



The Spring Framework: Foundations

Originals of Slides and Source Code for Examples:
<http://courses.coreservlets.com/Course-Materials/spring.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live Spring & Hibernate training, see courses at <http://courses.coreservlets.com/>.



Taught by the experts that brought you this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
 - Java 5, Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF, Ajax, GWT, custom mix of topics
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Spring, Hibernate/JPA, EJB3, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

- **Motivation**
- **Spring Hello World**
- **POJO development**
- **Runtime environment**
- **Dependency injection**
- **Inversion of control**

5

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com



Motivation

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Software Development Challenges

- **Solutions are complex**
- **Requirements are constantly in flux**
- **Software architecture must be flexible**
- **Software components must be verifiable**

7

Java EE training: <http://courses.coreservlets.com>

EJB 2.0 Approach

- **Complex products**
- **Unmaintainable systems**
- **Non-portable, framework-committed business components**
- **Unpredictable systems**

8

Java EE training: <http://courses.coreservlets.com>

Spring Approach

- **Products based on simplicity**
- **Maintainable systems**
- **Framework-independent software**
- **Portable components**
- **Testable components**
- **Reliable and predictable systems**

Pure Java

- **Founded on POJO-based development**
 - Ordinary Java classes that follow no special APIs
- **Non-invasive for pre-existing POJOs**
- **Rewards framework-independent business logic**
- **Encourages new software to be written as POJOs**
- **Results in highly portable, reusable, and verifiable software**

More With Less Custom Code

- **Expand capabilities with less code**
- **Extensive and tested service abstractions**
 - Email
 - JMS
 - JMX
 - JSF
 - JDBC
 - etc...
- **Replaces generic corporate libraries**
- **Mitigates custom integration activities**
- **Consistency eases integration because spring platform is easy to use**

11

Java EE training: <http://courses.coreservlets.com>

Modular

- **Helps only where needed**
 - Modularity allows only relevant components to be introduced into the application
 - For instance choose one:
 - Spring BeanFactory
 - Spring JMX
 - Spring JDBC
 - Framework can be interfaced in deep or shallow layers.
 - Interfaces are consistent at each layer
- **Turn-key solution**
 - Spring components can be integrated quickly, with minimal effort and predictable results
 - Interfaces are clear and consistent

12

Java EE training: <http://courses.coreservlets.com>

Widely Available

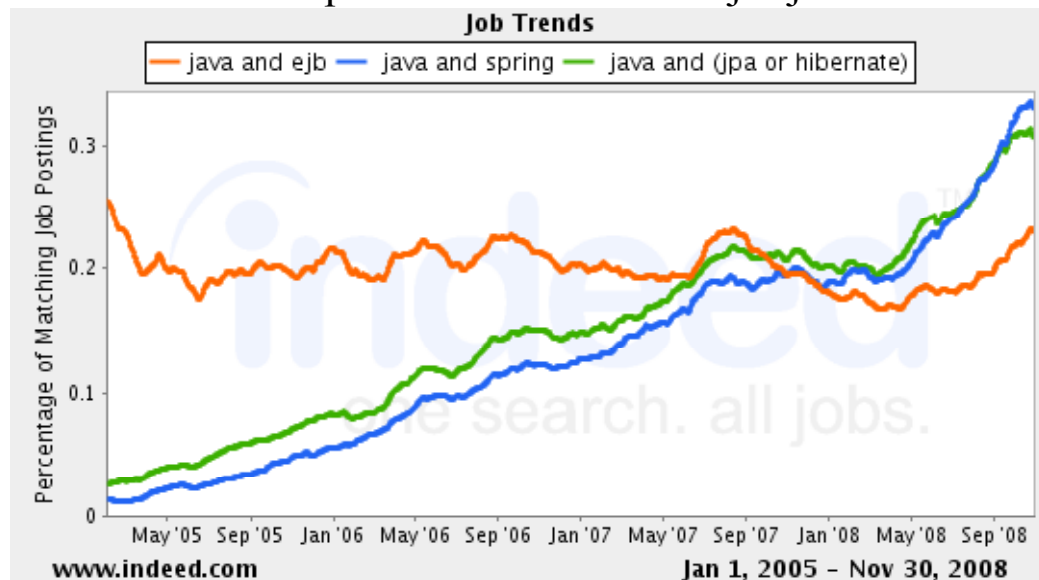
- **Spring is integrated into numerous frameworks**
- **Broad adoption possible because the container is portable and lightweight**
 - The container itself is designed as a POJO
- **Integration without third-party support**
- **Performance overhead is rarely a consideration**

