Developing a NodeJS mock back-end for a mobile application

Contents

Developing a NodeJS mock back-end for a mobile application

Introduction

This developerWorks article demonstrates how to integrate a mobile application with a mock back-end. The following documentation highlights the benefits of developing a mock back-end in parallel to the creation of your own custom REST APIs by using the Cúram REST API infrastructure.

The Adult Social Care mock back-end, which is attached to this developerWorks article, provides a worked example of mocking up REST APIs. These are based on requirements that are specified for the Adult Social Care mobile application. The example is designed to help you with creating and testing your own REST APIs. Availing of earlier testing opportunities helps ensure that the design meets the needs of each of the REST APIs.

The Adult Social Care mobile application was developed against a mock REST back-end that is implemented in NodeJS, Express, and JSON.

Prerequisite skills

It is assumed that you have a working knowledge of JavaScript and are familiar with the JSON data format. It is also assumed that you are familiar with the IBM Cúram REST API accelerator.

Related information:

IBM Cúram REST API accelerator for IBM Cúram Social Program Management Platform

About the authors

James Shannon is a Software Engineer for the Partner Enablement team in IBM Cúram. James is working in IBM Cúram for 7 months.

Caroline O'Connor is a Senior Software Engineer for the Partner Enablement team in IBM Cúram. Caroline is working in IBM Cúram for 10 years. Caroline recently worked for 6 years on a customer site in New Zealand as a Senior Technical Consultant.

Architecture overview

The following architecture diagram displays the connection between the Adult Social Care mobile application and the Adult Social Care mock back-end.

Adult Social Care mobile application

The mobile application is built by using IBM MobileFirst and is simulated through the IBM MobileFirst development environment. The IBM MobileFirst platform was used to build the mobile application. It is a suite of products that enables partners and customers to efficiently build and deliver mobile applications for their enterprise application.

From using the IBM MobileFirst simulator in your development environment, the mobile application is rendered with HTML and CSS. When the user selects an action in the mobile application, it triggers business logic that is written as a JavaScript application.

The JavaScript application sends a REST API request by using an IBM MobileFirst HTTP adapter with a data payload written in JSON.

Adult Social Care mock back-end

On the mock back-end side, a NodeJS server intercepts the call. The Node Server processes the REST API request with a JavaScript application.

