05/18/25

Ian Lewis

CSD 380

Module 12 Assignment

Building web applications for big industries like banking or healthcare isn’t just about writing good code, it’s about making sure that software meets strict rules and runs reliably in the real world. Chapter 23 of *Beginning Jakarta EE Web Development* by Luciano Manelli and Giulio Zambon offers two insightful case studies that show how Jakarta EE technology helps developers tackle those real-life challenges. The first case study focuses on compliance in regulated industries, while the second looks at how ATM systems use telemetry to stay reliable. Both examples offer practical lessons about how Jakarta EE can help teams build trustworthy, enterprise-level software.

In the first case study, “Providing Compliance in Regulated Environments,” the authors talk about how industries like finance or healthcare have to follow strict rules around data privacy, security, and auditing. These aren't optional as systems have to prove that they follow the rules, often through audits or certification. Because of this, the authors stress that developers should think about compliance from the very beginning of a project. Trying to add it in later can lead to major delays and messy code. Jakarta EE’s built-in features like filters, interceptors, and security tools make it easier to log activity, protect data, and prove that systems are secure. One key takeaway here is that it’s not just about writing the application, it’s about writing it in a way that keeps regulators happy and users safe. Documentation and automated logging are just as important as the application’s features (Manelli & Zambon, 2020).

The second case study, “Relying on Production Telemetry for ATM Systems,” is all about keeping things running smoothly in systems where even a few minutes of downtime can cause real problems. ATMs need to be online and working 24/7. To make that possible, developers use telemetry—real-time data that shows how the system is performing. The authors explain how Jakarta EE tools like message-driven beans (MDBs) allow the application to handle huge amounts of telemetry without slowing things down. For example, if one ATM starts acting up, the system can catch it early and alert someone before customers are affected. Over time, analyzing this data can even predict problems before they happen. The big lesson here is that smart use of data, combined with Jakarta EE's ability to process things asynchronously, leads to systems that are more reliable and easier to maintain (Manelli & Zambon, 2020).

Even though the two case studies deal with different types of challenges (compliance in one, system health in the other), they both show the importance of thinking beyond just the app's core features. Whether it's following legal rules or keeping machines running without interruption, developers need to plan for real-world needs from day one. Jakarta EE gives them the tools to do that in a clean, manageable way. Features like dependency injection, built-in security, and support for background processes make it easier to meet these demands without writing everything from scratch.

In short, these case studies show that Jakarta EE isn't just about building apps. It's about building the kind of apps that can survive in complex, high-pressure environments. When developers use the platform thoughtfully, they can create systems that are not only functional, but also safe, stable, and ready for anything.

References:

Manelli, L., & Zambon, G. (2020). *Beginning Jakarta EE Web Development: Using JSP, JSF, MySQL, and Apache Tomcat for Building Java Web Applications* (3rd ed.). Apress.