15.6 Comparing the Three Data-Sharing Approaches

In the MVC approach, a servlet responds to the initial request. The servlet invokes code that fetches or creates the business data, places that data in beans, stores the beans, and forwards the request to a JSP page to present the results. But, *where* does the servlet store the beans?

The most common answer is, in the request object. That is the only location to which the JSP page has sole access. However, you sometimes want to keep the results around for the same client (session-based sharing) or store Web-application-wide data (application-based sharing).

This section gives a brief example of each of these approaches.

Request-Based Sharing

In this example, our goal is to display a random number to the user. Each request should result in a new number, so request-based sharing is appropriate.

To implement this behavior, we need a bean to store numbers (Listing 15.8), a servlet to populate the bean with a random value (Listing 15.9), and a JSP page to display the results (Listing 15.10, Figure 15-6).

Listing 15.8 NumberBean.java

```
package coreservlets;

public class NumberBean {
  private double num = 0;

  public NumberBean(double number) {
    setNumber(number);
  }

  public double getNumber() {
    return(num);
  }

  public void setNumber(double number) {
    num = number;
  }
}
```

Listing 15.9 RandomNumberServlet.java

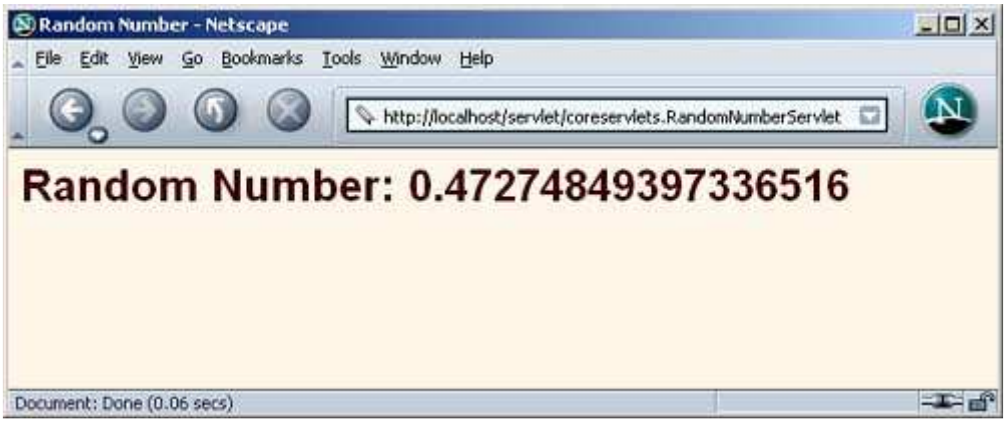
```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** Servlet that generates a random number, stores it in a bean,
 *  and forwards to JSP page to display it.
 */
public class RandomNumberServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    NumberBean bean = new NumberBean(Math.random());
```

```
    request.setAttribute("randomNum", bean);
    String address = "/WEB-INF/mvc-sharing/RandomNum.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

Listing 15.10 RandomNum.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Random Number</TITLE>
<LINK REL=STYLESHEET
      HREF="/bank-support/JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<jsp:useBean id="randomNum" type="coreservlets.NumberBean"
             scope="request" />
<H2>Random Number:
<jsp:getProperty name="randomNum" property="number" />
</H2>
</BODY></HTML>
```

Figure 15-6. Result of `RandomNumberServlet`.



Session-Based Sharing

In this example, our goal is to display users' first and last names. If the users fail to tell us their name, we want to use whatever name they gave us previously. If the users do not explicitly specify a name and no previous name is found, a warning should be displayed. Data is stored for each client, so session-based sharing is appropriate.

To implement this behavior, we need a bean to store names (Listing 15.11), a servlet to retrieve the bean from the session and populate it with first and last names (Listing 15.12), and a JSP page to display the results (Listing 15.13, Figures 15-7 and 15-8).

