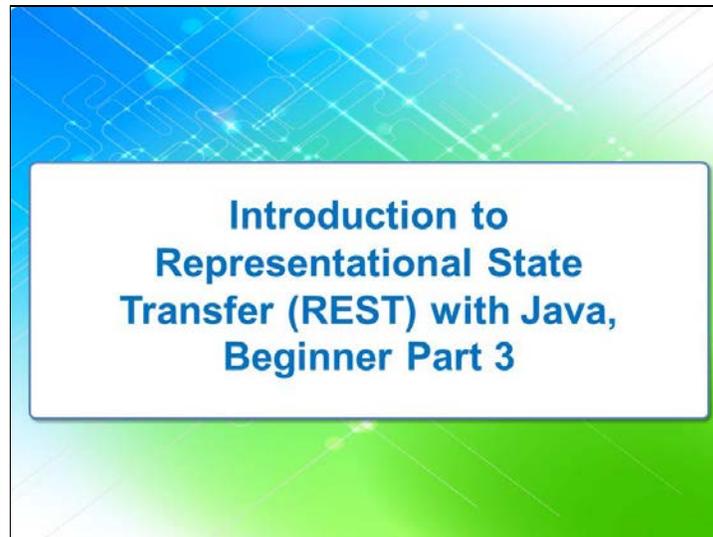


Slide 1

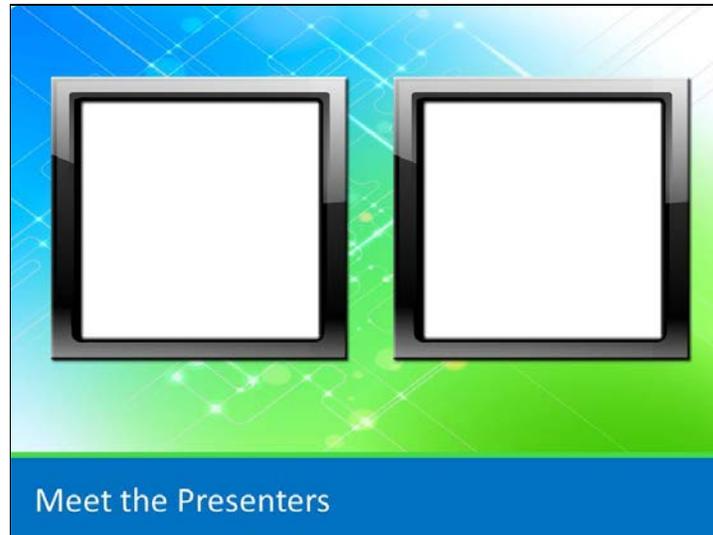


PRESENTER 1 SAY:

On behalf of IT Workforce Development, welcome to today's session of *Introduction to Representational State Transfer (REST) with Java, Beginner Part 3*. This is the last course in our three-part series on REST and RESTful web services.

In the second course, you learned about REST standardized actions, outputs, and design guidelines. In today's course, we'll talk about how to overcome problems when using REST, security issues, and best practices. We'll also demonstrate how to test and debug RESTful web services through live demonstrations and assign you a take-home exercise, so you can tap into what you've learned throughout these courses.

Slide 2



PRESENTER 1 SAY:

We'd like to take a moment to introduce ourselves. I'm PRESENTER 1.

PRESENTER 2 SAY:

And I'm PRESENTER 2.

We're looking forward to providing you with information on today's topic. But before we get started, let's talk about a few Lync items.

Slide 3

Lync Information

Chat/Questions: Use the Chat Window
Dial-in: Use the VANTS line in the invitation
Participation Credit: Submit TMS self-certification
Group Participation: Email vaitwd@va.gov
Handouts: Download via the paperclip icon



The image shows a screenshot of a Lync chat window. The window title is 'Parker, Peter (USA)'. The chat area shows 'Parker, Peter - Available'. Below the chat area is a toolbar with icons for 'IM', 'Call', 'Video', and 'Share'. A large paperclip icon is overlaid on the right side of the toolbar, indicating the location of the handout download icon.

PRESENTER 1 SAY:

We want this to be an engaging session, so we'll be asking you questions. Please use the Chat window to respond. You can also use the Chat window if you have questions for us. We'll answer your questions during Q&A at the end of the session.

We want to make sure you get credit for attending today's session. We will provide you with instructions on how to complete the self-certification at the end of our presentation so be sure to stick around.

If you are attending as a group, please email the VA ITWD mailbox. We'll be able to ensure you all receive completion credit.

PRESENTER 2 SAY:

To access handouts, select the paperclip icon in the Lync toolbar above the list of participants. Then, select the right arrow next to the file in the Attachments pop-up menu. From there you can select Open or Save As. If you chose Save As, you can select where you want to save the download.

Slide 4



PRESENTER 1 SAY:

In today's course, we're going to talk about common problems in REST and tools you can use to diagnose and fix them. We'll show you a demo of how to debug REST APIs using a web debugging proxy server application called Fiddler.

Next, we'll move on and talk about how REST uses protocols to secure REST APIs.

Finally, we'll discuss best practices in REST and answer any questions you have. As you start developing REST APIs, these best practices will be helpful in creating products that other developers will be able to use. We'll be providing you the opportunity to apply what you learned in this webinar in your homework exercise. For today's assignment, you'll debug a REST web service using Fiddler and a web browser. As we go through our topics, please reflect on how you can incorporate them into a web application you're building. We've got a lot to cover, so let's get started!

