SSH (Struts , Spring and Hibernate)

Ronghong Li, You Li, Miao Xu, Paul Filipow, Chao Rui

Introduction

The Model-View-Controller Pattern

- Model-View-Controller (MVC) is a design pattern
  - The model
    - Represents the underlying data and business logic in one place
    - Contains no information about the user interface
  - The view
    - The user interface – things the user can see and respond to
    - Represent a window into the model – there can be many of these
  - The controller
    - Connects the model and the view
    - Used to communicate between the model and view
- A fourth layer – persistence – is often added to the pattern

MVC – Benefits

- Promotes code reuse
  - The purpose of the model is to provide business logic and data access in one place
  - This logic can be reused in many applications at the same time without the need for any extra coding
- Reduces development time
  - The model, view, and controller can be developed in parallel
- More maintainable
  - The view can be changed without affecting the model
    - A Web page view can be changed to display a chart instead of a table with no change to the model
  - The model can be changed without affecting the view
    - For example, the way in which an insurance premium is calculated may change, but the interface to the business method remains the same
  - Data can be moved without affecting the view or model
    - The layering concept allows for more flexibility

JavaBean

- Has a set of public features
  - Properties, methods, events
- Requires a zero-argument constructor
- Implements **Serializable** interface
- Has public accessible getter/setter/is method
- A black-box independent unit

Spring Framework

What is Spring Framework

- The **Spring Framework** is an open source application framework for the Java platform.
- Spring is a lightweight framework that is based upon the theoretics of IoC and AOP. It uses POJO to build the J2EE application.

What is Spring Framework

- The core features of the Spring Framework can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the Spring Framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBean (EJB) model.

History

- The first version was written by Rod Johnson who released the framework with the publication of his book *Expert One-on-One J2EE Design and Development* in October 2002. The framework was first released under the Apache 2.0 license in June 2003. The first milestone release, 1.0, was released in March 2004, with further milestone releases in September 2004 and March 2005. The Spring 1.2.6 framework won a Jolt productivity award and a JAX Innovation Award in 2006.[2][3] Spring 2.0 was released in October 2006, and Spring 2.5 in November 2007. In December 2009 version 3.0 GA was released. The current version is 3.0.5.[4]

Why we want Spring

- Easy to test, easy to change, easy to maintain.
- Some pages about advantages:
- http://onjava.com/pub/a/onjava/2005/05/11/spring.html
- http://www.techfaq360.com/tutorial/spring/advantagespring.jsp
- http://www.wrox.com/WileyCDA/Section/Why-Use-the-Spring-Framework-.id-130098.html