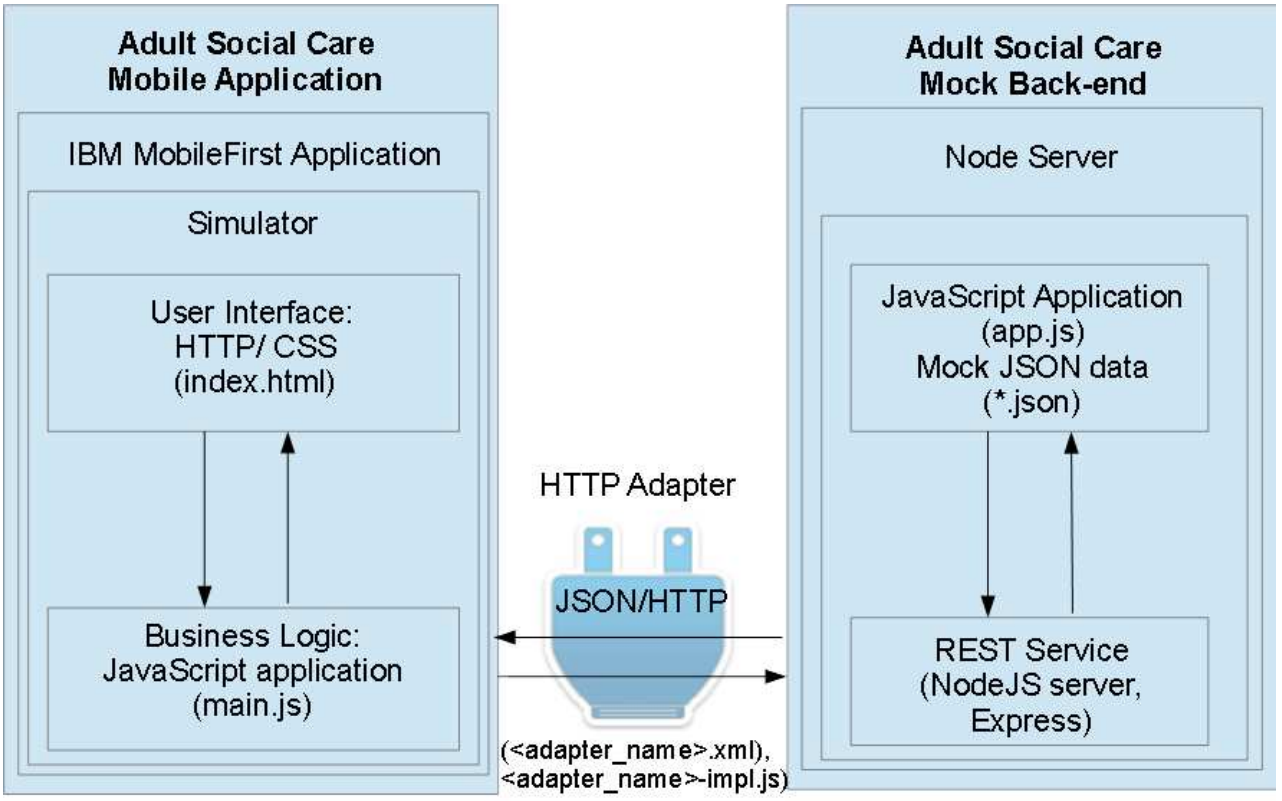


Figure 1. Architecture diagram

Installing the mock back-end and the mobile application

The following includes steps for installing the Adult Social Care mock back-end, the Adult Social Care mobile application, as well as instructions for integrating both.

Installing NodeJS and the mock back-end

Complete the following tasks to install and run a NodeJS server for the sample mock REST back-end.

Installing NodeJS and Express

Complete the following steps to install NodeJS and the Express library.

About this task

NodeJS is a JavaScript library to enable you to easily and quickly build efficient, scalable network applications.

Express is a minimal, robust NodeJS web application framework, which provides a set of features for web and mobile applications. Express also provides a range of HTTP methods which allow you to quickly build and test REST APIs.

Procedure

1. To download NodeJS, go to the following link Download Node.js.
2. From the download NodeJS window, select the installer for your platform.

3. Run the installer and follow the Node.js setup wizard steps. In this documentation, the variable %NODEJS_DIR% denotes the directory of the NodeJS installation on your system.
4. Open a command prompt and change to the %NODEJS_DIR% directory.
5. Run the following commands to install Express:

```
npm install -g express
npm install  -g express -generator
```

Installing the Adult Social Care mock back-end

The Adult Social Care mock back-end is available for download on developerWorks accompanied with this developerWorks documentation. Complete the following steps to install the Adult Social Care mock back-end.

Procedure

1. From the developerWorks article, download the Adult Social Care mock back-end.
2. Extract the Adult Social Care mock back-end to the NODEJS_DIR directory. A folder that is named ASCMockApp is added to the directory. This directory includes the following artifacts:

Table 1. ASC Mock Application artifacts

File name	Description
ASCMockApp/app.js	A JavaScript file, which initializes the mock data and defines the mock REST APIs.
ASCMockApp/ASCMockData/ asccase.json	A mock Adult Social Care case JSON object.
ASCMockApp/ASCMockData/ asccert.json	A mock Adult Social Care practitioner certificate and diagnosis JSON object.
ASCMockApp/ASCMockData/ changeReasonCT.json	A list of change reason types are specified in this code table specific to Adult Social Care.
ASCMockApp/ASCMockData/ diagnosesCT.json	A list of diagnosis types are specified in this code table specific to Adult Social Care.
ASCMockApp/ASCMockData/ persons.json	Two mock person details are specified in this JSON object.
ASCMockApp/ASCMockData/ pracCertTypeCT.json	A list of practitioner certificate types are specified in this code table specific to Adult Social Care.