Slide 5



PRESENTER 2 SAY:

Let's start today by talking about common problems encountered in REST application development and some possible solutions.

Slide 6



PRESENTER 2 SAY:

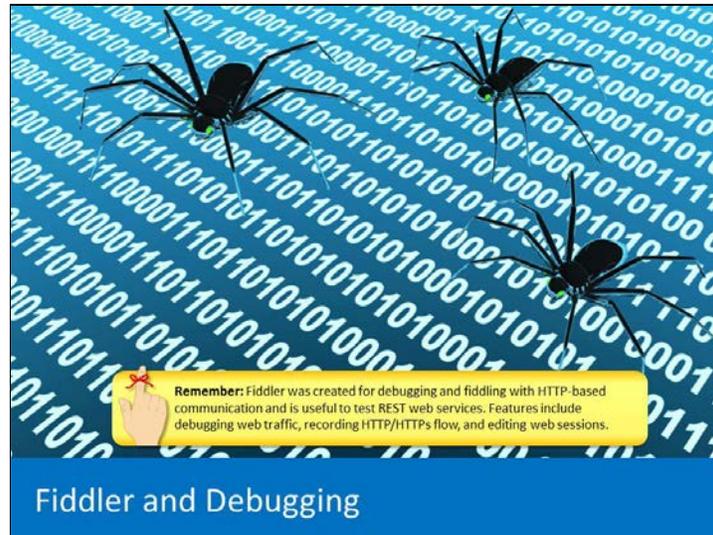
First, let's spend some time talking about two common problems you could run into while creating RESTful web services. These include accidentally introducing software bugs into your REST APIs and making invalid assumptions in your documentation.

First, let's talk about introducing a software bug into your REST API. Don't worry, even the best programmers find that they've sometimes introduced a bug into their code. Sometimes you'll get error messages, but, often, you'll notice something is wrong when your request times out because the server hasn't responded. In order to avoid this problem, use a Java debugger frequently. This will debug any errors in the code for the application you're creating. Also, in XML and JSON, parsing and building serialized data will be automatic. We're going to talk more about debugging in just a moment.

Another problem developers encounter is making invalid assumptions in their documentation. This can happen in situations where documentation doesn't match implementation, or when developers don't understand how to document a process, or when they don't authenticate a resource before retrieving it. Understanding what is expected from the server versus what actually happened is important to debugging these types of errors. Using tools to test and properly log errors helps decode what happened with the client and the server.

To find, isolate, and begin to fix these problems, we need to do some debugging. One of them is Fiddler, which is a third-party HTTP client debugging tool.

Slide 7



PRESENTER 1 SAY:

When testing and debugging a REST API, our first bit of advice is to breathe, keep calm, and love the REST API!

Our second, more practical bit of advice is to use a tool called Fiddler to test and debug REST APIs. This tool was created for fiddling with HTTP based communication and is useful to test REST web services. It's approved for general use in VA's Technical Reference Manual.

Fiddler's features include debugging web traffic, recording HTTP and HTTPS flow, and allowing developers to edit web sessions.

First, Fiddler debugs web traffic by decrypting data and decompressing web sessions from any client, browser, or system. Fiddler helps decrypt and translate HTTP traffic to display requests that developers wouldn't be able to interpret. In order to do this, Fiddler acts as a middle man to translate HTTP traffic. If a web server is configured to use HTTP compression to decrease message sizes, Fiddler can decompress these files. Fiddler also helps modify HTTP requests that could be unreadable by other users. Decrypting data, decompressing web sessions, and modifying HTTP requests result in a huge decrease in the number of bytes and files that are transmitted by the server.

Second, Fiddler records HTTP traffic between a machine and the Internet. It logs HTTP traffic between a browser and a web server that is being used to test REST APIs. It is crucial to inspect HTTP traffic to distinguish the error codes that are being received.

Last, Fiddler allows its users to edit their web sessions by setting breakpoints whenever a certain U-R-L is hit. The HTTP process is paused and Fiddler developers can manually alter

their request and response. Rewriting web traffic—data sent and received by visitors to a website—is useful for security and functionality testing because all code paths can be exhausted.

Slide 8



PRESENTER 2 SAY:

There are a few more debugging tools that are worth noting. TCPMon is a service from Apache that allows users to monitor messages passed in Transmission Control Protocol, or TCP, based conversations. TCP is a set of rules used along with Internet Protocol, or I-P. This service is based on a user interface, or U-I, and works on Java supported platforms. Using the TCPMon will allow you to see more raw TCP data.

Another tool worth mentioning is RESTClient. This tool tests REST APIs to get a request and response. It also tests a variety of HTTP communications. To download this tool, visit www.wiztools.org. Please note, we've provided this link and several other resources for you in the handout. Also, RESTClient is available for Macs for a \$15.00 fee.

Other open source tools can be found in Firefox, which is one of VA's approved browsers. It has additional add-ons that can be used by selecting the developer button.

Another tool is Wireshark, which is a network analysis tool to capture traffic in real time and display it in a human-readable format. It monitors traffic in and out of a single endpoint to dig deep into network traffic.