How Spring helps us

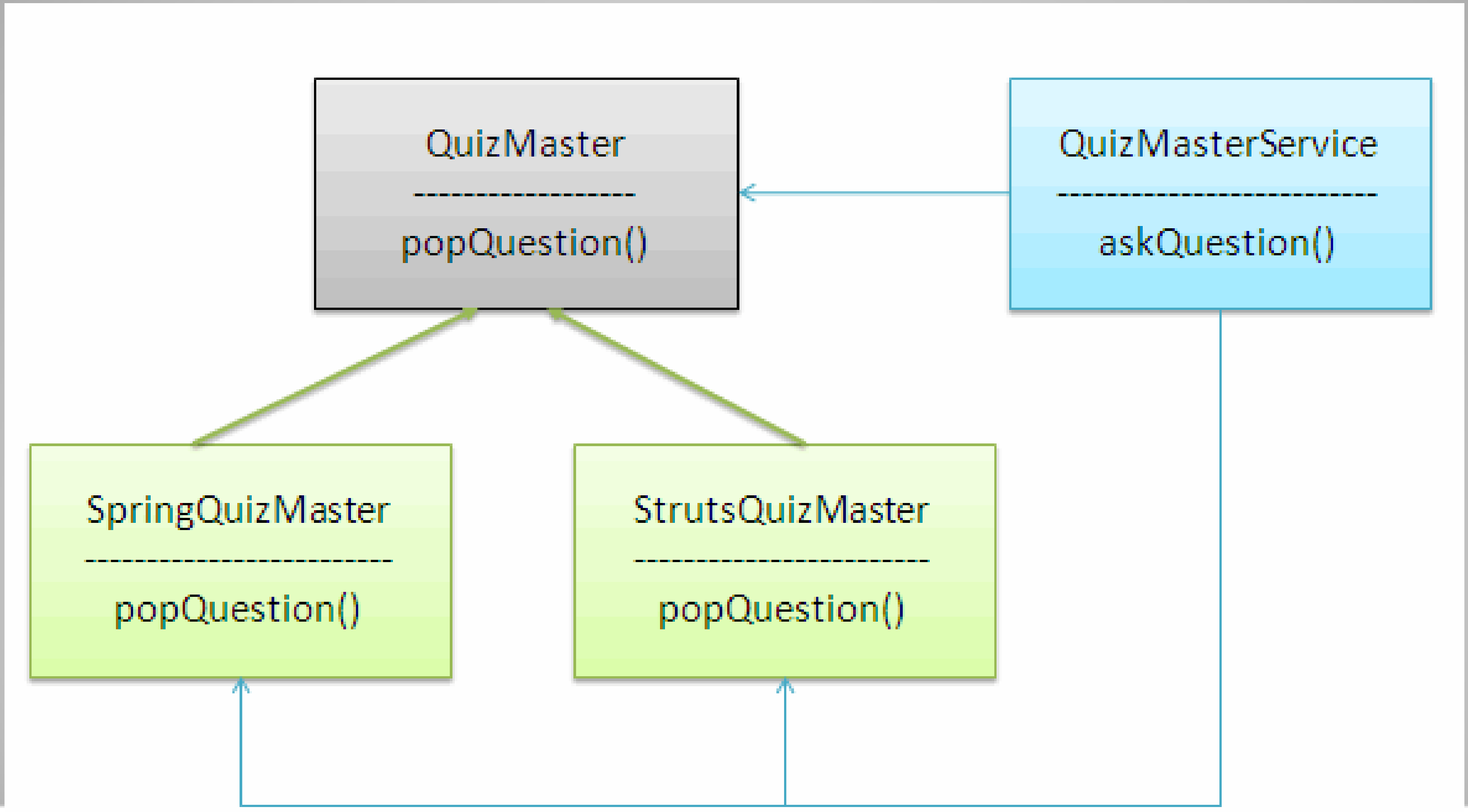
- Inversion of Control (IoC)
- Aspect-oriented Programming (AOP)
- Some other features.

