Chapter - 4

FAULT TOLERANT
WEB APPLICATIONS
CHAPTER 4

Chapter - 4. Fault Tolerant Web Applications

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Name of the Sub-Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>The importance of Fault Tolerant Web Applications</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Fault Tolerance Design Patterns</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Fault Tolerant Web Applications approach</td>
<td>68</td>
</tr>
<tr>
<td>4.4</td>
<td>Fault Tolerant Refactoring Algorithm</td>
<td>71</td>
</tr>
<tr>
<td>4.5</td>
<td>Case Studies</td>
<td>74</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Hospital Management System</td>
<td>74</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Library Management System</td>
<td>75</td>
</tr>
<tr>
<td>4.6</td>
<td>Summary</td>
<td>77</td>
</tr>
</tbody>
</table>
CHAPTER 4

Fault Tolerant Web Applications

This chapter throws light on improving fault tolerance of web applications. An architecture based approach is used to achieve it. Strategies or methods that can withstand faults in web applications at application level are identified. These strategies are incorporated into some design patterns to make web applications fault tolerant. The architectural pattern proposed in chapter 3 is enhanced to incorporate the design patterns as underlying structures in the architectural pattern.

Thus our architectural pattern has become useful tool to build highly robust web applications that exhibit desirable quality attributes such as fault tolerance. The enhanced architecture is applied to two case study web applications (HMS and LMS) built using Java Enterprise Edition. The fault tolerance feature of these web applications is evaluated with standard metrics.

We built two Fault Tolerant (FT) design patterns and incorporated into XWADF so as to make design level decisions that help in developing fault tolerant web applications. We implemented two FT design patterns namely Fault Tolerant Exception Handling Pattern (FTEHP) and Fault Tolerant Security Pattern (FTSP) using
Aspect Oriented Programming (AOP) which separates cross cutting concerns from pure business logic thus getting rid of code pollution.

The first Design Pattern takes care of faults pertaining to common runtime errors (Exceptions) while the second Design Pattern takes care of security related faults. The FTSP is implemented on top of SSL for fool proof security. We applied the enhanced XWADF to two existing web applications namely HMS and LMS. The empirical results revealed that our architecture based approach to build fault tolerant applications cloud make applications fault tolerant.

4.1 The importance of Fault Tolerant Web Applications

Web applications are widely used by businesses to reach global audience. E-commerce and other commercial applications that leverage business performance need to be given highest priority since their performance leads success. Having said this it is essential to make such applications fault tolerant. Faults might occur for many reasons such as software, hardware, network and human errors.

Building a fail-safe web application is challenging task. Apart from measures taken at server level with respect to fault tolerant behavior of web server the application level fault tolerant structure or strategies play an important role to achieve fail-safe standard. Fault tolerance is also known as graceful degradation which helps the application to achieve operational continuity.

It can recover faults from unknown or unexpected faults. Rather than failing, the fault tolerant web application continues working,
most probably, in less than ideal fashion until the fault is overcome. At many levels faults may occur. In this chapter we focus on the faults at application level and let the application withstand potential faults. Troger [145] described the phases of fault tolerance as shown in Figure 4.1.

![Figure 4.1: Phases in Fault Tolerance](image)

As can be seen in Figure 4.1, it is evident that fault tolerance has phases such as fault activation, error detection, error process (error recovery and error mitigation) and then fault treatment. These phases are self-explanatory and intuitive in nature. The architectural pattern XWADF that we proposed and improved in [146], [147] to improve performance of web applications with features such as availability and scalability is further extended in this chapter in order to make the architecture fault tolerant.
4.2. Fault Tolerant Design Patterns

Our architectural pattern presented in the preceding section has been improved to incorporate FT design patterns. Interestingly fault tolerance and security come under non-functional requirements. For clean coding strategy we adapt AOP paradigm to implement the FT design patterns. Our FT design patterns cover both runtime abnormal situations and also security attacks.

Therefore we implement two FT patterns namely Fault Tolerant Exception Handling Pattern (FTEHP) and Fault Tolerant Security Pattern (FTSP). Before describing the implementation of these FT patterns I would like to introduce AOP.

