Getting Started on Enterprise Systems

Established in Being
Perform Action

Enterprise Systems Main Points

• Enterprise Architectures incorporate many architectural, networking, and design patterns to streamline the development of enterprise applications dealing with large amounts of data and users.

• By using Enterprise Architecture Design Mechanisms and Design Implementations like Spring MVC, JPA , JSF, and other services we can build successful and extensible industrial grade enterprise applications with standard components.

- **Science of Consciousness**

“Established in being, perform action” is a recipe for highly successful action. By experiencing pure consciousness during TM and then bringing that coherence and intelligence into our software development, we become better designers, architects, and coders.

Java EE and Spring - Overview

- Java Enterprise Edition, launched 1999
  - At first, very heavy-weight -- J2EE
- Spring (a competitor to J2EE) was created to provide a lighter-weight EA solution -- **2002**
- Java EE7 – **2013** - simpler, easier to use and configure, and follows examples from STS (Spring Tool Suite)
- Spring Boot – **2015** - is the Spring answer to simplifying Spring configuration

Java EE and Spring - Overview

- Java EE
  - Provides a set of standards for Transactions (JTA), Messaging (JMS), Persistence (JPA), Presentation (JSF), Business Logic and Control (EJB), and web services (JAX-WS & JAX-RS).
  - Hibernate is JPA compliant (JPA was based on Hibernate)

- Spring is open source and may or may not follow the Java EE standards.

- STS adds new capabilities with each release. Support for Microservices is a good example.

Java EE and Spring Containers

- Java EE is a superset of Java SE.
  - All Java SE API libraries can be used in Java EE.

  Spring also supports Java SE
- Both Java EE and Spring provide built-in services in containers.

Containers and Deployment

- Containers
  - Interface between component and low-level platform-specific functionality
  - In JEE7 - A container for every component type
    - Web
    - EJB
    - application client
    - A WAR, EAR, or JAR would be created for JEE7 deployment
- In Spring a WAR file would be deployed on a Tomcat server
- With SpringBoot a simple JAR file can be run on the embedded server.

Configuring Containers

- JEE7 Convention over Configuration.
  - Programming by exception.
- Meta-data programming.
  - Annotations.
  - Deployment descriptors.
- Spring Boot – provides autoconfiguration to provide the beans your application needs.
- With Spring Boot *starters* common groups of dependencies are added to your build.

Configuring Containers

- What are annotations?

In your code identify what container services you need for the class.

- There are hundreds of JEE7 and Spring annotations:

- We can create a useful app with a handful of standard annotations or… we could create a monster component..

Configuring Containers

AntiPattern – *a monster component*

*http://*antoniogoncalves.org/2013/07/03/monster-component-in-java-ee-7/

- Java EE is a managed environment

1. take a Java class, add a @Stateless annotation, and the container gives you transactions, security, pooling…
2. take another class, add a @Path annotation and the container gives you REST invocation.
3. add both @Stateless and @Path to the same class, you accumulate the services of a stateless EJB and a REST Web Service.
4. Continue (Goncalves stopped at 17 annotations…….)

Configuring Containers

- What are Deployment descriptors?

xml file deployed with the component in the container.

- Java EE: application.xml, application-client.xml
- CDI: beans.xml
- JCA: ra.xml (Java Connector Architecture)
- EJB: ejb-jar.xml
- JSF: faces-config.xml
- JPA: persistence.xml
- Bean Validation: validation.xml
- Servlet: web.xml, web-fragment.xml
- SOAP and RESTful WS: webservice.xml

Spring Boot Configuration

- We only need to set up spring boot *starter* dependencies in our pom.xml

For example, as in lab 4 Spring Boot Demo AddStudent:

spring-boot-starter-thymeleaf

spring-boot-starter-web

mysql-connector-java

spring-boot-starter-test

spring-boot-starter-data-jpa

Spring Boot Configuration

•We configure our project in:
src/main/resources/ application.properties

- spring.datasource.url = jdbc:mysql://localhost:3306/msched?createDatabaseIfNotExist=true
- spring.datasource.driverClassName = com.mysql.jdbc.Driver
- spring.datasource.password=root
- spring.datasource.username=root
- spring.jpa.properties.hibernate.dialect: org.hibernate.dialect.MySQL5Dialect
- spring.jpa.hibernate.ddl-auto = create
- spring.jpa.show-sql= true
- spring.jpa.properties.hibernate.hbm2ddl.import_files: import_data.sql
- spring.messages.basename=messages
- ##server.contextPath=/StudentSpringBoot

Spring Boot Configuration

•We direct Spring Boot to run our application with the following simple class

```
@SpringBootApplication
public class StudentSpringBootApplication  {

public static void main(String[] args) {
SpringApplication.run(
        StudentSpringBootApplication.class, args);
}
```

Spring and Spring Boot Lab

Assignment 5 Spring Proof of Concept

- ***Each person in the group*** will have a POC running on their development machine that:

1) takes in User Input to the User Interface/web page
2) input is handled by that page's controller
3) an entity class is created or read from the DB
4) the entity class is updated and saved to the DB
5) data from an entity class is displayed to the UI/web page

Spring and Spring Boot Lab

We have a four demos to start on your Proof of Concept.

See attachments to *Assignment 5 Spring MVC Proof of Concept:*

- SpringMVCDemo
- SpringBootDemoAddStudent
- SpringBoot Demo steps for Creating Entry and Block
- SpringBoot Demo for AddStudent Using JSPs instead of Thymeleaf

Enterprise Systems Main Points

• Enterprise Architectures incorporate many architectural, networking, and design patterns to streamline the development of enterprise applications dealing with large amounts of data and users.

• By using Enterprise Architecture Design Mechanisms and Design Implementations like Spring MVC, JPA , JSF, and other services we can build successful and extensible industrial grade enterprise applications with standard components.

- **Science of Consciousness**

“Established in being, perform action” is a recipe for highly successful action. By experiencing pure consciousness during TM and then bringing that coherence and intelligence into our software development, we become better designers, architects, and coders.