How Spring helps us

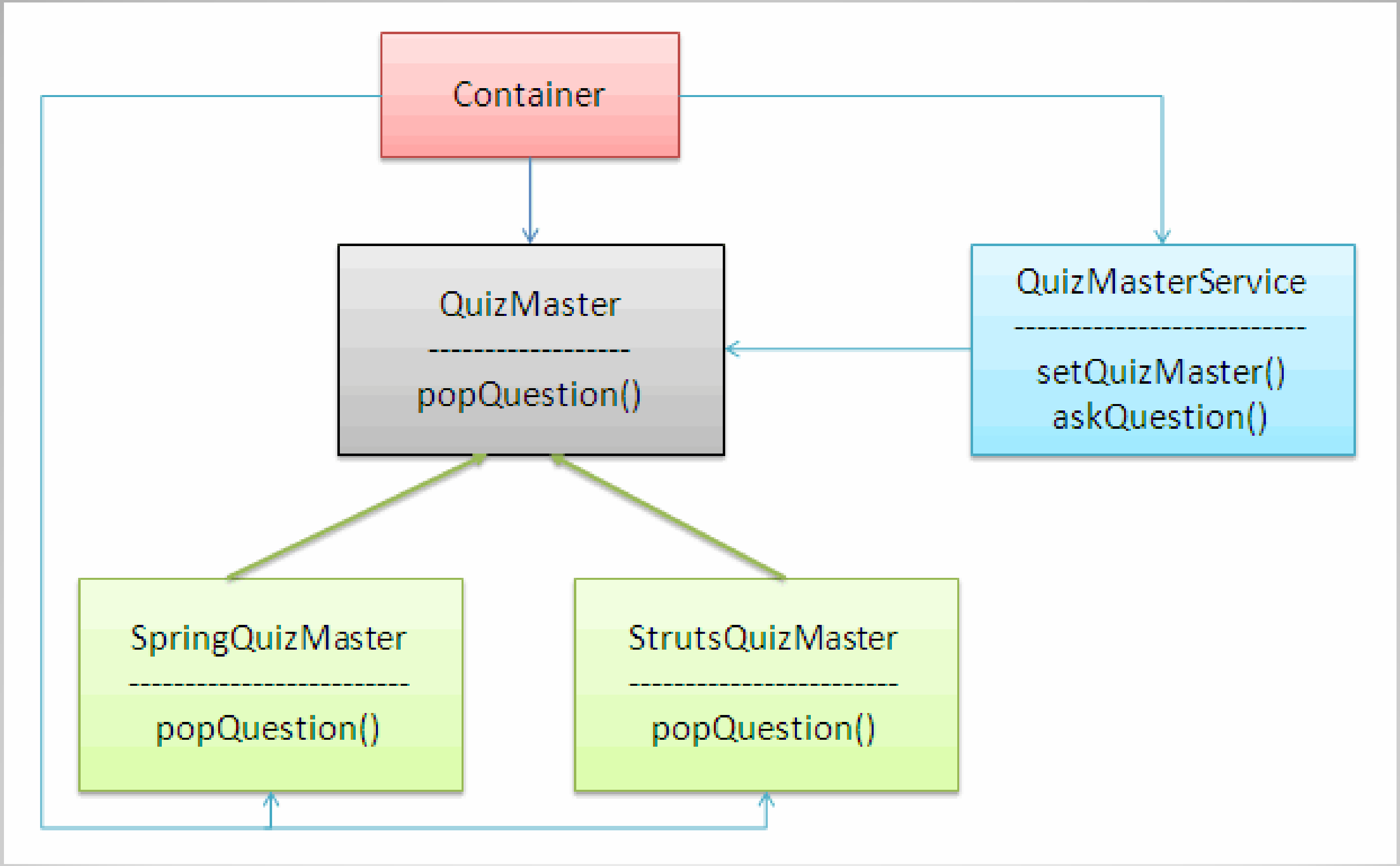
- Inversion of Control
- In computer programming, **Inversion of control** (**IoC**) is an abstract principle describing an aspect of some software architecture designs in which the flow of control of a system is inverted in comparison to procedural programming.
- In traditional programming the flow of the business logic is controlled by a central piece of code, which calls reusable subroutines that perform specific functions. Using Inversion of Control this "central control" design principle is abandoned. The caller's code deals with the program's execution order, but the business knowledge is encapsulated by the called subroutines.
- Spring achieve this with dependency injection.

How Spring helps us

- Example about Dependency Injection:



How Spring helps us



http://www.vaannila.com/spring/spring-ioc-1.html

How Spring helps us

- Aspect-Oriented Programming
- Aspect-oriented programming entails breaking down program logic into distinct parts (so-called *concerns*, cohesive areas of functionality). All programming paradigms support some level of grouping and encapsulation of concerns into separate, independent entities by providing abstractions (e.g., procedures, modules, classes, methods) that can be used for implementing, abstracting and composing these concerns. But some concerns defy these forms of implementation and are called *crosscutting concerns* because they "cut across" multiple abstractions in a program.

How Spring helps us

- The Spring Framework has its own AOP framework which modularizes cross-cutting concerns in aspects. The motivation for creating a separate AOP framework comes from the belief that it would be possible to provide basic AOP features without too much complexity in either design, implementation, or configuration. The SpAOP framework also takes full advantage of the Spring Container.

How Spring helps us

- AOP is used in the Spring Framework to:
- provide declarative enterprise services, especially as a replacement for EJB declarative services. The most important such service is *declarative transaction management*.
- allow users to implement custom aspects, complementing their use of OOP with AOP.
- More info: http://static.springsource.org/spring/docs/2.5.x/reference/aop.html

Other Modules

- Data access framework
- Transaction management framework
- Model-view-controller framework
- Remote access framework
- Convention-Over-Configuration Rapid Application Development
- Batch Framework

Some cons of Spring

- Confusing for starter
- May make code harder to read, maybe even harder to manage.
- Of course it is resource consuming.