PRESENTER 1 SAY:

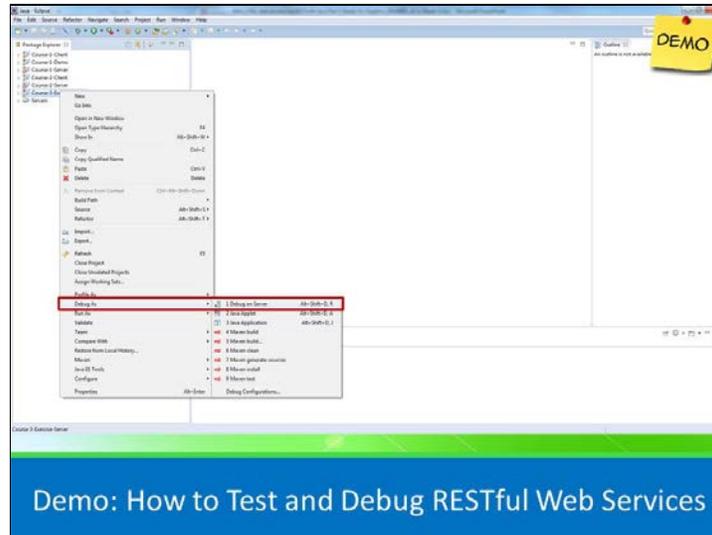
If you're trying to debug your REST API and don't have a Windows desktop computer available, there are additional resources that can be used. On Android devices, LogCat can be used to debug REST APIs. It provides a way to collect and view system debug outputs. iPhones and iPads use Safari, which has a native web application tool built in. This tool isn't automatically turned on, so you'll need to activate it. Once it's activated, you can use it along with Modus Create products, a third-party vendor, to debug your REST APIs.

PRESENTER 2 SAY:

We wanted to mention these tools just to tell you what was out there. For the purposes of this course, we're going to use Fiddler because it's a VA-approved debugging application. Some of these other tools may be approved for general use or approved with constraints. Check VA's Technical Reference Manual for more information.

Let's move on and show you how to test and debug REST APIs using Fiddler.

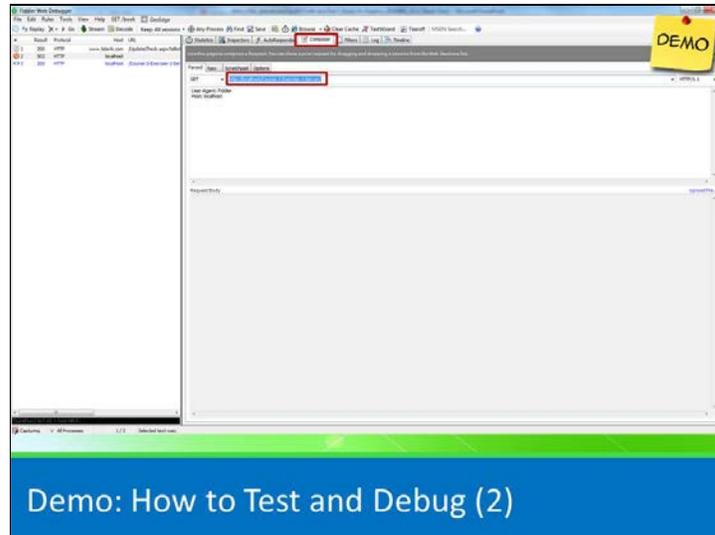
Slide 9



PRESENTER 1 SAY:

We start our server with our course three demo, which also happens to be your homework for today that you'll do on your own after this course. First, we'll open up Eclipse, and select the course three folder for our demo, and then right-click to debug on server. This runs the code.

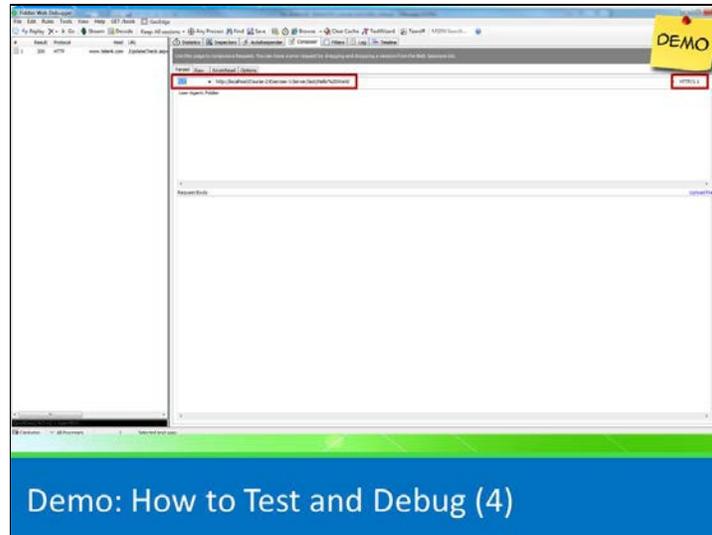
Slide 10



PRESENTER 1 SAY:

Next, we'll move into Fiddler. For this demonstration, we'll access the API in Fiddler. For now, we're going to focus on the Composer tab. You can compose an HTTP message there. Copy the URL where the API lives from Eclipse into Fiddler.