Installing the mobile application

For installing the Adult Social Care mobile application, you are required to install the associated third-party libraries.

Installing IBM MobileFirst development environment

Complete the following steps for installing the Eclipse Marketplace widget and the IBM MobileFirst Studio in Eclipse.

Before you begin

For installing IBM MobileFirst, you must have an existing instance of the Eclipse development environment. You need to install a compatible version of Eclipse such as Juno SR2, Kepler SR1, Kepler SR2, Luna SR1, or Luna SR2.

Procedure

Installing the Eclipse Marketplace widget
Some Eclipse versions come without the Eclipse marketplace widget, which is required to install IBM MobileFirst. Follow these steps to add the Eclipse Marketplace widget only if it does not exist already in your Eclipse.

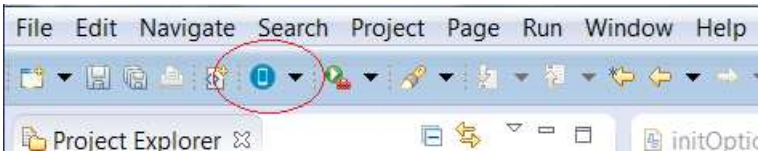
1. From Eclipse, click **Help** > **Install New Software**.
2. From the **Work with** input field, select the drop-down and click **All Available Sites**.
3. In filter text input field type **Marketplace**.
4. Under General Purpose Tools, select **Marketplace Client**.
5. Click **Next** and finish the installation.

Installing MobileFirst Studio in Eclipse

6. From Eclipse, click **Help** > **Eclipse Marketplace**.
7. From the search bar, enter **Mobile First Platform** and click **Go**.
8. Scroll down to IBM MobileFirst Platform Studio and click **Install**.
9. Select **IBM MobileFirst Platform Studio** if it is not already selected and click **Confirm**.
10. **Accept** the terms and conditions, click **Finish**.
11. If you get a security warning click **OK**.
12. Restart Eclipse when prompted.

What to do next

You are able to access the MobileFirst menu from Eclipse. It is denoted by a blue mobile phone icon.



Installing the Adult Social Care mobile accelerator

The Adult Social Care mobile application accelerator is available for download from the Cúram Accelerators community on developerWorks. Complete the following steps to get the mobile application up and running with IBM MobileFirst.

Procedure

1. Download the Adult Social Care mobile accelerator from the Cúram Accelerators community.
2. Extract the Adult Social Care mobile accelerator at the root of the Cúram installation `%CURAM_DIR%`. By default, `C:\IBM\Curam\Development`. A folder that is named `Mobile` is added to `%CURAM_DIR%` directory. In this documentation, the variable `%MOBILE%` denotes the directory of the mobile application.
3. From Eclipse, click **File** > **Import** > **General** > **Existing Projects into Workspace**.
4. From the Select root directory option, click **Browse** and navigate to the `%CURAM_DIR%\Mobile` directory.
5. From the list of available applications, select **ASCMobile** and click **Finish**.

Installing third-party libraries

The following third-party libraries are required to be installed for the Adult Social Care mobile application user interface.

About this task

For this task, the path *%MOBILE_APPLICATION%* refers to the location: `%CURAM_DIR%\Mobile\ASCMobileApplication\ASCMobile\apps\ASCMobileApplication\common\`.

Table 2. Third-part libraries. The following table lists all third-party libraries that are required and their version numbers.