13

Java EE training: <http://courses.coreservlets.com>

Spring Jobs

- **From indeed.com**
 - Claims to compile data from most major job sites



14

Java EE training: <http://courses.coreservlets.com>



Spring Setup

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Spring Download

- **<http://www.springframework.org/download>**
 - Current version: 2.5.5 (6/2008)
 - Requires JDK 1.4+
 - [spring-framework-2.5.5-with-dependencies.zip](#)
 - Spring Framework binaries and source
 - Third-party binaries
 - Documentation
 - API
 - HTML reference
 - Project samples
 - HOW-TO guides

Spring Blank Project

- **spring-blank.zip**

- Available from
<http://courses.coreservlets.com/Course-Materials/spring.html>

| Path | Description |
|------------------|--|
| src | Empty applicationContext.xml. For new Java source files. |
| lib | Minimum Spring JARs for API and runtime access to the Spring IoC container |
| build.xml | Optional Apache ANT build configuration |
| pom.xml | Optional Maven 2 build configuration |

Spring Blank Project and Eclipse

- **Download spring-blank.zip**

- <http://courses.coreservlets.com/Course-Materials/spring.html>

- **Import archive as an existing project into the current workspace**

- From the Eclipse menu bar select **File** and **Import**
- From the **Import (Select)** dialog, select **Existing Projects into Workspace** and **Next**
- From the **Import (Import Projects)** dialog, select the radio button **Select archive file** and **Browse**
- Locate and select **spring-blank.zip** and select **Open**
- Verify the project entry, **spring-blank**, to be present in the project list
- Select **Finish**

Spring Blank Project and Apache Ant

- Download and unpack spring-blank.zip
- Install Apache Ant, version 1.6.5+
- Execute various Ant build commands

| Command | Description |
|----------------|--|
| clean | Removes the build directory target and all nested build artifacts |
| compile | Compiles production Java source contents under src/main/java and places class binaries into target/classes |
| test | Executes the compile command and compiles and executes tests found under src/test/java |
| package | Packages production Java source and resource contents into a jar file. The jar package is placed in the build directory, target |

19

Java EE training: <http://courses.coreservlets.com>

Spring Blank Project and Maven 2

- Download and unpack spring-blank.zip
- Execute various Maven commands

| Command | Description |
|----------------|--|
| clean | Removes the build directory target and all nested build artifacts |
| compile | Compiles production Java source contents under src/main/java and places class binaries into target/classes |
| test | Executes the compile command and compiles and executes tests found under src/test/java |
| package | Packages production Java source and resource contents into a jar file. The jar package is placed in the build directory, target |

- See pom.xml configuration for additional dependency options

20

Java EE training: <http://courses.coreservlets.com>

Spring Documentation

- **Top-level documentation page**
 - <http://www.springframework.org/documentation>
- **Wiki**
 - <http://opensource.atlassian.com/confluence/spring>
- **Forum**
 - <http://forum.springframework.org>
- **Books**
 - *Spring Recipes*. APress 2008
 - *Spring in Action*. Manning 2007
 - *Agile Java Development with Spring, Hibernate and Eclipse*. Sams 2006

21

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com

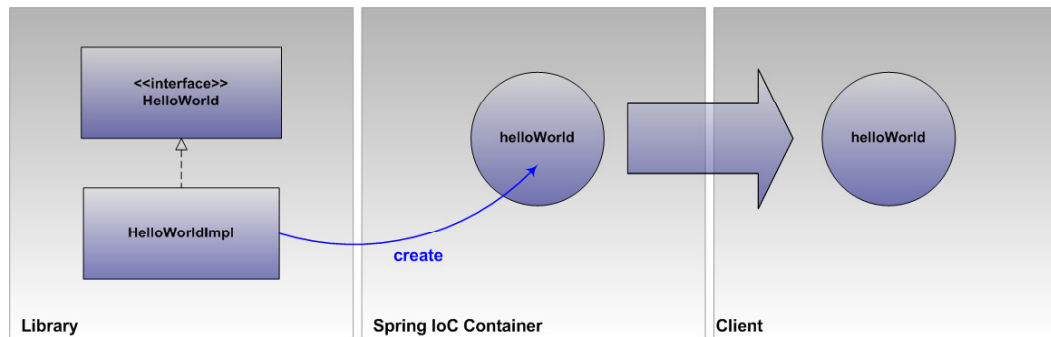


Spring Hello World

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Spring Hello World

- **Code a plain Java class model**
 - Use the interface pattern by coding a **HelloWorld** interface and a **HelloWorldImpl** implementation
- **Configure the Spring IoC container**
- **Instantiate the Spring IoC container**
- **Acquire the object from the Spring IoC container**
 - The client must only have knowledge of the interface, **HelloWorld**

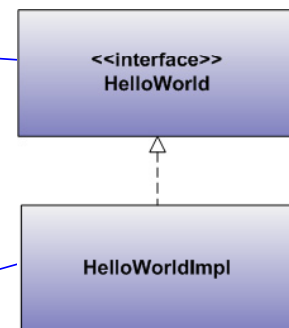


23

Java EE training: <http://courses.coreservlets.com>

Plain Java Class Model

```
public interface HelloWorld {  
    public void execute();  
}  
  
public class HelloWorldImpl  
    implements HelloWorld {  
    public void execute() {  
        System.out.println("Hello World!");  
    }  
}
```