Listing 15.11 NameBean.java

```
package coreservlets;

public class NameBean {
  private String firstName = "Missing first name";
  private String lastName = "Missing last name";
```

```
  public NameBean() {}

  public NameBean(String firstName, String lastName) {
    setFirstName(firstName);
    setLastName(lastName);
  }

  public String getFirstName() {
    return(firstName);
  }

  public void setFirstName(String newFirstName) {
    firstName = newFirstName;
  }

  public String getLastName() {
    return(lastName);
  }

  public void setLastName(String newLastName) {
    lastName = newLastName;
  }
}
```

Listing 15.12 RegistrationServlet.java

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** Reads firstName and lastName request parameters and forwards
 *  to JSP page to display them. Uses session-based bean sharing
 *  to remember previous values.
 */

public class RegistrationServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession();
    NameBean nameBean =
      (NameBean)session.getAttribute("nameBean");
    if (nameBean == null) {
      nameBean = new NameBean();
      session.setAttribute("nameBean", nameBean);
    }
    String firstName = request.getParameter("firstName");
    if ((firstName != null) && (!firstName.trim().equals(""))) {
      nameBean.setFirstName(firstName);
    }
    String lastName = request.getParameter("lastName");
    if ((lastName != null) && (!lastName.trim().equals(""))) {
      nameBean.setLastName(lastName);
    }
    String address = "/WEB-INF/mvc-sharing/ShowName.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

Listing 15.13 ShowName.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Thanks for Registering</TITLE>
<LINK REL=STYLESHEET
      HREF="/bank-support/JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Thanks for Registering</H1>
<jsp:useBean id="nameBean" type="coreservlets.NameBean"
             scope="session" />
<H2>First Name:
<jsp:getProperty name="nameBean" property="firstName" /></H2>
<H2>Last Name:
<jsp:getProperty name="nameBean" property="lastName" /></H2>
</BODY></HTML>
```

Figure 15-7. Result of `RegistrationServlet` when one parameter is missing and no session data is found.

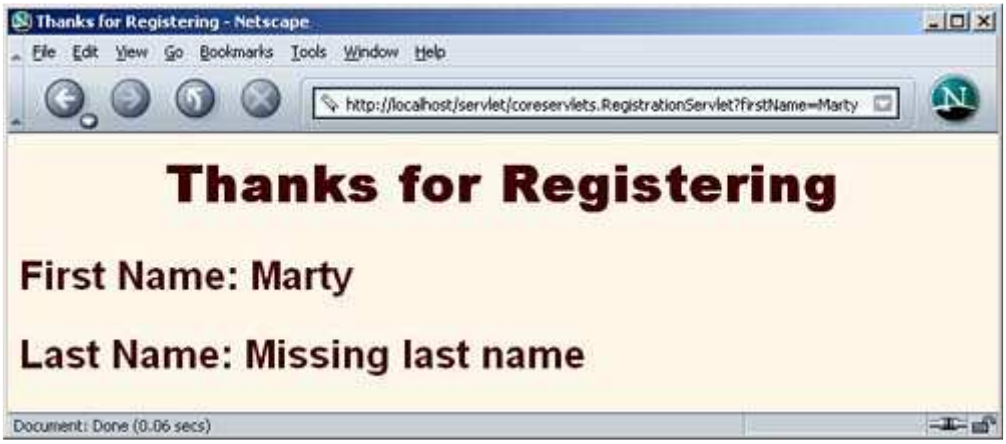
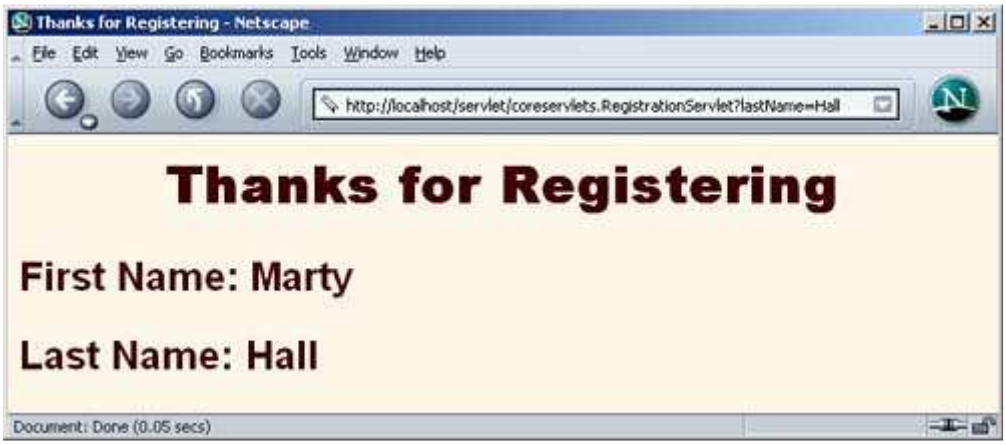


Figure 15-8. Result of `RegistrationServlet` when one parameter is missing and session data is found.



Application-Based Sharing

In this example, our goal is to display a prime number of a specified length. If the user fails to tell us the desired length, we want to use whatever prime number we most recently computed

for *any* user. Data is shared among multiple clients, so application-based sharing is appropriate.

To implement this behavior, we need a bean to store prime numbers (Listing 15.14, which uses the `Primes` class presented earlier in Section 7.4), a servlet to populate the bean and store it in the `ServletContext` (Listing 15.15), and a JSP page to display the results (Listing 15.16, Figures 15-9 and 15-10).

Listing 15.14 PrimeBean.java