Library name	Version no.
jQuery UI	1.11.0
Foundation	5.0

Procedure

Installing jQuery UI

1. From a web browser, go to the following URL: JQuery-UI.
2. Select jQuery UI 1.11.0 (concatenated JS and CSS files).
3. Save and extract the file.
4. From the extracted folder, copy the `jquery-ui.js` file to the `%MOBILE_APPLICATION%\js` directory.
5. From the extracted folder, copy the `jquery-ui.css` file to the `%MOBILE_APPLICATION%\css` directory.

Installing Foundation

6. From a web browser, go to the following URL: Foundation 5.
7. From the Download Foundation 5 window, select **Download Everything**.
8. From the Opening foundation-5.5.2.zip window, select **Save File** and click **OK**.
9. Extract the file to a temporary folder.
10. From the extracted folder, copy the content from the `css` directory to the `%MOBILE_APPLICATION%\css\lib` directory.
11. From the extracted folder, copy the content from the `js` directory to the `%MOBILE_APPLICATION%\js\lib` directory.

Integrating the mock back-end with the mobile application

Complete the following tasks to integrate the mock back-end with the mobile application. For the purposes of this task, the mock application data represents a registered person on the Cúram system.

Updating the IBM MobileFirst adapter settings

Complete the following tasks to update the mobile application to point at your mock back-end instead of Cúram.

Procedure

1. From Eclipse, navigate to the `%MOBILE%\adapters\ClientAdapter` directory open the `ClientAdapter.xml` file.
2. Inside the file, change the *domain* and *port* values to be as follows:

Table 3. Adapter connection settings

port	domain	Description
3000	*localhost*	Using the default NodeJS port *3000* on the *localhost* domain.

Results

The IBM MobileFirst adapter now points to your mock back-end server instead of Cúram for the person search features of the application. The search results in the application display your mock person when you enter their name as the search criteria.

Starting the NodeJS server

Complete the following steps to start the NodeJS server that uses the Express web framework library.

Procedure

1. Open a command prompt and change to the `%NODEJS_DIR%` directory.
2. Run the following command:
   ```
   node /ASCMockApp/app.js
   ```

Results

Your server is now running on the port you specified in the command. You can view your server through a web browser at the following URL: https://localhost:3000/.

Deploying the mobile application

From your IBM MobileFirst development environment, you need to deploy the mobile application and adapters before you can preview the mobile application that uses the simulator. Complete the following steps to view the Adult Social Care mobile application.

Procedure

1. From Eclipse, right-click on the `ASCMobile` directory and click **Open MobileFirst Console**. You can ignore the message in red in the Eclipse console, which includes the host name and other console warnings. The Eclipse console displays (in green) Application 'ASCMobileApplication' with all environments build finished. Once the deployment is complete, the IBM MobileFirst Platform Operations Console is opened in an internet browser where you can see the mobile application and the adapters deployed.
2. From the `Home>ASCMobile` directory, select **Applications** to view the mobile application deployed.
3. From the `Home>ASCMobile>Applications` directory, select **ASCMobileApplication**.
4. From the Common Resources section, select **Preview** to view the mobile application through the IBM MobileFirst web simulation environment.

Performing the Adult Social Care mobile application caseworker scenario

Detailed steps are provided for you to walk through the sample mobile application, and experience the full end-to-end business work flow.

Before you begin

You need to install the Adult Social Care mobile application accelerator and start your NodeJS server.

Searching for a client

Complete the following steps to find a client who is already registered as a person on the system.

Procedure

1. From the mobile application landing page, enter the client's full name in the search for a client field and click **Search**.

Table 4. Search for a client value

Name	Value
Full Name	James Smith

2. From the Search Results page, if the search criteria is met, click on the person link to open the Client Profile page.

Editing a practitioner certificate

Complete the following steps to edit the practitioner certificate evidence record.

Procedure

1. From the Client Profile window, click **Edit**.
2. From the Edit Practitioner Certificate window, modify the following details and click **Save**.

