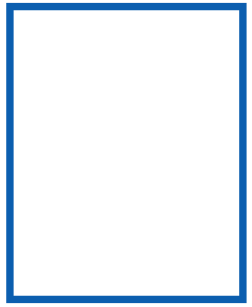
Java EE 7 Fundamentals

Introduction

Antonio Goncalves

@agoncal | www.antoniogoncalves.org

Course Outline



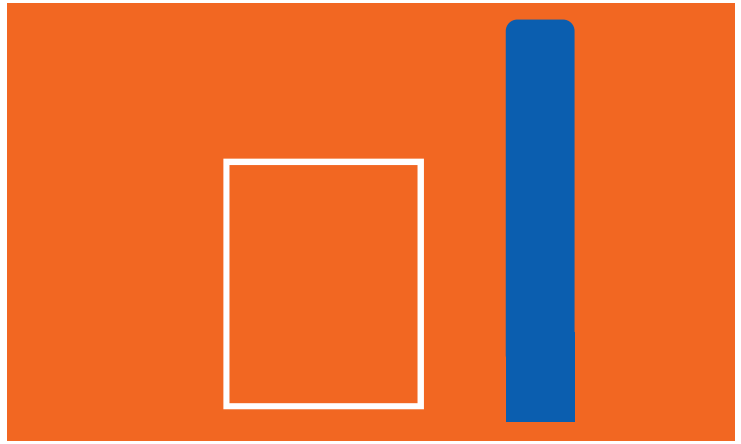
- Introduction
- Understanding Java EE
- Creating a common tier
- Addressing business concerns
- Implementing Web application
- Interoperating with external services
- Architectural summary