Aspect Oriented Programming (AOP)

This is a programming approach which is based on Object Oriented Programming (OOP). It complements OOP by allowing separation of concerns. To state differently, it helps Java developers to separate business logic from other concerns or aspects. Especially non-functional requirements can be implemented using AOP. Thus separation of functional and non-functional requirements is possible.

Since FT and security are non-functional requirements, they are implemented using AOP. Thus the application BL code is kept clean as it does not pollute with cross cutting concerns like security and FT. The Figure 4.2 shows without using AOP and Figure 4.3 shows with using AOP in Java programming.
As can be seen in Figure 4.2 the coding of non-functional requirements such as logging, security, and profiling are mixed in an object along with BL. This is polluting the code. In Figure 4.3 the separation of such concerns from pure BL is shown. At runtime the cross-cutting aspects are mixed with pure BL without polluting the source code. This weaving is achieved using a GOF pattern [148] known as Decorator.
Fault Tolerant Exception Handling Pattern (FTEHP)

This pattern is implemented using Java language in AOP paradigm. The AOP framework used is AspectJ [149] which is a full-fledged AOP framework available. We defined an aspect by name FTExceAspect which performs activities such as fault detection and isolation, fault recovery and fault treatment. It also takes necessary steps to prevent fault propagation. Instead of causing the system fail, this pattern will give priority to operational continuity.

Fault Tolerant Security Pattern (FTSP)

This pattern is responsible to handle security faults. It takes care of general security mechanisms [150] such as authentication, authorization, confidentiality, and non-repudiation. The functionality is implemented using an aspect developed in Aspect J. The name of aspect we implemented is FTSecAspect. The underlying protocol used in this pattern is Secure Sockets Layer (SSL). The SSL is configured by modifying server.xml found in Tomcat’s configurations directory.

4.3. Fault Tolerant Web Applications Approach

Our FT patterns namely Fault Tolerant Exception Handling Pattern (FTEHP) and Fault Tolerant Security Pattern (FTSP) are incorporated into XWADF so as to make the architecture with Fault Tolerant Web Applications approach. The resultant architecture is as shown in Figure 4.4.
Building fault tolerant features into the architectural pattern of web application is very important so as to enable developers to follow the best practices. Thus the enhanced XWADF architecture when followed yields fault tolerant web applications. The two patterns for Fault Tolerance namely FTEHP and FTSP are specialized reusable components that can be reused in every web application instead of reinventing the wheel. Thus our architectural approach ensures highly fault tolerant web applications. The FTEHP and FTSP is as follows

FTEHP  \(\rightarrow\) Aspect J Frame work  \(\rightarrow\) FTExceAspect

Where FTExceAspect  \(\rightarrow\) Fault Detection + Fault Recovery

FTSP  \(\rightarrow\) Aspect J Frame work  \(\rightarrow\) FTSecAspect

Where FTSecAspect  \(\rightarrow\) Security
The different notations in the above diagram is given by:

- **Model** → M
- **Controller** → C
- **View** → V
- **Cp** → Connection Pool
- **P** → POJO
- **Da** → DAO Database Access Object
- **FTEHP** → Fault Tolerant Exception Handling Pattern
- **FTSP** → Fault Tolerant Security Pattern
- **S** → SQL Mapper
- **Ca** → Cache
- **Dp** → Delegation Pattern
- **Dt** → DTO Data Transfer Object (or Value Object)
- **De** → Decorator
- **Req** → Request
- **Res** → Response

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

1. **M** ← Cp + P + Ca [First Cp, P, Ca carried in Model M]
2. **M** ← S + Da + Dt [Next S, Da, Dt carried in Model M]
3. **M** ← FTEHP + FTSP [Next FTEHP and FTSP carried in M]
4. **C** ← Dp + Dt [First Dp and Dt carried in Controller C]
5. **C** ← FTEHP + FTSP [Next FTEHP and FTSP carried in C]
6. **V** ← De [Finally response is generated in View V]
4.4 Fault Tolerant Refactoring Algorithm

Existing web applications that do not rectify for faults 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 fault tolerance in future.