Name	Description	Value
Change Reason	The reason why the evidence needs to be changed.	Reported by Client.
Specified Certification To Date	The last day that is covered by the practitioner certification.	Current date + 8 weeks.
Comments	Comments that are captured by the caseworker.	Extended the certification by 6 months due to new diagnosis.

Creating a diagnosis

Complete the following steps to create a diagnosis that is associated with the practitioner certificate.

Procedure

1. From the Client Profile window, click **New Diagnosis**.
2. From the New Diagnosis window, enter the following values and click **Save**.

Name	Description	Value
Received Date	The date that the practitioner certificate evidence was received by the organization.	Current date
Examination Date	The date on which the medical examination took place.	Current date
Diagnosis Type	The code for the physical disorder or illness the claimant is suffering.	Cardiovascular disease

Developing a mock back-end

Developing a mock back-end allows your mobile application to be tested against a representation of your final application back-end. Testing as early as possible during the development lifecycle helps ensure that the design meets the needs of each of the REST APIs.

High-level concepts for creating a mock back-end

The following section describes the high-level concepts to be considered when you are creating a mock application back-end.

Ensuring the mock back-end is an accurate reflection of the final application data is the most important aspect to consider during its development. There a number of advantages and disadvantages to this approach, which are as follows:

Advantages

1. It provides a much earlier opportunity to test your application against a data set than if you were to wait for the full back-end to be developed.
2. A mock back-end is as lightweight and simple as possible.
3. Complex logic should not be replicated such as calculations, rules that are handled by Cúram.

4. If the retrieval of data from the mock back-end is implemented as a set of simple reads and writes, you can rapidly test edge case data.
5. For implementing your REST APIs in Cúram, keeping your mock back-end up to date means you can quickly test your mobile application against data changes.

Disadvantages

1. It requires extra effort to implement.
2. The mock back-end might need frequent updates if the content or structure changes regularly.

There are several factors to keep in mind while you are creating your mock back-end as follows:

- Try to replicate the structure and content of the JSON representation as accurately as possible before creating your REST APIs in Cúram.
- To capitalize on earlier testing opportunities, it is favorable to include edge case data instances to test that the application logic is performed as expected.

Creating and running a NodeJS server

Complete the following steps to create your own sample NodeJS server that uses the Express web framework library.

Procedure

1. From the `%NODEJS_DIR%` NodeJS installation directory, create a new directory called `MyMockApp`.
2. From the `%NODEJS_DIR%/MyMockApp` directory, create a new JavaScript file and name it `app.js`.
3. Open this new file in a text editor.
4. Add the following lines of code to include the Express library and to tell NodeJS to use it:

```
//Import Express.
var  express  = require ( 'express' );
//Create JavaScript variable to represent express functionality on the server.
var app = express();
//Tell the application that you wish to use the JSON format on your server.
app.use(express.json());
//start server (default port 3000).
app.listen(process.env.PORT || 3000);
```

5. Open a command prompt and change to the `%NODEJS_DIR%` directory.
6. Run the following command:

```
node /MyMockApp/app.js
```

Results

Your server is now running on the port you specified in the command. You can view your server through a web browser at the following URL: https://localhost:3000/.

Creating mock data

Complete the following tasks to set up a mock data object representation of a piece of application data. For the purposes of this document, you must create a piece of data, which represents an Adult Social Care case.

Creating a mock application data

The JSON format is used to represent Adult Social Care cases and their associated data, this task involves creating a mock representation of an Adult Social Care case.