24

Java EE training: <http://courses.coreservlets.com>

Spring IoC Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

  <bean id="helloWorld"
        class="coreservlets.HelloWorldImpl" />




</beans>
```

25

Java EE training: <http://courses.coreservlets.com>

Executing Spring Hello World

```
import org.springframework.beans.factory.*;
import org.springframework.context.support.*;

public class Main{
  public static void main(String[] args) {
    BeanFactory beanFactory =  Spring IoC container
      new ClassPathXmlApplicationContext (
        "applicationContext.xml");
    HelloWorld helloWorld =  HelloWorld interface
      (HelloWorld)  beanFactory.getBean("helloWorld");
    helloWorld.execute();
  }
}
```

Standard output

```
Hello World!
```

26

Java EE training: <http://courses.coreservlets.com>



Background: POJO Development

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Introduction

- **Plain Old Java Object**
- **What is it?**
 - Business logic
 - Framework independent
- **What it's not**
 - Limited to the value object pattern
 - Framework implementation software
- **Features**
 - Portable
 - Testable
 - Flexible

POJO Development Process

- **Describe the system agents and interactions**
 - POJO behavioral classes, domain model, and dependencies
- **Determine component responsibilities**
 - Methods
- **Identify information items discovered during program execution**
 - Method parameters
- **Identify information available during initialization**
 - Initialization parameters for constructor, setter, or factory

29

Java EE training: <http://courses.coreservlets.com>

