Java EE: The Big Picture

Introduction

Antonio Goncalves

@agoncal | www.antoniogoncalves.org

Course Outline

- Introduction
- Java platform
- Java EE platform and programming model
- Enterprise applications
- What is Java EE?
- Is Java EE right for your organization?

Audience

- Technical
- Business
- CTO
- Technical Architect
- Team Leader
- Developer

Module Outline

- Definitions
- Java Platform
- Enterprise Applications
- Programming Model

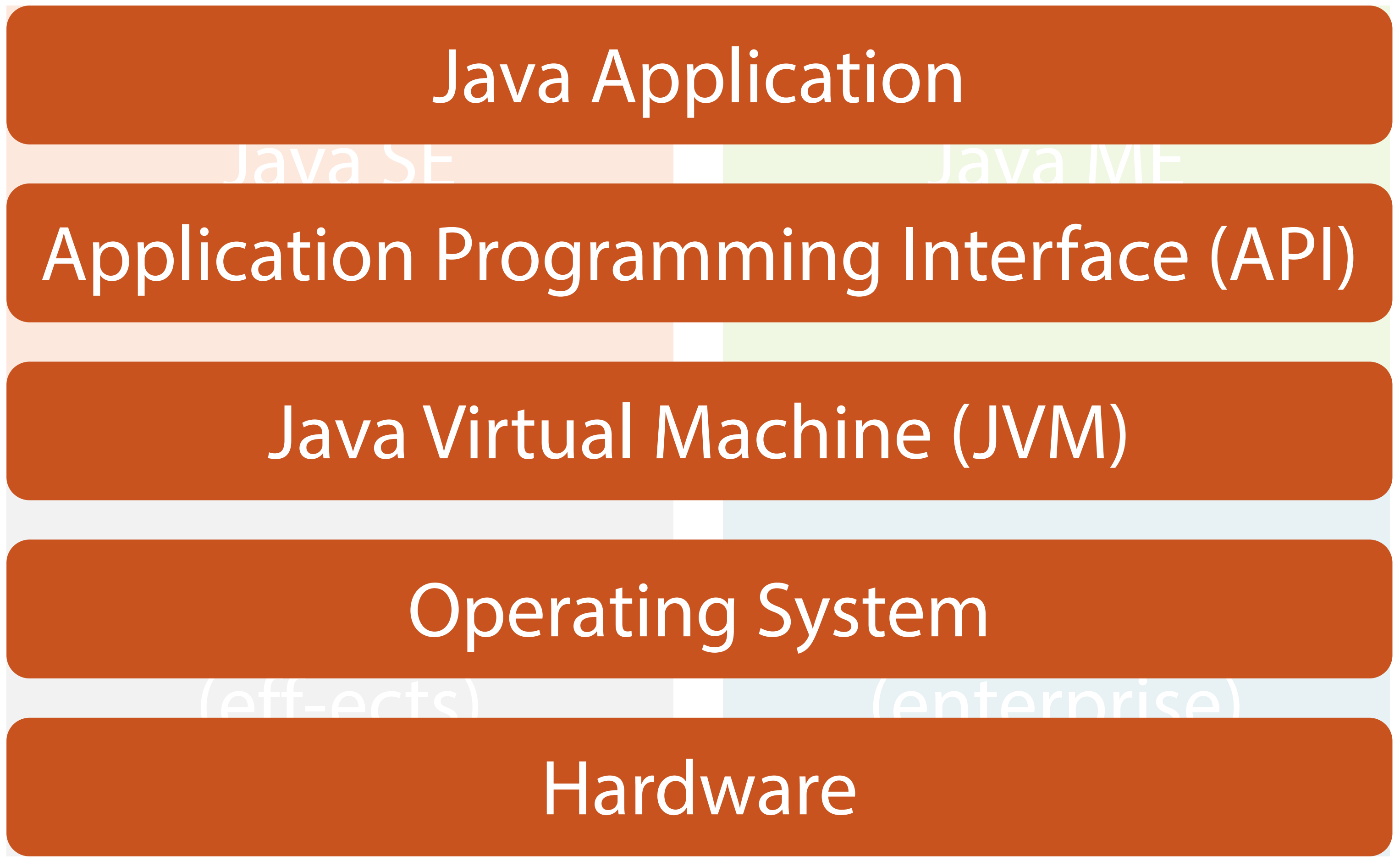
The Java Platform

- Java technology
- Java
  - Object-oriented programming language
  - C-like syntax
  - Portable
- Java platform
  - Environment
  - Java applications run