Audience

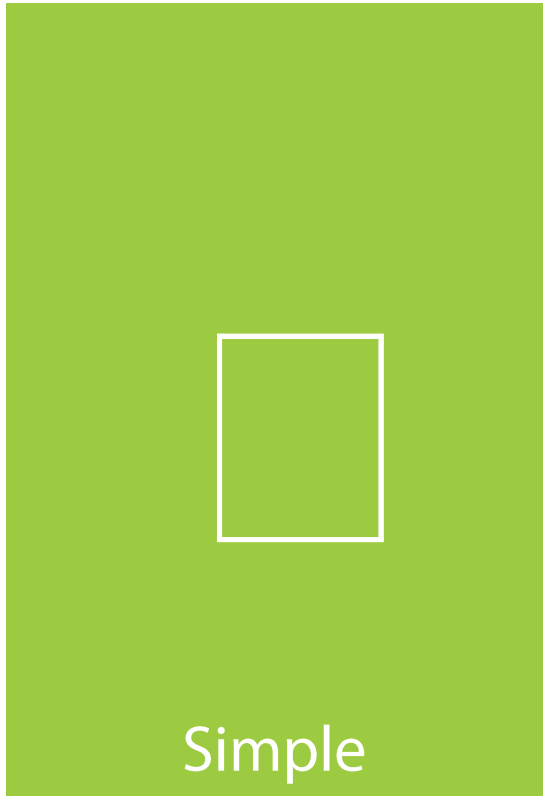
Technical

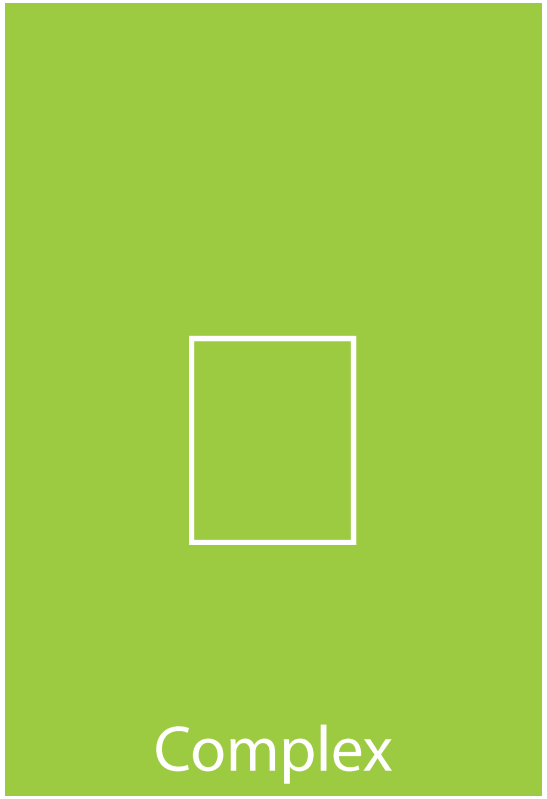
Technical Architect

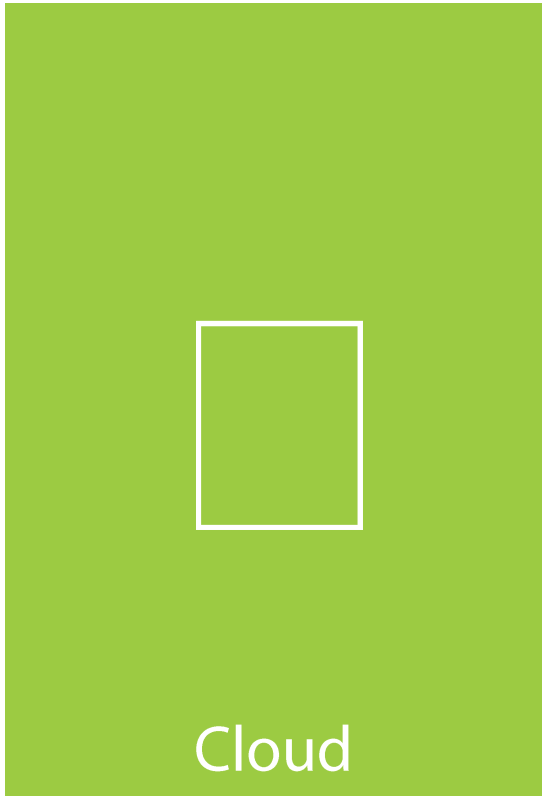
Developer

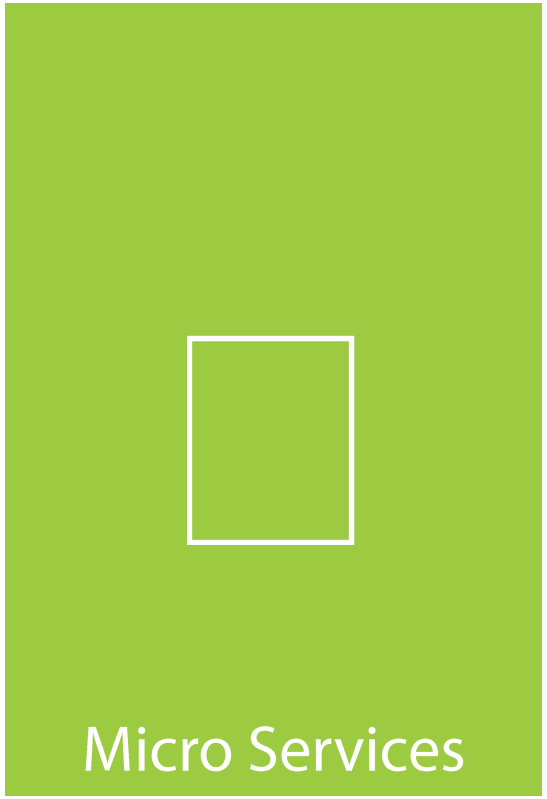


Usage of Java EE









Module Outline

- Enterprise application
- eCommerce website
- Persistence
- How do we usually develop it?
- What's wrong with the way we do?
- How can Java EE help?

Today's Applications

- Enterprises need
  - Applications to fulfill business needs
  - Business is complex and changes quickly
  - Distributed, available, internationalized
- Applications need
  - Lower response time
  - Crash prevention
  - Mobile and web interfaces
- Java Enterprise Edition

CD-Book Store

E-commerce web application

Books and CDs

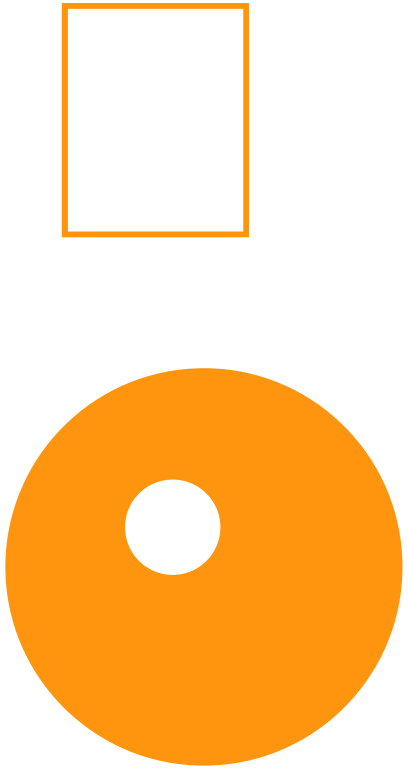
Authentication

Shopping cart

REST interface

What's Wrong with Developing with Java SE?