POJO Development Process Example



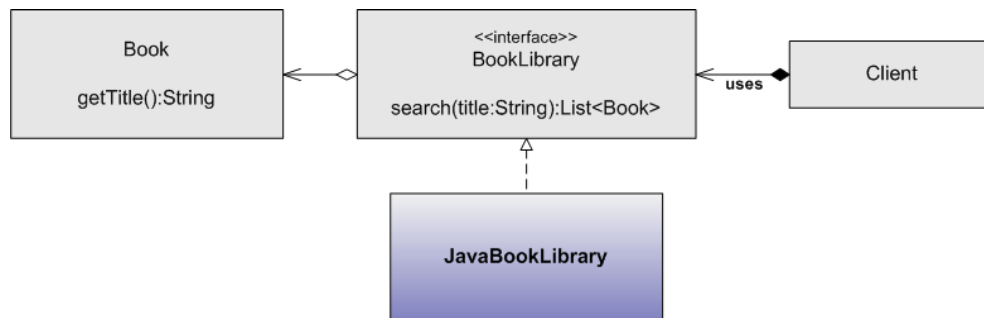
- **Agents**
 - BookLibrary and Client
- **Interactions**
 - Client uses BookLibrary
 - BookLibrary aggregates Book
- **Responsibilities**
 - BookLibrary must search for books by title
 - Clients must supply search parameters; i.e. title values

30

Java EE training: <http://courses.coreservlets.com>

POJO Development Process

- Develop implementation

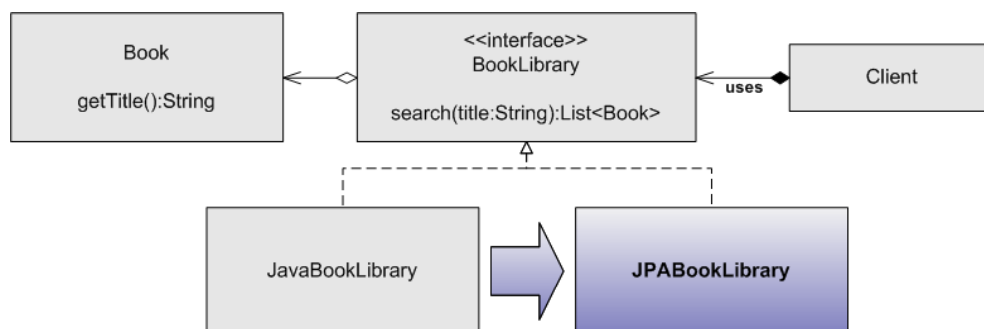


31

Java EE training: <http://courses.coreservlets.com>

POJO Development Process

- Plan for change

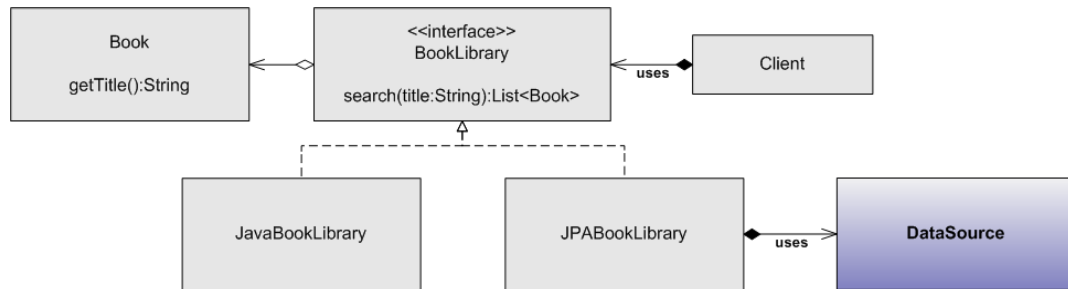


32

Java EE training: <http://courses.coreservlets.com>

POJO Development Process

- Plan for new and additional dependencies

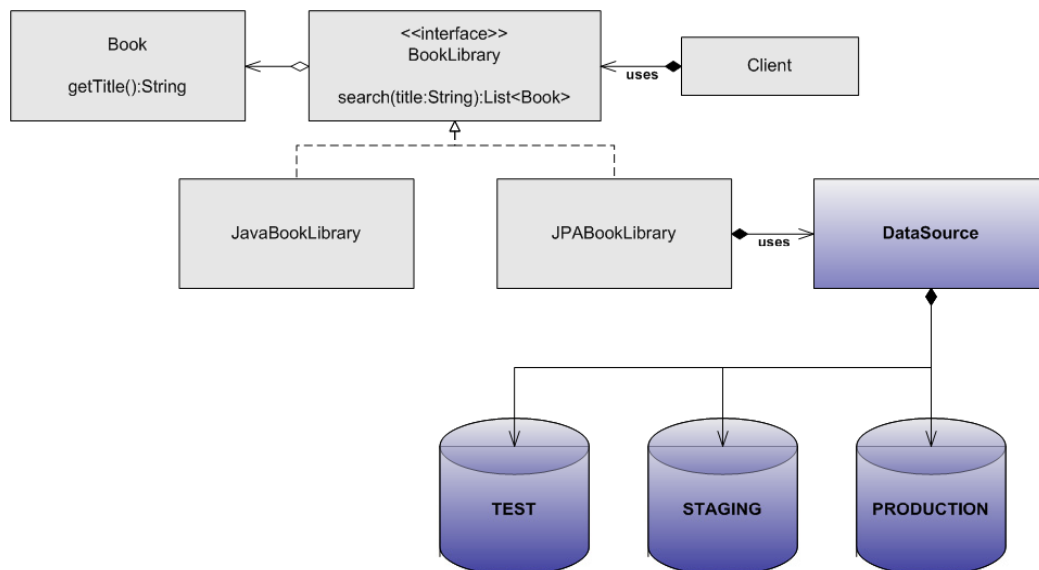


33

Java EE training: <http://courses.coreservlets.com>

POJO Development Process

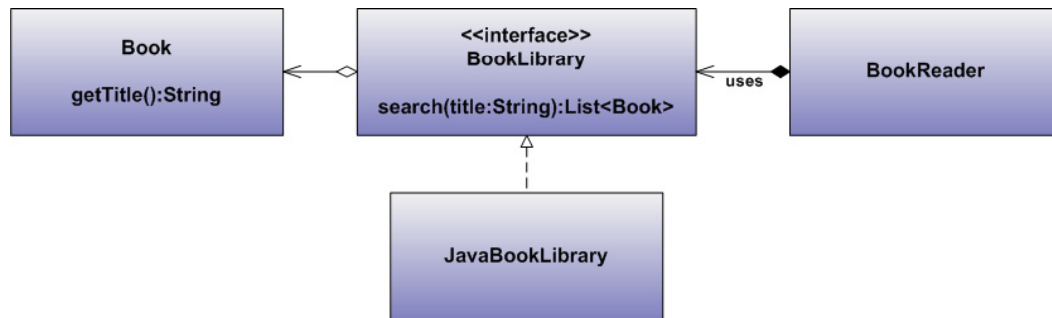
- Plan for complex configuration requirements



34

Java EE training: <http://courses.coreservlets.com>

POJO Implementation Example

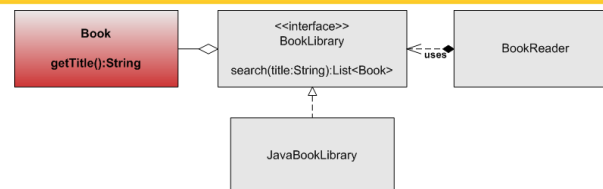


35

Java EE training: <http://courses.coreservlets.com>

Book Implementation

```
public class Book {  
  
    private String title;  
  
    public Book(String title) {  
        this.title = title;  
    }  
  
    public String getTitle() {  
        return title;  
    }  
  
    public String toString() {  
        return title;  
    }  
}
```



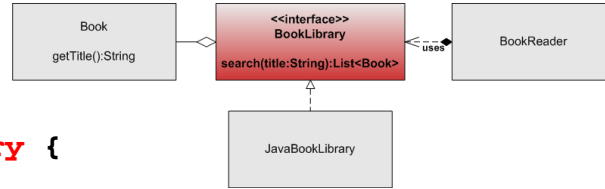
36

Java EE training: <http://courses.coreservlets.com>

BookLibrary Implementation