Slide 12

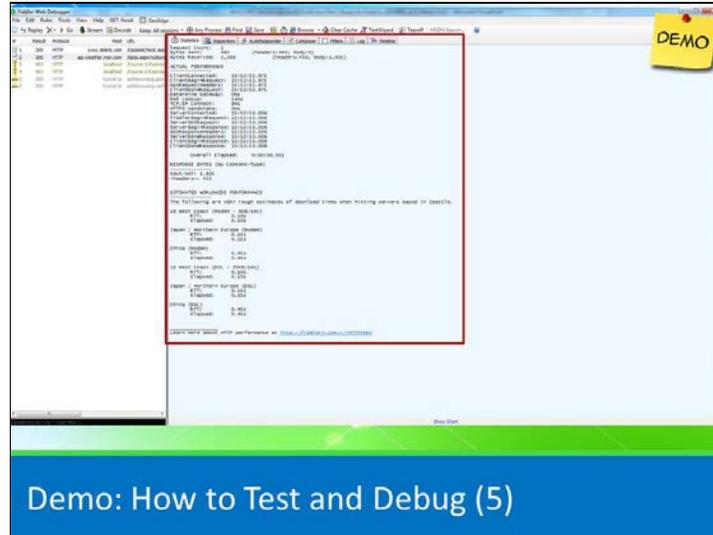


PRESENTER 1 SAY:

On the left of the composer tab, you can see the standard actions buttons. There's a drop-down menu, and there are a lot of these, but right now, we're going to focus on the PUT action. On the right, you can see the protocol version. For now, we're going to select "HTTP 1.1." In the URL, I'm going to add "slash test" after value, then add "slash Hello percent twenty World."

Note that all URLs must be escaped. That's what the "percent twenty" here represents. Escaping a URL is taking characters that are invalid for HTTP resources and converting them into an encoded format that is a valid URL. In this example, "percent twenty" represents a space. If you want to know more about escaping, you can do an internet search for URL encoding. Now, we execute to test.

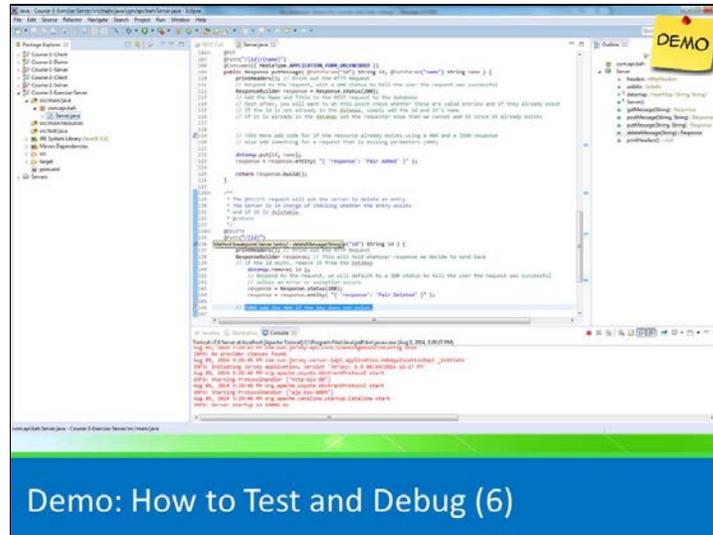
Slide 13



PRESENTER 1 SAY:

If we go into the local host item in the HTTP request log, it takes us into the statistics tab, which shows the response. Here is your status code with the text of the response. It looks like everything ran properly.

Slide 14



PRESENTER 1 SAY:

But what if there is a problem? What if you get an error message? Debugging is a way to pinpoint a problem or find the origin of an error message. One of the best things you can do to debug an issue is to add breakpoints.

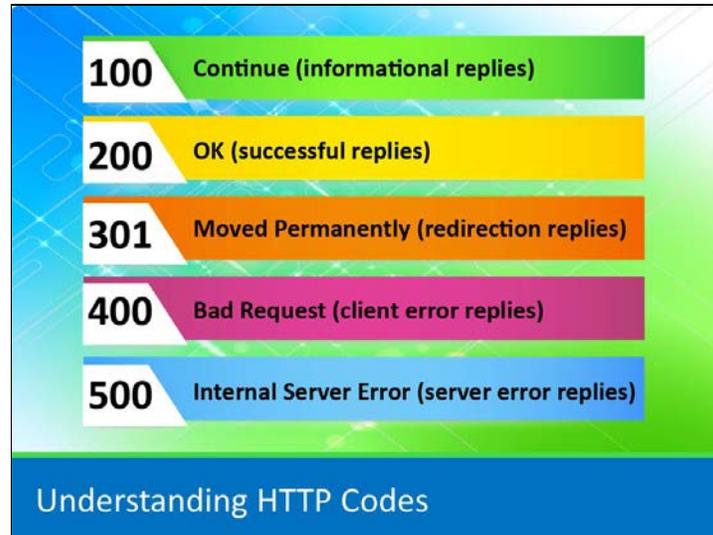
A breakpoint is a manually added break in the execution of the application. It's a way for developers to figure out where an issue is because the debugger will stop execution at the point in the code when it gets to the breakpoint. It stops the application execution and you can see exactly what's happening.

To add a breakpoint, you click into the function where you want to put a breakpoint. See the blue bar to the left of the code? Wherever you want your breakpoint at the end of the line, double-click on that blue bar and you will see a little blue circle. You can also go to the Run menu and select "toggle breakpoint," but you must be on the line where you want it. Control-Shift-B in your line of code is another shortcut for that. Let's add a breakpoint and run the code.

You can see that my application ran, but it stopped execution when it hit our breakpoint.

So now you know the basics of how to test and debug your application using Fiddler. Now, let's talk about HTTP response codes and how they help you figure out what's wrong.

Slide 15



PRESENTER 2 SAY:

In course two, we mentioned that REST outputs are HTTP responses communicating results. HTTP response codes also help troubleshoot problems with REST APIs. These codes help determine what the error is. Many of the errors you'll see when debugging are errors that occur when the REST API reacts in an unexpected way. Decoding a server status is the first step in detecting what the error could be.