Struts

Overview

- Apache Struts is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture.
- - Wikipedia

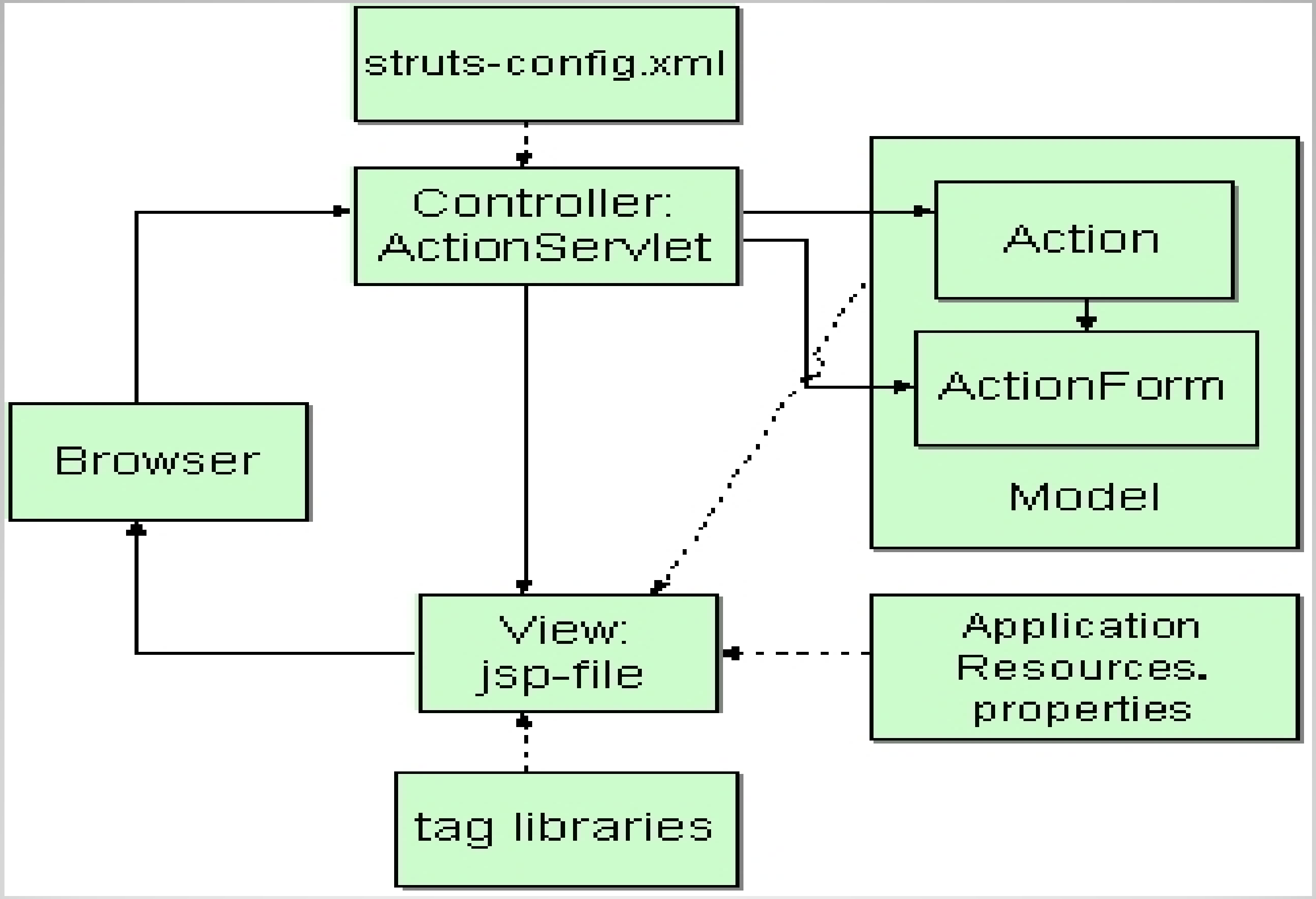
Overview

- A flexible control layer based on standard technologies like Java Servlets, JavaBeans, ResourceBundles, and XML, as well as various Apache Commons packages, like BeanUtils and Chain of Responsibility. The framework helps you create an extensible development environment for your application, based on published standards and proven design patterns. - Apache