```
import java.util.List;
```

```
public interface BookLibrary {  
  
    public List<Book> search(String title);  
  
}
```

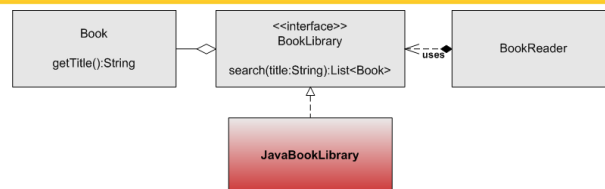


37

Java EE training: <http://courses.coreservlets.com>

BookLibrary Implementation

```
public class JavaBookLibrary  
    implements BookLibrary {  
  
    private List<Book> books;  
  
    public JavaBookLibrary() {  
        this.books = Arrays.<Book>asList(  
            new Book("Core Servlets and JavaServer Pages"),  
            new Book("More Servlets and JavaServer Pages"));  
    }  
  
    public List<Book> search(String title) {  
        List<Book>results = new ArrayList<Book>();  
        for(Book book : books){  
            if(book.getTitle().contains(title)){  
                results.add(book);  
            }  
        }  
        return results;  
    }  
}
```



38

Java EE training: <http://courses.coreservlets.com>

Client Implementation

```
public class BookReader {
```

```
    private BookLibrary bookLibrary;
```

```
    public BookReader() {
```

```
        this.bookLibrary = new JavaBookLibrary();
```

```
    }
```

```
    public List read() {
```

```
        List<Book> books = bookLibrary.search("Java");
```

```
        for(Book book : books) {
```

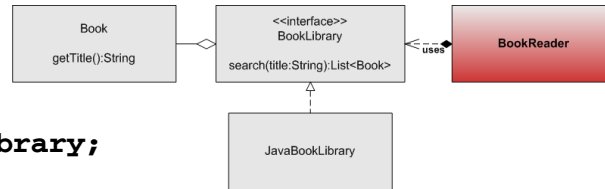
```
            System.out.printf("Reading: %s%n", book);
```

```
        }
```

```
        return books;
```

```
    }
```

```
}
```



39

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com



Runtime Environment

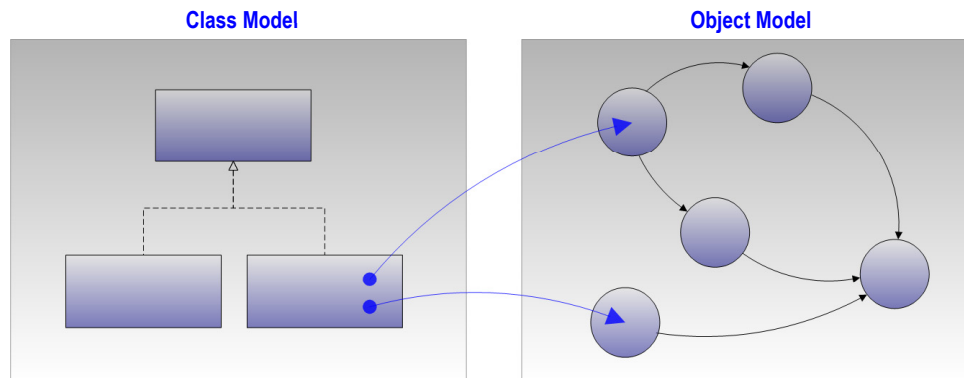
Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Runtime Model

- Transition from a class system to an object system
- An object model provides a unique and specific instantiation of the class specification

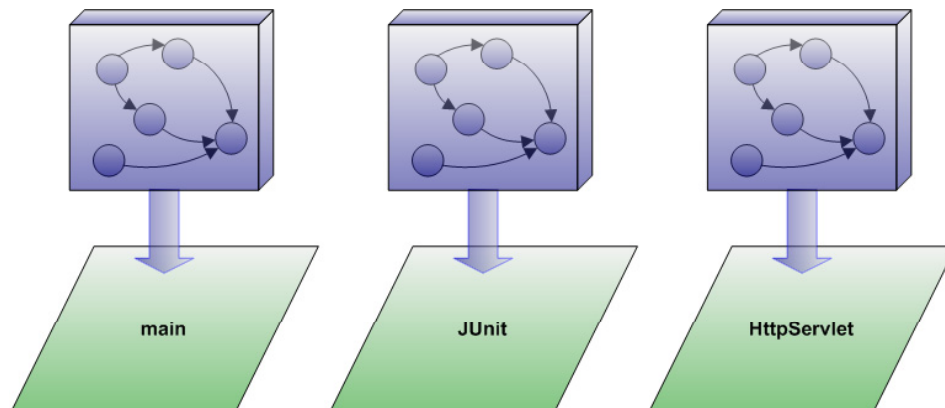


41

Java EE training: <http://courses.coreservlets.com>

Runtime Context

- Multiple deployment contexts
- Complex object models should be portable
- Object models should be configurable to support changes between environments



42

Java EE training: <http://courses.coreservlets.com>

Runtime Example