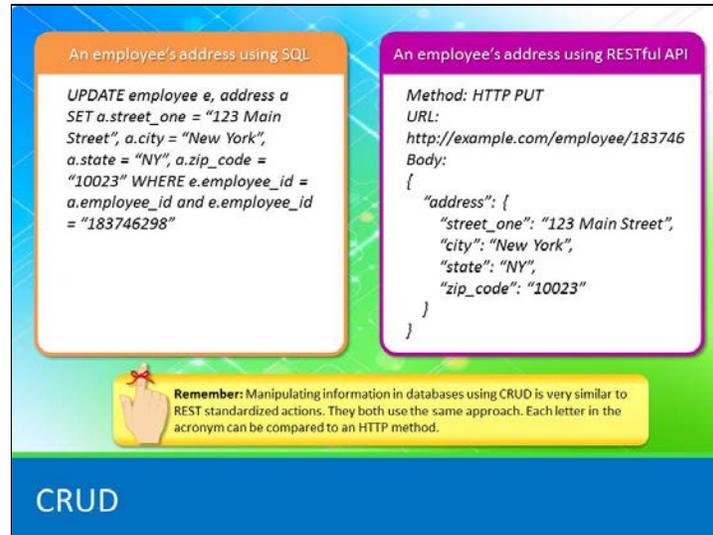
HTTP responses fall into five categories and are distinguished by the first digit in each response. An informational reply starts with a one, a successful reply begins with a two, a redirection reply starts with a three, a client error reply begins with a four, and a server error reply starts with a five. Categorizing response codes allows for a uniform response.

When implementing HTTP status codes for REST APIs, it's best to not implement all of them. There are over 50 codes and most developers haven't memorized all of them. Rather, focus on response codes that cover successful and unsuccessful responses. For example, use codes that will send a successful response, which starts with a two; codes that would send a client error, which starts with a four; and codes that would send a server error, which starts with a five.

The benefit of implementing HTTP response codes in your REST APIs is that they'll work even if no error codes are implemented. However, having no HTTP response codes will result in a poorly created REST API. If something goes wrong, a developer won't know where to start when trying to fix the problems.

Let's move on to discuss persistent data and CRUD.

Slide 16



An employee's address using SQL

```
UPDATE employee e, address a
SET a.street_one = "123 Main
Street", a.city = "New York",
a.state = "NY", a.zip_code =
"10023" WHERE e.employee_id =
a.employee_id and e.employee_id
= "183746298"
```

An employee's address using RESTful API

Method: HTTP PUT
URL:
`http://example.com/employee/183746`
Body:

```
{
  "address": {
    "street_one": "123 Main Street",
    "city": "New York",
    "state": "NY",
    "zip_code": "10023"
  }
}
```

 **Remember:** Manipulating information in databases using CRUD is very similar to REST standardized actions. They both use the same approach. Each letter in the acronym can be compared to an HTTP method.

CRUD

PRESENTER 2 SAY:

The term CRUD stands for create, read, update, and delete. It's an abstract concept used to manipulate persistent data, which is another source of errors in REST APIs.

Persistent data lives beyond a request that a server has just performed. Examples of persistent data include names and addresses of VA employees in a database or a file on the server's hard disk. Manipulating information in databases using CRUD is very similar to REST APIs and standardized actions. They both use the same approach. Each letter in the CRUD acronym can be compared to an HTTP method. An HTTP GET would be similar to read, an HTTP PUT would be similar to update, an HTTP POST would be similar to create, and an HTTP DELETE would be the same as delete. On the screen you'll notice a comparison of a SQL update operation and an HTTP PUT.

Slide 17



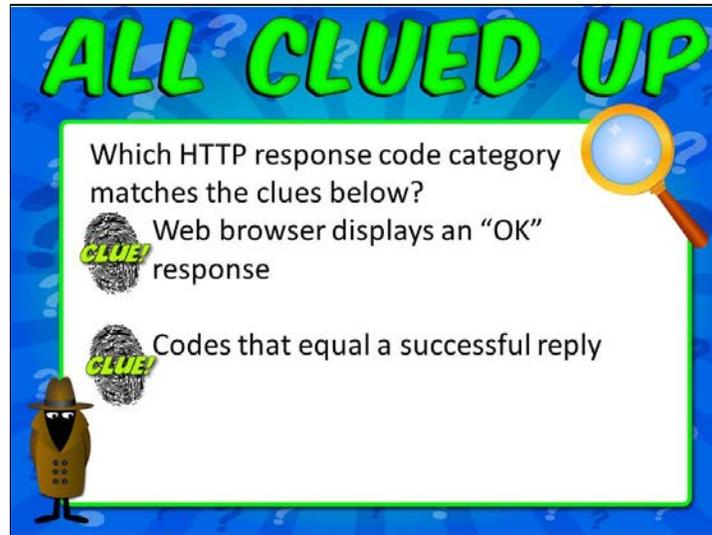
PRESENTER 1 SAY:

Now that we've discussed the different tools used to overcome and diagnose problems in REST, let's test your knowledge with a game of All Clued Up!

We'll need your participation, so pay close attention. Clues to an HTTP response code category will be displayed on the screen and you'll have the chance to name the category we're describing. Once you've decided on an answer, type it in the Chat window located on the left side of your screen.

Here's the first description.

Slide 18



PRESENTER 1 SAY:

Which HTTP response code category on the screen matches the clues below? Your clues are:

- A web browser displays a response, "OK"
- Codes that equal a successful reply.