```
package coreservlets;

import java.math.BigInteger;

public class PrimeBean {
  private BigInteger prime;

  public PrimeBean(String lengthString) {
    int length = 150;
    try {
      length = Integer.parseInt(lengthString);
    } catch(NumberFormatException nfe) {}
    setPrime(Primes.nextPrime(Primes.random(length)));
  }

  public BigInteger getPrime() {
    return(prime);
  }

  public void setPrime(BigInteger newPrime) {
    prime = newPrime;
  }
}
```

Listing 15.15 PrimeServlet.java

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class PrimeServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    String length = request.getParameter("primeLength");
    ServletContext context = getServletContext();
    synchronized(this) {
      if ((context.getAttribute("primeBean") == null) ||
          (length != null)) {
        PrimeBean primeBean = new PrimeBean(length);
        context.setAttribute("primeBean", primeBean);
      }
      String address = "/WEB-INF/mvc-sharing/ShowPrime.jsp";
      RequestDispatcher dispatcher =
        request.getRequestDispatcher(address);
      dispatcher.forward(request, response);
    }
  }
}
```

Listing 15.16 ShowPrime.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>A Prime Number</TITLE>
<LINK REL=STYLESHEET
      HREF="/bank-support/JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>A Prime Number</H1>
<jsp:useBean id="primeBean" type="coreservlets.PrimeBean"
             scope="application" />
<jsp:getProperty name="primeBean" property="prime" />
</BODY></HTML>
```

Figure 15-9. Result of `PrimeServlet` when an explicit prime size is given: a new prime of that size is computed.

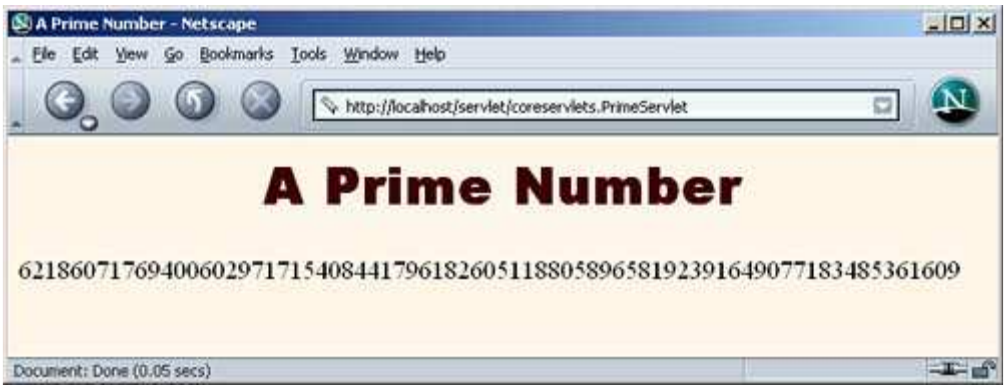


Figure 15-10. Result of `PrimeServlet` when no explicit prime size is given: the previous number is shown and no new prime is computed.