the proposed enhanced architectural framework with suggested design patterns.
The above algorithm is presented in the form of flowchart for easy understanding.
Figure 5.5: Flow chart of Refactoring algorithm for Scalability and Availability
5.5 Case Studies

To evaluate the Quality of web applications in terms of Scalability and Availability two case studies are used. Scalability is also measured in terms of response time and throughput.

5.5.1 Hospital Management System

The first case study considered is Hospital Management System. This Hospital Management System allows users to download the prescriptions. There may be many users who download the prescriptions. When there is a single user then the time taken to download that prescription is less. When number of users increases then the time taken to download that prescriptions will be more.

Here Scalability is evaluated in terms of response time and throughput. Hence when a single user is downloading a prescription, the time taken will be less. But when the number of users increases automatically the downloading time increases.

This downloading time depends on number of entities (records) in the database. When number of entities increases the downloading time also increases.

![Figure 5.6: Users accessing data entities from hospital database](image)
From the figure 5.6 the entities taken in the hospital database are 50000, 75000, and 100000. By using web container (MVC) & by using enhanced XWADF the scalability in terms of response time and throughput values are determined by different users for different entities and related graphs are drawn. This scalability in terms of response time and throughput values and resultant graphs are shown in chapter 6. All resultant graphs are compared.

The number of users considered is 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50. The software tool used for this purpose is LoadUIWeb 2. Here the scalability in terms of throughput is evaluated by the amount of work accomplished in one second time period per user.

The availability measure of web application for Hospital Management System is evaluated and 50000 and 100000 database entities are taken. For both the entities the availability measure is evaluated by using with and without design patterns. The resultant graphs are drawn in chapter 6 and compared the results.

5.5.2 Library Management System

The second case study considered is Library Management System. This Library Management System allows users to download the different types of books. There may be many users who download the different types of books. When there is only a single user then the time taken to download the books is less. When number of users increases, then the time taken to download those books will be more.
Here also scalability is determined in terms of response time and throughput. Hence when a single user is downloading response time will be more. But when the number of users increases, automatically it increases the response time.

This downloading time depends on number of entities (records) in the database. When number of entities increases the downloading time also increases.

![Diagram showing users accessing data entities from library database](#)

**Figure 5.7 : Users accessing data entities from library database**

From the figure 5.7 the entities taken in the library database are 10000, 25000, and 50000. By using web container (MVC) & by using enhanced XWADF the scalability in terms of response time and throughput values are determined by different users for different entities and related to that graphs are drawn. This scalability in terms of response time and throughput values and resultant graphs are shown in chapter 6. All resultants graphs are compared.

The number of users considered is also 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50. The software tool used for this purpose is LoadUIWeb 2. Here the scalability in terms of throughput is
determined when the amount of work accomplished in 1 sec time period.

The availability measure of web application for Hospital Management System is evaluated. Here the database entities are taken 25000 and 50000. For both the entities the availability measure is determined by using with and without design patterns. The resultant graphs are drawn in chapter 6 and compared the results.
5.6 Summary

In this chapter we used scalability and availability of web applications. Scalability means when number of users increases it should be scalable. For this we used response time and throughput. When the number of users increases, the scalability in terms of response time will be in small variation so that it should be improved scalability.

Here we used enhanced XWADF that used two design patterns Object State Patterns and Interaction Patterns. These design patterns creates small objects and makes more interaction between Model, Controller and database so that the response time will less when compared with using without enhanced architectural pattern.

The proposed enhanced XWADF is applied successfully to two case studies HMS and LMS and results are found. These results shows that the proposed approach is more useful for improving scalability in terms of response time and throughput. Similarly we used availability. We found availability with and without using the design patterns.

When we use these design patterns, the availability is more compared with without using these design patterns. Similarly the proposed enhanced XWADF is applied to two case studies HMS and LMS and found that the proposed enhanced approach is better for Availability.