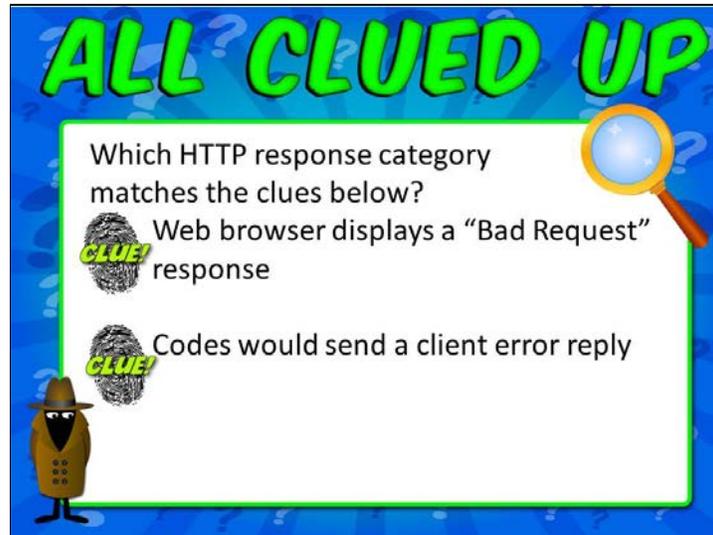
PRESENTER 2 SAY:

Take your time and think about the categories we just discussed and type your answer in the Chat window. Remember there are five categories and they're distinguished by the first digit in each response. HTTP response codes are classified in categories to allow for uniform responses.

If you guessed category two, you're correct! Category two codes indicate the action requested by the client was received and handled successfully. We had a slide on HTTP response code categories earlier. You can access the course slides handout if you want to reference it.

Let's move on to our next set of clues.

Slide 19



PRESENTER 1 SAY:

Which HTTP response category on the screen matches the clues below? Your clues are:

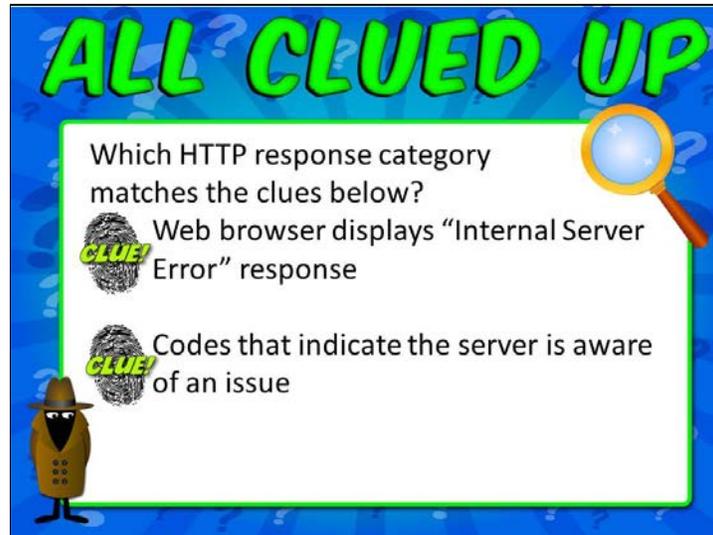
- A web browser displays a response, “Bad Request”
- Client error codes.

PRESENTER 2 SAY:

When you’re ready to respond, type your answer in the Chat window.

If you guessed category four, you’re correct! Category four codes send client error responses where the client seems to have an error. Let’s move on to our next set of clues.

Slide 20



PRESENTER 1 SAY:

Which HTTP response category on the screen matches the clues below? Your clues are:

- A web browser displays a response, “Internal Server Error”
- Codes that indicate the server is aware of an issue.

PRESENTER 2 SAY:

When you’re ready to respond, type your answer in the Chat window.

If you guessed category five, you’re correct! Category five codes indicate that the server failed to complete a valid request. Remember these three HTTP response code categories—two, four, and five—are categories we suggest creating for your REST APIs.

Slide 21



PRESENTER 2 SAY:

Now, let's discuss security issues that you may encounter while using REST APIs.

Slide 22



PRESENTER 2 SAY:

Since REST works with the Internet, it has the same security issues as anything else online. However, there are some technologies we can use to overcome these problems and help secure REST applications.

An important considerations for securing REST applications is to use the right security protocol. In practice, most RESTful applications use HTTP as a transport layer. When hosting REST applications, make sure and use Hypertext Transfer Protocol Secure, or HTTPS, instead of HTTP. As the name mentions, HTTPS is the secure way of transferring important information, such as addresses, credit card information, and Social Security numbers. Unlike HTTP, HTTPS isn't a protocol. It's a result of adding an additional level to HTTP with a secure sockets layer, or S-S-L, and a transport layer security, or T-L-S, protocol or SSL/TLS protocol which secures the HTTP protocol. HTTPS is implemented at a server level, not when developing the REST applications, and rides on top of HTTP. If you think of HTTP as a bridge, REST would be the cars driving on top. Once the bridge is secured through HTTPS, then REST is also safe.

Let's now talk about using different authentication protocols with REST APIs.