Procedure

1. From the `%NODEJS_DIR%\MyMockApp` directory, create a new folder called `mockdata`.

2. From the%NODEJS_DIR%\MyMockApp\mockdata directory, create a new JavaScript file called mockperson.json.
3. Open %NODEJS_DIR%\MyMockApp\mockperson.json inside a text editor.
4. Enter the following JSON object that represents an Adult Social Care case:

```
{
    "concern_role_id": "101",
    "full_name": "James Smith",
 "email" : "James@curamsoftware.com",
    "dateOfBirth": "1964-09-26",
    "primaryAddress" : {
      "displayText" : "1074, Park Terrace\nFairfield\nMidway, Utah, 12345\nUnited States",
      "addressType" : {
        "tableName" : "AddressType",
        "value" : "AT1",
        "description" : "Private"
      }
    },
    "primaryPhoneNumber" : {
      "countryCode" : "1",
      "areaCode" : "555",
      "number" : "3477455",
      "extension" : ""
    },
    "versionNo": "101"
}
```

5. Save your file.

Importing the mock data

Complete the following steps to import your JSON data into your server by using the NodeJS file system.

Procedure

1. Open your %NODEJS_DIR%\MyMockApp\app.js file in a text editor.
2. Add the following lines of code to your file to import your JSON object into a JavaScript variable:

```
//Tell your server that you are using the file system module.
var fs = require("fs");
//Import the JSON object from your file into the server.
var person = JSON.parse(fs.readFileSync("./mockdata/person.json", "utf8"));
```

Results

Your JSON object is parsed into the JavaScript variable *person* when your server first runs and can now be used in any server code you write.

Implementing your back-end and mapping a data set

Complete the following tasks to implement a set of functions on your server for mapping incoming requests to their related data sets. Processing the types of requests that are received and generating a response to the request are also covered.

Implementing a response

Complete the following steps to implement a response function for a GET request inside your NodeJS server.

About this task

This task covers the implementation of a response function for a GET request from a specific URL.

Procedure

1. Open your %NODEJS_DIR%\MyMockApp\app.js file in a text editor.
2. Add the following code to the file to allow for processing of an incoming GET request:

```
//function to return person(s) by name.
var getPersonsByName = function(req, res) {
 var responseList = [];
 var key = req.param("full_name");
 //for every person in the person database with a name matching the "full_name" query parameter
 //add it to a response list to be returned by the request.
 for (concern_role_id in personArray) {
  if(personArray[concern_role_id]!==undefined && personArray[concern_role_id].full_name === key)
   responseList.push(personArray[concern_role_id]);
 }
 //convert the response list to JSON.
 if(responseList.length>0){
  res.statusCode = 200;
  res.json(responseList);
 }
 else{
  res.statusCode = 404;
  return res.send('Not found');
 }
}

//Tell the server to process all GET requests from the URL: "/Rest/v1/persons/:concern_role_id" by running the function
app.get('/Rest/v1/persons/', getPersonsByName);
```

Mapping a data set

Complete the following steps to implement a data mapper structure in JavaScript. The data mapper allows the data on the server to be divided up and associated with the URL of a received request.

Procedure

1. Open your %NODEJS_DIR%\MyMockApp\app.js file inside a text editor.
2. Add the following code to implement a basic data mapper for your imported JSON person data:

```
//set URL paths for displaying clients and filtering by concern_role_id
var personMap = {
 "/Rest/v1/persons/": personDatabase
}

//set concern_role_id as primary key for the person database
var personKeyMap = {
 "/Rest/v1/persons/": "concern_role_id"
}
```

3. Save your file.

Testing the mock back-end

You can test the example person mock back-end against the Adult Social Care mobile application.

Procedure

Follow the existing instructions for integrating your mock back-end with the mobile application.

1. Update the IBM MobileFirst adapter settings.
2. Start the NodeJS server. For starting the NodeJS server, you need to run the following command that points to the new application you created.

```
node /MyMockApp/app.js
```

3. Deploy the mobile application.
4. Perform the searching for a client scenario.

Related tasks:

“Integrating the mock back-end with the mobile application” on page 5
Complete the following tasks to integrate the mock back-end with the mobile application. For the purposes of this task, the mock application data represents a registered person on the Cúram system.

Notices

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources.

IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing

application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at " Copyright and trademark information " at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.