//Algorithm: Fault Tolerant Refactoring Algorithm

<table>
<thead>
<tr>
<th>Inputs: Existing Web Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outputs: Enhanced XWADF for Fault Tolerance</td>
</tr>
<tr>
<td>Assumptions: Java Web Application with Servlets and JSP s as Web Resources</td>
</tr>
<tr>
<td>Process:</td>
</tr>
<tr>
<td>1. START</td>
</tr>
<tr>
<td>2. If Web Application is in MVC Pattern Then</td>
</tr>
<tr>
<td>a. Configure connection pool in web server/application server</td>
</tr>
<tr>
<td>b. Implement a design pattern that gets connection from pool</td>
</tr>
<tr>
<td>c. Define POJO for every relational table</td>
</tr>
<tr>
<td>d. Move the Database interaction logic to DAO's</td>
</tr>
<tr>
<td>e. Implement a design pattern FTEHP for exception handling</td>
</tr>
<tr>
<td>f. Implement a design pattern FTSP for Security</td>
</tr>
<tr>
<td>g. Implement SQL Mapper design pattern and use</td>
</tr>
</tbody>
</table>
The Fault Tolerant 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 flow chart for easy understanding.
Start

Is Web App is in MVC Pattern

yes

Configure connection pool

Implement Connection from pool

Define POJO

Configure connection pool

Move Database to DAO

Implement FTEHP

Implement FTSP

Implement SQL Mapper

Implement Design Pattern for Caching

Apply Caching in Model Layer

no

Move Presentation Logic to View Layer

Move Business Logic to Model Layer
4.5 Case Studies

To evaluate the quality of web applications in terms of fault tolerance two case studies are used.

4.5.1 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 only 1 user, the number of faults will be less. When number of users increases then the number of faults will be more because there may be security fail or no exception handling.

So here quality is evaluated in terms of fault tolerance. Hence when one user is downloading it will be downloaded fast because the faults are less. But when number of users increases downloading time
increases and there may be occupying faults. This downloading time depends on number of entities (records) in the database. When number of entities increases, the downloading time also increases.

![Diagram](image)

**Figure 4.7: Users accessing entities from hospital database**

In the above figure 4.7 the entities taken in the database are 50000, and 100000. For each entities number of faults are determined and related graphs are drawn. The faults that occur with and without design patterns are determined. The resultant graphs are drawn in chapter 6 and that graphs are compared. The number of users considered is 10 users.

### 4.5.2 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 kinds of books. When there is only 1 user, then the number of faults will be less. When number of users increases then the number of faults also increases because there may be possibility of runtime errors.
The quality is evaluated in terms of fault tolerance. Hence when single user is downloading, faults will be less and hence the downloading time is also less. But when number of users increases, automatically it increases the faults. This downloading time depends on number of entities in the database.

![Diagram](image)

**Figure 4.8:** Users accessing entities from library database

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

From the figure 4.8 the entities taken in the database are 25000, and 50000. For each entity, the faults are determined and related graphs are drawn. The fault tolerance with and without design patterns are determined. The resultant graphs are drawn in chapter 6 and that graphs are compared. The number of users considered is 10 users.
4.6 Summary

We need quality of web applications design for downloading the different items from web sites. While downloading faults may occur, which may affect the downloading. So in this chapter we proposed two design patterns for fault tolerance and the proposed XWADF architectural pattern is enhanced by using these design patterns.

The two design patterns proposed here are Fault Tolerant Exception Handling Pattern (FTEHP) and Fault Tolerant Security Pattern (FTSP). The first Design Pattern takes care of faults pertaining to common runtime errors while the second Design Pattern takes care of security related faults.

In addition to this “Fault Tolerant Refactoring Algorithm” is proposed. This algorithm helps developers to upgrade their enterprise web applications in conformity with the proposed enhanced architectural pattern XWADF for quality gains in terms of Fault Tolerance.

This proposed enhanced XWADF architectural pattern is applied successfully to two case studies known as HMS and LMS and evaluate the results. These results show that the number of faults is less using enhanced XWADF when compared with without using these architectural patterns.