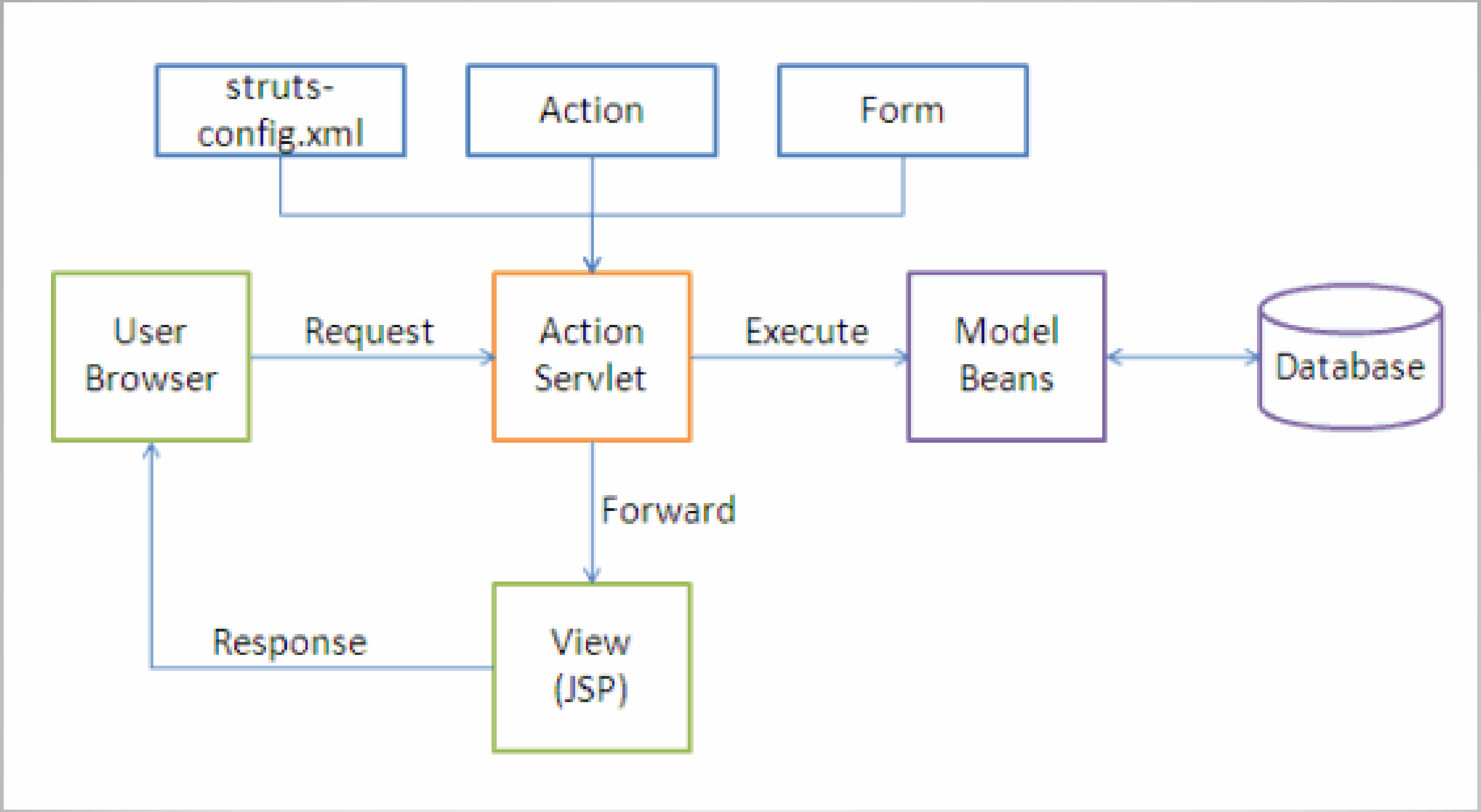
Struts' role

- The goal of Struts is to separate the model (application logic that interacts with a database) from the view (HTML pages presented to the client) and the controller (instance that passes information between view and model).
- Struts provides the controller and facilitates the writing of templates for the view or presentation layer.
- Creating a central configuration file struts-config.xml that binds together model, view and controller.