```
public class Main {  
  
    public static void main(String[] args) {  
  
        BookReader client = new BookReader();  
  
        List<Book> books = client.read();  
  
        System.out.printf("Client read: %s books%n",  
                           books.size());  
  
    }  
}
```

Standard output

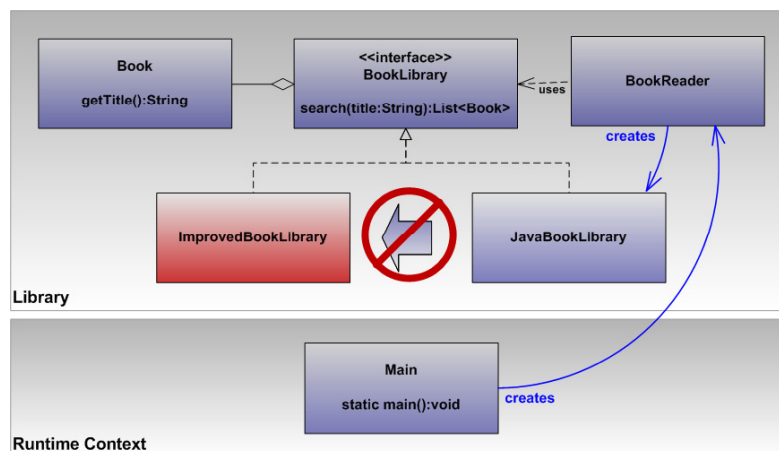
```
Reading: Core Servlets and JavaServer Pages  
Reading: More Servlets and JavaServer Pages  
Client read: 2 books
```

43

Java EE training: <http://courses.coreservlets.com>

Model Analysis

- **Hard-coded implementation choices**
 - Object model **cannot** be reconfigured
 - Future implementation types **cannot** be used without modifying and rebuilding BookReader

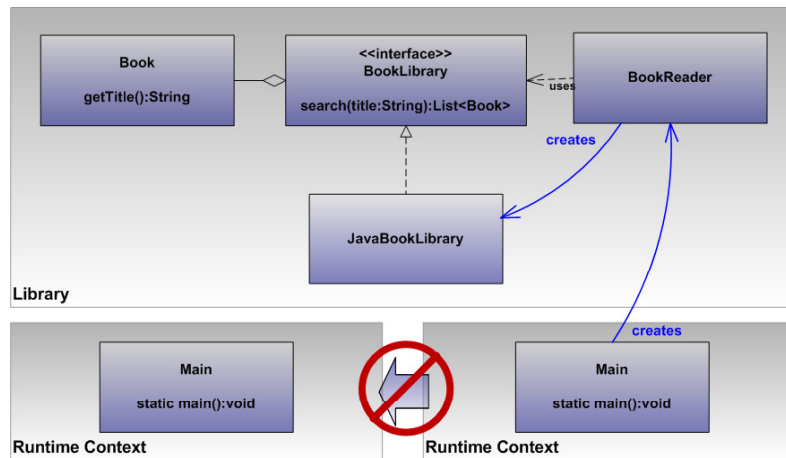


44

Java EE training: <http://courses.coreservlets.com>

Model Analysis

- **Hard-coded model configuration**
 - Object model is **not** portable



45

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com



Manual (Non-Spring) Dependency Injection

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Dependency Injection

- **Delivers object dependencies at runtime**
- **Encourages the separation of responsibilities**
- **When used with the interface pattern**
 - Isolates implementations from clients
 - Minimizes the impact on clients when implementations evolve

Dependency Injection Process

- **Design depending types to receive implementations**
 - Allow dependencies to be supplied using property setters or constructors
 - Other dependency injection methods are also available, such as field injection, but requires third-party or Java reflection support
- **Avoid constructing objects from the client to fulfill dependencies**
 - For example, do not use the **new** operator to manage services

Dependency Injection Candidate

```
public class BookReader {  
  
    private BookLibrary bookLibrary;  
  
    public BookReader() {  
        this.bookLibrary = new JavaBookLibrary();  
    }  
  
    public List read() {  
        List<Book> books = bookLibrary.search("Java");  
        for(Book book : books){  
            System.out.printf("Reading: %s%n", book);  
        }  
        return books;  
    }  
}
```

Creates dependency

49

Java EE training: <http://courses.coreservlets.com>

Dependency Injection Example