Slide 23



PRESENTER 2 SAY:

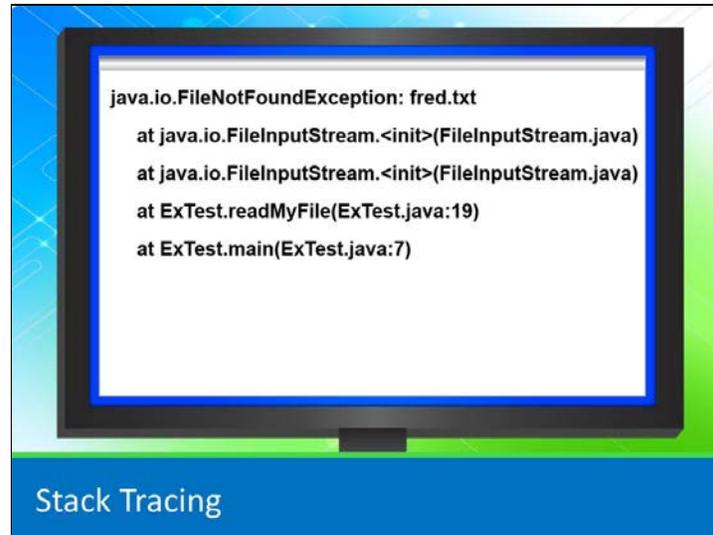
Public key infrastructure, or P-K-I, provides methods for securely authenticating users and encrypting data. Popular examples of PKI protocols include TLS and SSL. They are used to establish encrypted links between a server, such as a website, and a client, such as a Firefox browser. PKI protocols are used for securing users' data, such as Personal Identifiable Information, P-I-I, and Personal Health Information, P-H-I, on the Internet.

PKI can be used with different types of authentication, including basic authentication, which sends passwords in clear text. This authentication method is the easiest to apply, but isn't very secure and offers the least amount of security compared to other protocols. When using basic authentication, you're only sending a username and password that is easily intercepted. Because of inherent lack of security, we do not recommend using basic authentication when sending personal information.

Alternately, you can use more complex authentication methods, such as certificate-based authentication when sending personal information. For example, if you use SSL, when a browser tries to access a secured website, the browser and web server establish a five-step process called an SSL handshake. First, the browser connects to a secured web server with SSL and asks the server to identify itself. Second, the server identifies itself by sending a copy of the SSL certificate. Third, the browser checks the certificate to see if it's valid or expired. If it's valid, the certificate is encrypted and a session key is sent back. Fourth, the session key is decrypted and sends back an acknowledgement. Fifth, the server and browser encrypt all transmitted data with the session key and begin the session. While this process sounds lengthy, it's invisible to the user and happens instantly.

SSL creates a trusted online environment where customers feel confident leaving their personal information. Always make sure you use the authentications methods required by VA policy.

Slide 24



```
java.io.FileNotFoundException: fred.txt
at java.io.FileInputStream.<init>(FileInputStream.java)
at java.io.FileInputStream.<init>(FileInputStream.java)
at ExTest.readMyFile(ExTest.java:19)
at ExTest.main(ExTest.java:7)
```

Stack Tracing

PRESENTER 2 SAY:

We want to talk about one more item relating to security and REST APIs, and that's stack tracing.

Stack traces are reports of active stack frames at a certain point in time during the execution of a program. They're often used during debugging to report error logs to the programmer. They show where the error occurs from a certain function and narrow the error down to a specific line. In REST APIs, it's okay to create a log, but don't send stack trace messages back to the user and client. If the stack traces are returned, they become an information leak. This leak reveals information about your implementation and attackers can gain certain information about your system. Hackers may also be able to use a debugging-based approach to exploit flaws in your site and break into your database. On the screen, we've shown you an example of a stack trace that would be printed when a developer is trying to access a non-existent file.

Slide 25

Poll

What is the best way to secure REST APIs when PII and/or PHI data are being transferred to or from the server?

- a. Use HTTPS instead of HTTP
- b. Use certificate-based authentication
- c. Use basic authentication

PRESENTER 1 SAY:

Now that we've talked about securing REST APIs, we'd like to see what you've learned. So, based on our discussion, what is the best way to secure REST APIs when PII and/or PHI data are being transferred to or from the server?

Your choices are:

- Use HTTPS instead of HTTP
- Use certificate-based authentication
- Use basic authentication

Select what you think is the correct answer.

PRESENTER 2 SAY:

Let's look at how you responded.

If you guessed b. Use certificate-based authentication, you are correct!

PRESENTER 1 SAY:

Know that your level of security will determine which protocol you use on your REST API. Using certificate-based authentication secures a user's personal information. Now, we'll move on and discuss best practices when using REST.

Slide 26



PRESENTER 1 SAY:

We've discussed some best practices for security REST APIs. Now, let's discuss some other general best practices to use in developing REST APIs.

Slide 27



PRESENTER 1 SAY:

We understand that there are several best practices to review but have narrowed our best practices down to four. First, create usable REST APIs; second, follow suggested requirements to create valuable REST APIs; third, use version numbers in REST APIs; and fourth, limit fields that are returned by REST APIs.

Slide 28



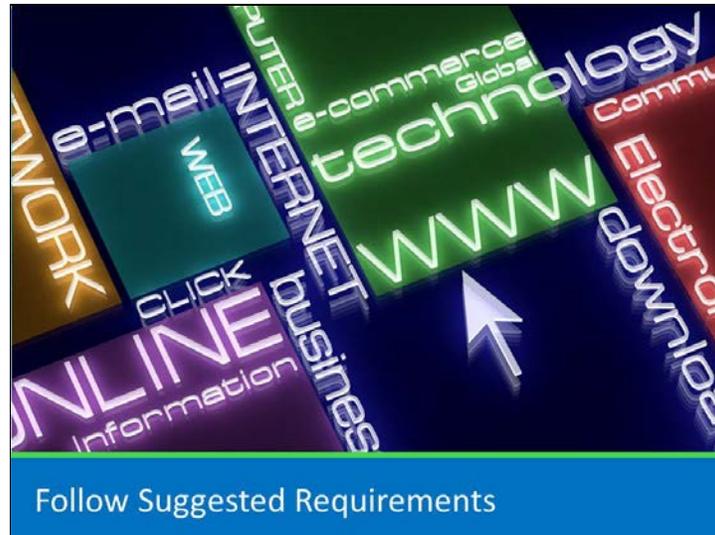
PRESENTER 2 SAY:

Our first and foremost tip for REST best practices is to create REST APIs that are useful and usable to others. Your REST API could be secure, RESTful, and discoverable, but if it can't be used by other programmers, then they will find another REST API to use. The key to creating useful REST APIs is to think about how others would use it. Then, design it for them.