The Four Platforms

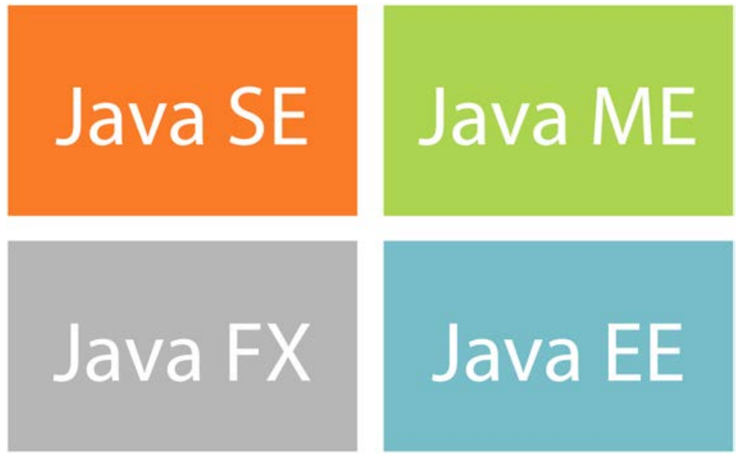
Java SE (standard)	Java ME (micro)
Java FX (eff-ects)	Java EE (enterprise)

The Four Platforms



Each Platform

- JVM and API
- Run on any compatible system
- Take advantage of the Java language
- One of the most widely used platforms
- Development of just about any solution
- Enterprise applications



Java SE

- Java Standard Edition
- Core platform
- Core libraries and APIs
- Basic types and objects to high-level classes
- JVM
- Development tools
- Deployment and monitoring
- …

JDK (Java Development Toolkit)

Java SE
Java ME
Java FX
Java EE

Java ME

- Java Micro Edition
- Subset of Java SE
- Mobile devices
- Small-footprint JVM
- Small devices
- Internet of things

Java SE
Java ME
Java FX
Java EE

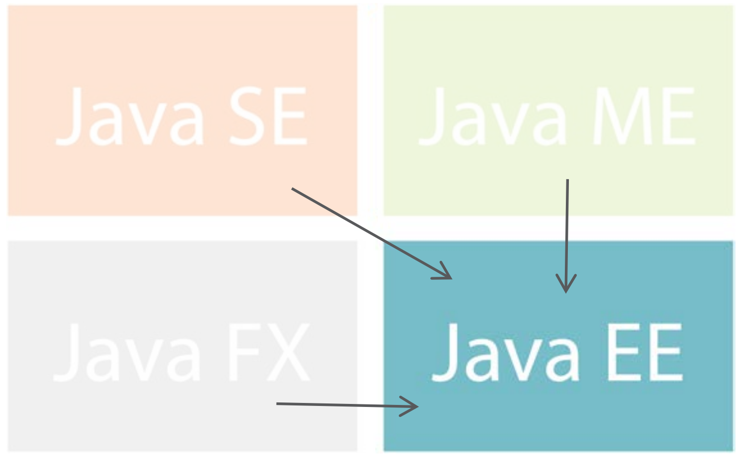
Java FX

- Rich internet applications
- User-interface API
- Hardware-accelerated graphics
- High-performance clients
- Modern look-and-feel
- Connect to remote services

Java SE
Java ME
Java FX
Java EE

Java EE

- Java Enterprise Edition
- Java EE extends Java SE
- Enterprise software
- Large scale
- Distributed system
- Consider Java EE instead of Java SE