```
public class BookReader {  
  
    private BookLibrary bookLibrary;  
  
    public BookReader(BookLibrary bookLibrary) {  
        this.bookLibrary = bookLibrary;  
    }  
  
    public List read() {  
        List<Book> books = bookLibrary.search("Java");  
        for(Book book : books){  
            System.out.printf("Reading: %s%n", book);  
        }  
        return books;  
    }  
}
```

Dependency injection interface

Interface type
NOT the implementation type

50

Java EE training: <http://courses.coreservlets.com>

Runtime Example

```
public class Main {  
    public static void main(String[] args) {  
        BookLibrary service = new JavaBookLibrary();  
        BookReader client = new BookReader(service);  
        List<Book>books = client.read();  
        System.out.printf("Client read: %s books%n",  
            books.size());  
    }  
}
```

Dependency creation moved out of BookReader

Dependency injection

Standard output

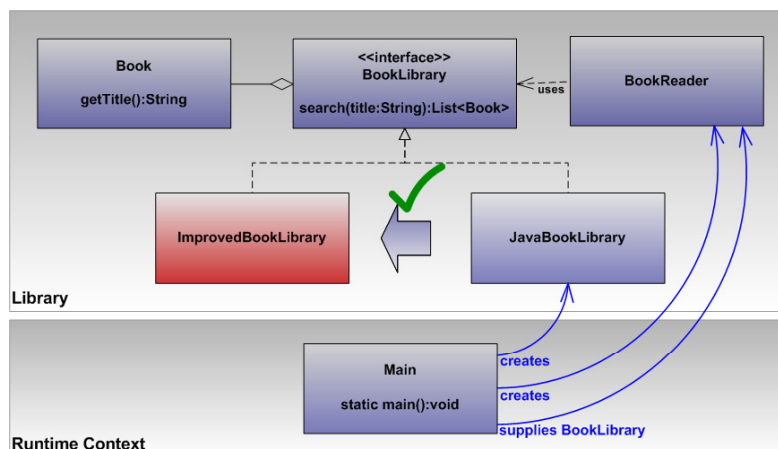
```
Reading: Core Servlets and JavaServer Pages  
Reading: More Servlets and JavaServer Pages  
Client read: 2 books
```

51

Java EE training: <http://courses.coreservlets.com>

Model Analysis

- **Dynamic implementation choices**
 - Object model **can** be reconfigured
 - Future implementation types **can** be used without modifying and rebuilding BookReader

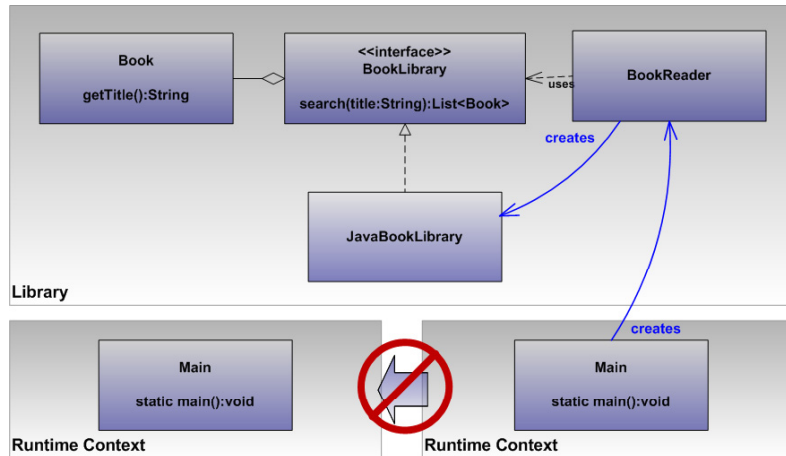


52

Java EE training: <http://courses.coreservlets.com>

Model Analysis

- **Hard-coded model configuration**
 - Object model is **not** portable



53

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com



Manual (Non-Spring) Inversion of Control

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Introduction

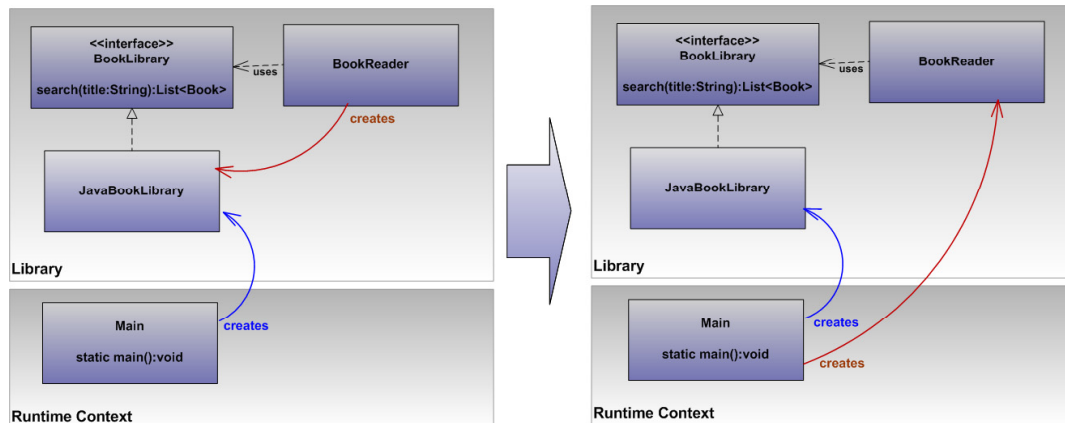
- **Inversion of Control**
- **Separate program control responsibilities**
 - Object instantiation
 - Dependency injection
- **Dependency injection is a type of IoC**

55

Java EE training: <http://courses.coreservlets.com>

Inversion of Control Example

- **Previously used IoC**
 - Dependency injection example demonstrated inversion of control
 - Moved **JavaBookLibrary** selection and instantiation out of **BookReader** and into **Main**



56

Java EE training: <http://courses.coreservlets.com>

IoC Framework

- **Service provider or plugin framework**
 - Interface
 - Providers
 - **Registration system**
 - **Access API**
 - Joshua Bloch from *Effective Java*
- **Process**
 - Framework uses supplied APIs
 - Framework handles creation
 - Framework handles dependency injection
 - Runtime context uses framework

57

Java EE training: <http://courses.coreservlets.com>

IoC Framework Example

