Testing the Java EE Application

Antonio Goncalves
JAVA CHAMPION

@agoncal www.antoniogoncalves.org

Previous Module

- Book Repository
- Java Persistence API (JPA)
- CRUD operations
- Java Persistence Query Language (JPQL)
- Querying entities
- Java Transaction API (JTA)
- Transaction demarcation

Overview

- Understanding Testing
- Unit testing
- Integration testing
- JUnit
- Arquillian

What Is Testing?

- Assert the code is correct
- Automate the test execution
- Help designing APIs
- Tests document the code
- Several testing strategies

Unit Test	Integration Test
Single functionality	Several pieces work together
In isolation	Cover whole application
Narrow business scope	More effort to put together
Fast, simple, easy-to-run	Require external resources
Mock	No mock
No dependencies on outside systems	Demonstrates the system works
JUnit	Arquillian

Unit Tests

Test one class

In isolation

No container service

Mock container services

```java
public class BookRepository {

  @PersistenceContext
  private EntityManager em;

  @Transactional(REQUIRED)
  public Book create(Book book) {
    em.persist(book);
    return book;
  }
}
```

```java
public class BookRepositoryTest {

  @Test
  public void shouldCreateABook() {
    BookRepository repository =
                new BookRepository();

    Book book = new Book("...");
    book = repository.create(book);

    assertNotNull(book);
  }
}
```

Integration Tests

- Access the container services
- Package the business code
- Test class in the package
- Deploy it to WildFly
- Execute tests inside the container

```java
public class BookRepository {

  @PersistenceContext
  private EntityManager em;

  @Transactional(REQUIRED)
  public Book create(Book book) {
    em.persist(book);
    return book;
  }
}
```

```java
@RunWith(Arquillian.class)
public class BookRepositoryTest {

  @Inject
  private BookRepository repository;

  @Test
  public void shouldCreateABook() {
    BookRepository repository =
                new BookRepository();

    Book book = new Book("...");
    book = repository.create(book);

    assertNotNull(book);
    assertNotNull(book.getId());
  }
}
```

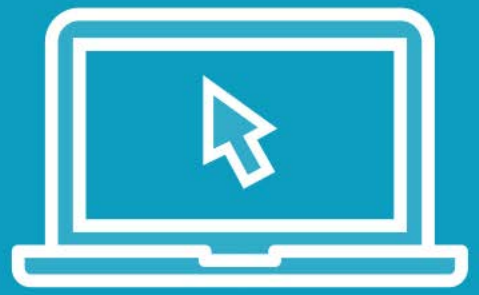
Packaging Tests

- Package the business code
- ShrinkWrap
- Creates deployable archives
- Jar, war, zip files
- Add external dependencies
- Archive is then deployed

@Deployment

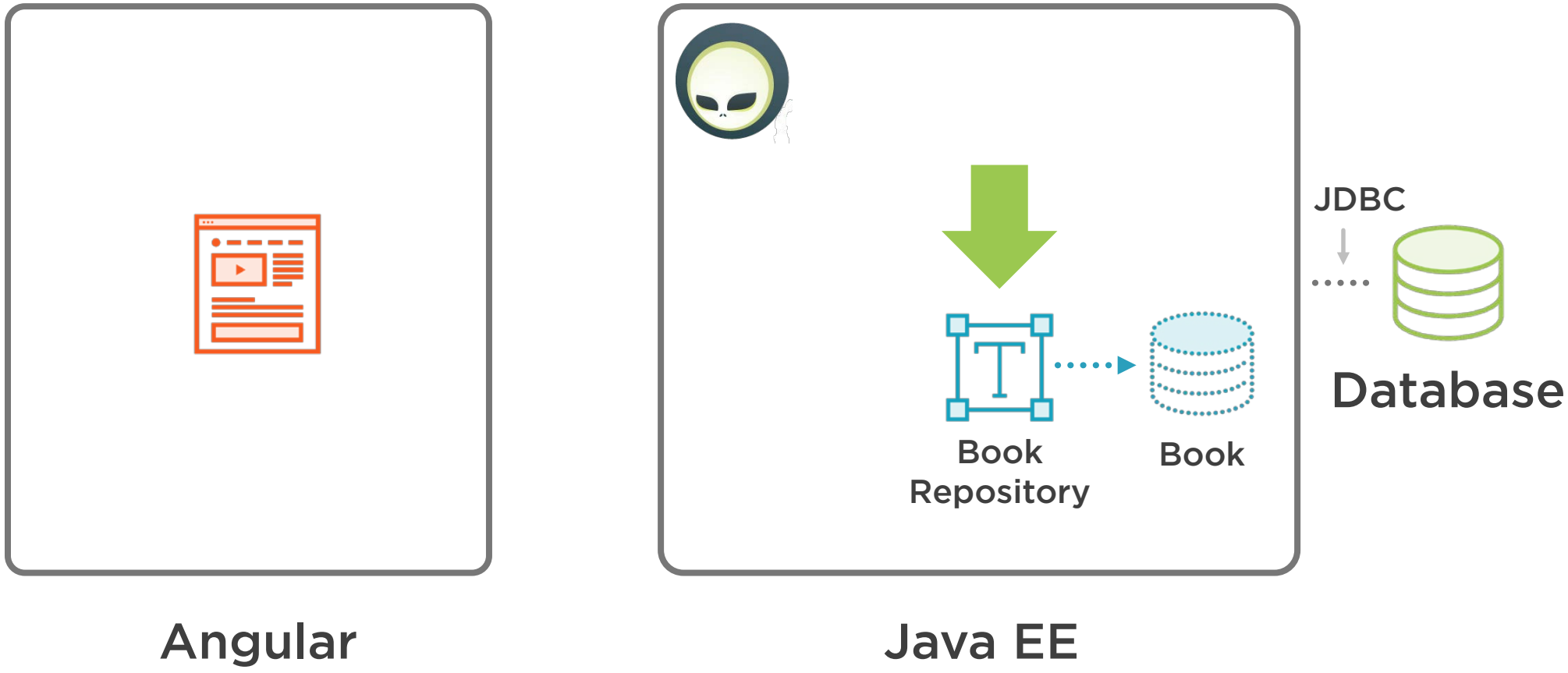
```
@RunWith(Arquillian.class)
public class BookRepositoryTest {

  @Test
  public void shouldCreateABook() {...}

  @Deployment
  public static Archive<?> createDeploymentPackage() {

    return ShrinkWrap.create(JavaArchive.class)
        .addClass(Book.class)
        .addClass(Language.class)
        .addClass(BookRepository.class)
        .addAsManifestResource("persistence.xml");
  }
}
```

Demo



- Integration test
- JUnit and Arquillian
- Testing the Book Repository
- ShrinkWrap
- Testing the CRUD operations
- WildFly

Anatomy of the BookStore Application



Summary

- Defined what testing is
- Unit tests
- Integration tests
- JUnit tests in isolation
- Arquillian tests inside the container

Next Module

- Business constraints
- Validate constraints
- Store and process valid data
- Bean Validation
- Add more test