Applications

Java SE

Java ME

Java FX

Java EE

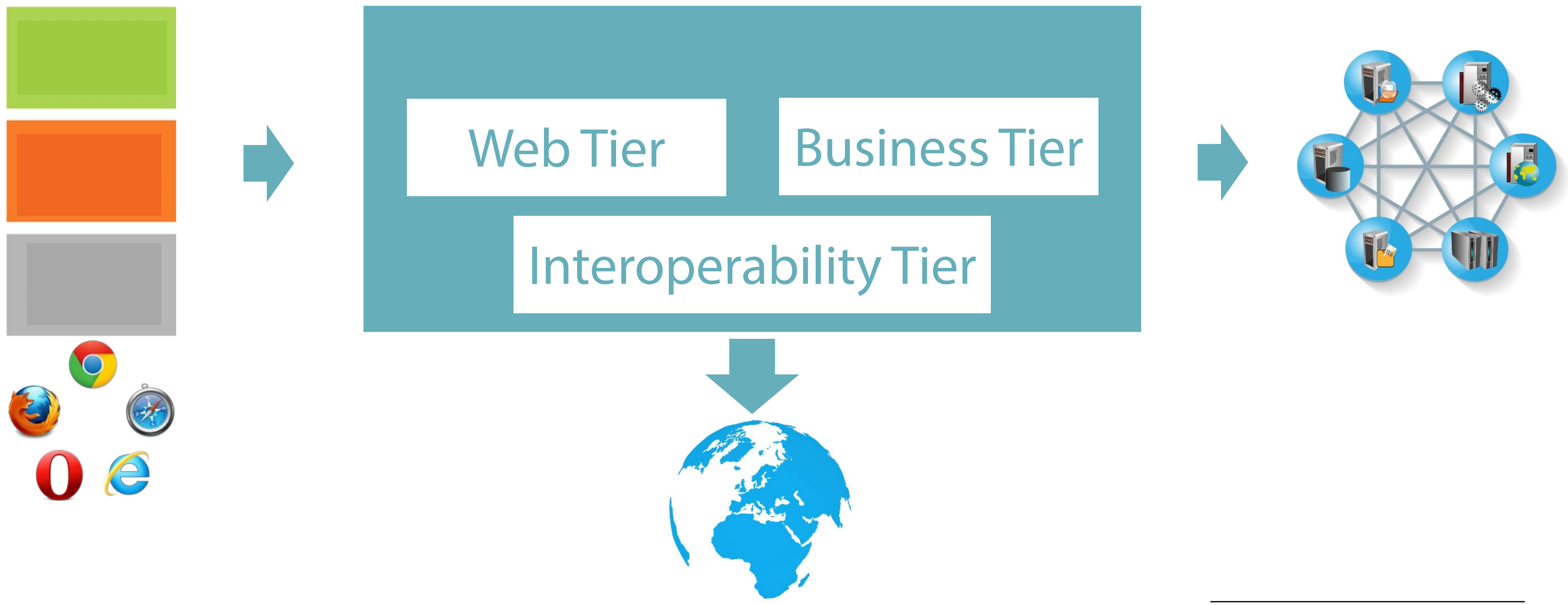
Enterprise Applications

- Multi-tiered
- Scalable
- Reliable
- Secure

} Enterprise application

- Solve the problems of large enterprises
- For individual developers and small organizations

Tiered Application



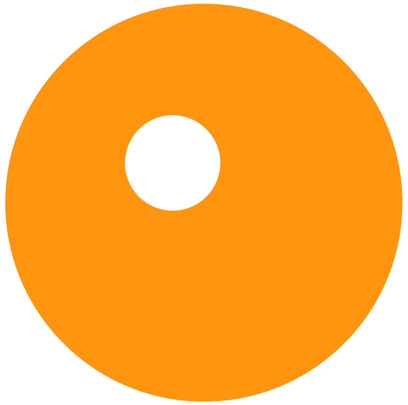
Client tier

Middle tier

EIS tier

Java SE vs. Java EE

- Java SE
  - APIs handle collections
  - The JVM is a container
  - Lower-level services
- Java EE
  - APIs handle transactions, messaging, persistence…
  - Code runs in a container
  - Higher-level services



Java EE Reduces Complexity

- Enterprise applications are powerful
- But complex
- Java EE reduces complexity
- Programming model
- APIs
- Runtime environment
- Developers concentrate on business requirements

The Java EE Programming Model

- Simplified programming model
- Convention over Configuration
- Container takes default decisions
- Brings services
- Use metadata to deviate from convention
- Information understood by the container

Manipulating Persisted Data in Java SE

```
public class Book {

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

  // Constructors, getters & setters
}
```

Java SE Class Manipulating a Book

```
public class Main {

  public static void main(String[] args) {

    persistBook(new Book(1L, "H2G2", "Scifi Book", 12.5f, "1234-5678-5678", 247));

    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

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

  String query = "INSERT INTO BOOK (ID, TITLE, DESCRIPTION, PRICE, 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.getPrice ());
    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, PRICE, 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.setPrice (rs.getFloat("PRICE"));
      book.setIsbn(rs.getString("ISBN"));
    } }
  return book;
}
```

What’s Wrong with Java SE?

- SQL is not Java
- Low-level API (JDBC)
- SQL is not easy to refactor
- JDBC is verbose
- Hard to read
- Hard to maintain

Manipulating Persisted Data in Java EE

```java
@Entity
public class Book {

  @Id
  private Long id;
  private String title;
  private String description;
  private Float price;
  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);
  }
}
```

Advantages of Java EE

- No manual mapping
- No SQL statements
- Non intrusive
- Metadata (`@Entity`, `@Id`)
- Higher-level of abstraction

Convention over Configuration

```
@Entity
@Table(name = "t_book")
public class Book {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Long id;
  @Column(name = "book_title", nullable = false)
  private String title;
  @Column(length = 2000)
  private String description;
  @Column(name = "unit_cost")
  private Float price;
  private String isbn;

  // Constructors, getters & setters
}
```

Summary

- Java ecosystem
- Four platforms
- Enterprise application
- Java EE programming model
  - Convention over configuration
  - Java EE container

What’s Next

- What is Java EE?
- Internal architecture
  - Components
  - Container
  - Services
- Implementations
- Demo