- Java SE or Java EE?
- Java Standard Edition
  - APIs handle collections
  - JVM is a container
  - Lower-level services
- Java Enterprise Edition
  - Handles transactions, messaging, persistence…
  - Code runs in a container
  - Higher-level services



Mapping Objects to a Relational Database

- Objects
- Relational databases
- Object-relational mapping
- Rely on external frameworks
- Low level Java SE APIs
- JDBC

A Book Class

```
public class Book {

  private Long id;
  private String title;
  private String description;
  private Float unitCost;
  private String isbn;

  // Constructors, getters & setters
}
```

A Main Class Manipulating a Book

```
public class Main {

  public static void main(String[] args) {

    persistBook(new Book(1L, "H2G2", "Best Scifi Book", 12.5f, "1234-5678-5678"));

    Book book = findBook(1L);

    System.out.println(book);
  }
```

Getting a Database Connection

```
static {
  try {
    Class.forName("org.apache.derby.jdbc.ClientDriver");
  } catch (ClassNotFoundException e) {
    e.printStackTrace();
  }
}

private static Connection getConnection() throws SQLException {
  return DriverManager.getConnection(
         "jdbc:derby://localhost:1527/module01-db", "app", "app");
}
```

Persisting a Book to the Database

```
private static void persistBook(Book book) {

  String query = "INSERT INTO BOOK (ID, TITLE, DESCRIPTION, UNITCOST, ISBN)
                                VALUES (?, ?, ?, ?, ?)";

  try (PreparedStatement stmt = getConnection().prepareStatement(query)) {

    stmt.setLong(1, book.getId());
    stmt.setString(2, book.getTitle());
    stmt.setString(3, book.getDescription());
    stmt.setFloat(4, book.getUnitCost());
    stmt.setString(5, book.getIsbn());

    stmt.executeUpdate();
  }
}
```

Retrieving a Book from the Database

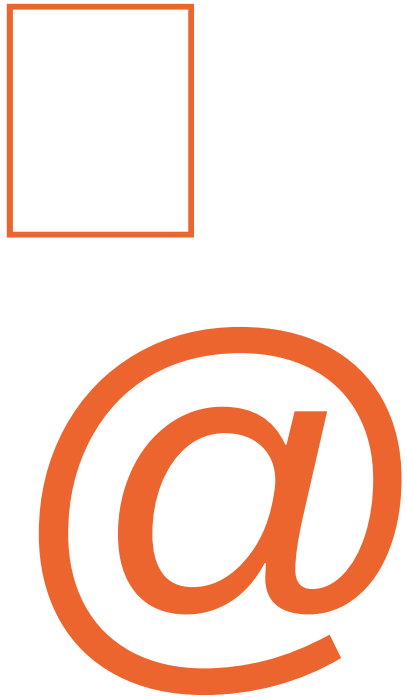
```
private static Book findBook(Long id) {
  Book book = new Book(id);
  String query = "SELECT ID, TITLE, DESCRIPTION, UNITCOST, ISBN
                  FROM BOOK WHERE ID = ?";

  try (PreparedStatement stmt = getConnection().prepareStatement(query)) {

    stmt.setLong(1, id);
    ResultSet rs = stmt.executeQuery();
    while (rs.next()) {
      book.setTitle(rs.getString("TITLE"));
      book.setDescription(rs.getString("DESCRIPTION"));
      book.setUnitCost(rs.getFloat("UNITCOST"));
      book.setIsbn(rs.getString("ISBN"));
    }
  }
  return book;
}
```

What's Wrong with Java SE APIs?

- SQL is not Java
- JDBC is a low level API
- SQL is not easy to refactor
- JDBC is verbose
- Hard to read
- Hard to maintain

Manipulating Data with JPA

- JDBC is a low level API part of Java SE
- Object-relational mapping
- JPA higher level part of Java EE
- Metadata mapping
- Removes boiler plate code
- (C)reate, (R)ead, (U)pdate, (D)elete
- Object-oriented query language



A Book Entity

```
@Entity
public class Book {

  @Id
  private Long id;
  private String title;
  private String description;
  private Float unitCost;
  private String isbn;

  // Constructors, getters & setters
}
```

A Service Manipulating a Book Entity

```
@Transactional
public class BookService {

  @Inject
  private EntityManager em;

  public void persistBook(Book book) {
    em.persist(book);
  }

  public Book findBook(Long id) {
    return em.find(Book.class, id);
  }
}
```

Thanks to Java EE!

- No manual mapping
- No SQL statements
- Simplified programming model
- Convention over Configuration
- Brings higher level services
- Metadata
- Information understood by the container

Summary

- Enterprise applications are powerful
- Complex to develop
- Java EE reduces complexity
- Programming model
- APIs
- Runtime environment
- Concentrate on business requirements

What's Next

- Enterprise applications
- Architectural layers
- Java Enterprise Edition
- Specifications in Java EE 7
- Programming model
- Containers implementing Java EE 7
- Java Community Process