```
import coreservlets.BookReader;
import coreservlets.JavaBookLibrary;

public class ServiceProviderFramework {

    private BookReader bookReader;

    public ServiceProviderFramework() {
        this.bookReader =
            new BookReader(new JavaBookLibrary());
    }

    public BookReader getBookReaderInstance() {
        return this.bookReader;
    }
}
```

Implicit registration

Access API

58

Java EE training: <http://courses.coreservlets.com>

IoC Framework Example

```
public class Main {  
    public static void main(String[] args) {  
  
        ServiceProviderFramework framework =  
            new ServiceProviderFramework();  
  
        BookReader client = framework.getBookReaderInstance();  
  
        List books = client.read();  
  
        System.out.printf("Client read: %s books%n",  
                           books.size());  
    }  
}
```

Framework instantiation

Access API

Standard output

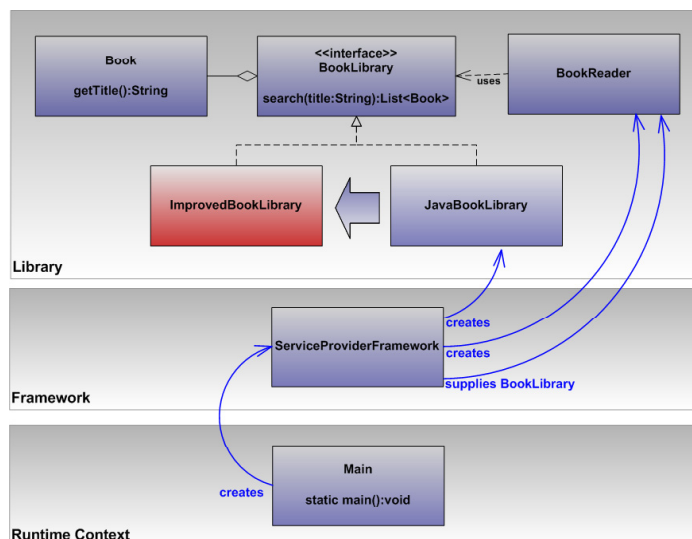
```
Reading: Core Servlets and JavaServer Pages  
Reading: More Servlets and JavaServer Pages  
Client read: 2 books
```

59

Java EE training: <http://courses.coreservlets.com>

Model Analysis

- Dynamic implementation choices
- Portable model configuration



60

Java EE training: <http://courses.coreservlets.com>



Wrapup

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Summary

- **Develop POJOs**
 - Avoid framework dependencies
 - Capture business logic
 - Avoid implementation commitments by using inversion of control and dependency injection patterns
- **Create a new XML file, `applicationContext.xml`, based on `spring-beans.xsd`**
 - Place `applicationContext.xml` in the classpath
- **Register POJOs**
 - Declare POJOs using XML `bean` elements
 - Use bean attributes `id` and `class` for specifying the name and type, respectively

Summary (Continued)

- **Instantiate a Spring IoC container**
 - Use the BeanFactory implementation `ClassPathXmlApplicationContext` for integration with configuration files located in the classpath
 - See: `org.springframework.context.support.ClassPathXmlApplicationContext`
- **Access the Spring IoC container**
 - Retrieve objects from the Spring IoC container using the bean accessor methods
 - For example, `BeanFactory#getBean(...):Object`
 - Specify the object name for the method parameter
 - `beanFactory.getBean("bookLibrary");`

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com



Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.