Struts' role

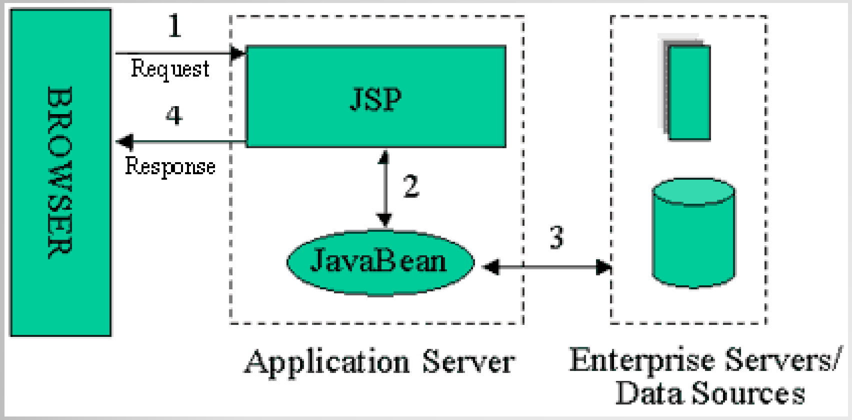

struts-config.xml
Controller: ActionServlet
Action
ActionForm
Model
Browser
View: jsp-file
Application Resources. properties
tag libraries

How does Struts work?



Struts or not?

- If you need to write a very simple application, with a handful of pages, then you might consider a "Model 1" solution that uses only server pages.



- But, if you are writing a more complicated application, with dozens of pages, that need to be maintained over time, then Struts can help.

Hibernate

Hibernate

- A framework for mapping an object-oriented domain model to a traditional relational database.
- Uses a technique called “object-relational mapping”
- Maps Java classes to relational databases tables and vice-versa

Hibernate

- Object-oriented programming even when saving/retrieving data from database.
- No need for the database queries.
- Offers full data independence:
- No need to change database if application changes,
- No need to change application if database changes.

Object-Relational Mapping

- An object database that can be read and written by incompatible OO languages
- Compromise between RDMS and OODBMS:
- Higher abstraction than traditional relational databases
- More data independence than fully object-oriented database systems

Hibernate Operation

- Hibernate converts Java objects to SQL datatypes for use in a relational DBMS
- Completely hides SQL and database issues from developers

Hibernate Operation

- Instead of this (JDBC):

```
String createTableToffees = "CREATE TABLE TOFFEES    " +   "(T_NAME
    VARCHAR(32), SUP_ID INTEGER, PRICE FLOAT, " +  "SALES INTEGER,
    TOTAL INTEGER)";
Statement stmt = m_con.createStatement();
stmt.executeUpdate(createTableToffees);
stmt.executeUpdate("INSERT INTO TOFFEES " + "VALUES ('Quadbury', 101,
    7.99, 0, 0)");
```

- We just need this (Hibernate):

```
Toffe t = new Toffee('Quadbury', 101, 7.99, 0, 0);
Session.save(t);
```

- Hibernate takes care of all the tables, queries, and datatypes!

Hibernate Operation

- Hibernate uses either XML or Java annotations to handle object-SQL conversion.
- jBoss provides tools to create these easily
- Hibernate can be used with any RDMS (MySQL, MSSQL, Oracle, etc.)

MyEclipse

MyEclipse

- Created and maintained by Genuitec.
- A commercially available Java EE and Ajax IDE Development.
- Build upon the Eclipse platform.
- Integrate proprietary and open source solution.
- Has abundant resource and support.
- Fully support HTML, Struts, JSF, CSS, JavaScript SQP and Hibernate.

My Eclipse (version)

- Two main version.
- Blue edition (professional).
- Spring edition(Standard).
- Need Eclipse before version 6.0.
- Current version 8.6

Advantage

- abundant of resource.
- Fully support HTML, Struts, JSF, CSS, JavaScript SQP and Hibernate.
- Easy for new programmer.
- More efficacy.

Disadvantages

- Not Free $158.95 for blue version.
- Waste more system resource.
- Slow down the system.
- Makes programmer rely more on Myeclipse. They may forget some basic knowledge.

Questions?