We suggest using RESTful API Modeling Language, known as RAML. It's a way of creating REST APIs that are readable by both humans and computers. RAML focuses on describing resources, methods, and responses in a clear manner. It's aimed to encourage better REST API arrangements. To learn more about how to implement RAML in your REST APIs, check out their website at raml.org. A link to RAML's website has been included in your handout.

We also suggest approaching developers and asking them what features they would want in a REST API and how they would develop it. REST API developers have started focusing more on experience and less on interface.

Slide 29



PRESENTER 1 SAY:

For our second tip, we'd like to talk about standards to incorporate in your REST APIs to make them practical and usable. There are several REST API design opinions that are vague in their guidelines or don't apply to real-world situations. This makes the standards hard to understand. Let's discuss three requirements that a valuable REST API should strive for.

First, a REST API should be created using web standards where they make sense. There is no need to use criteria if it isn't specific to the REST API that you're creating.

Second, a REST API should be simple, intuitive, and reliable to make adoption easy. We mentioned earlier that your REST API is only as good as its documentation, so create REST APIs that can be easily used by others.

Third, a REST API should be well organized while balancing other demands. It's important to keep in mind that REST APIs will have other requirements, such as security and HTTPS requirements that will need to be met. Carefully consider what your options are and create a REST API that meets your needs.

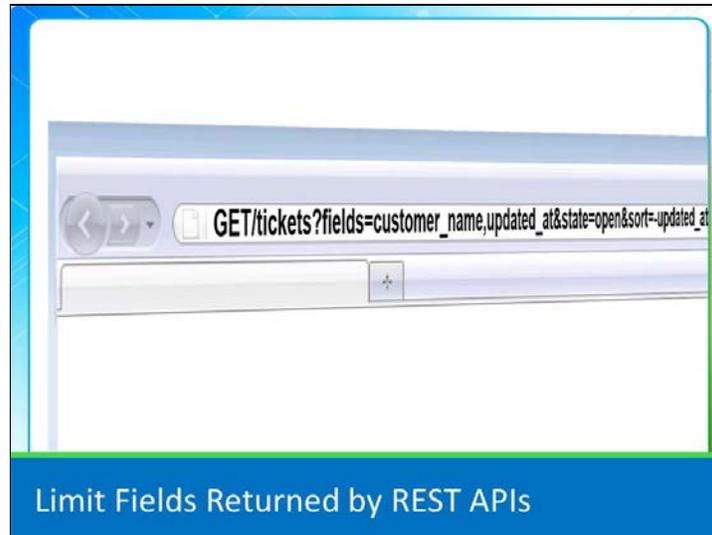
Slide 30



PRESENTER 2 SAY:

Our third tip is to use version numbers when creating your REST APIs. This will help prevent invalid calls being used from endpoints that have been updated. It will also make for an easier transition between older versions of REST APIs that are being offered for a short period of time. It's a best practice to include the version number in the URL, for example v-1 Also use the date-base sub versions in the REST API, for example 2020-10-30. By using date-base sub versions, a REST API can be chosen using a custom HTTP action with the date-base sub version in the call. Like all programming, change is inevitable. Using version numbers will help manage change, organize your REST API, and give you version control over older documents.

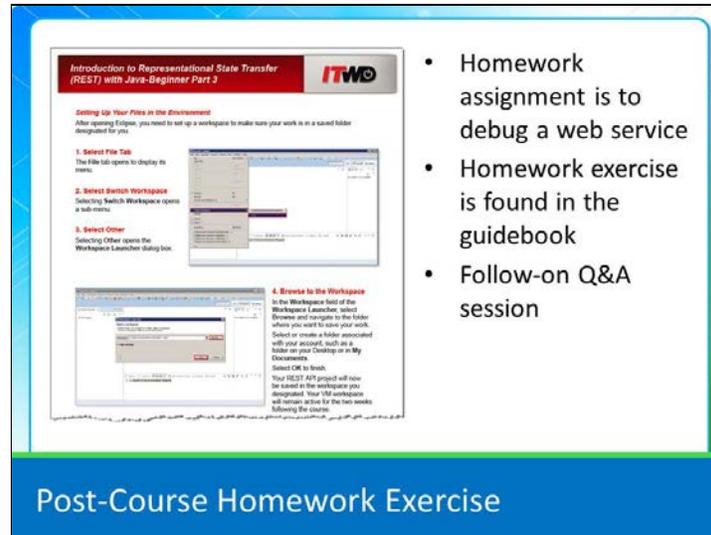
Slide 31



PRESENTER 1 SAY:

Our last tip is to limit which fields are returned to the user by the REST API. It's unnecessary for the user to see the entire resource. As a developer, choosing which fields are returned minimizes traffic and speeds up returned results. We suggest creating a query parameter that uses a comma to separate lists of information. For example, the following request on the screen would retrieve just enough information to display a sorted listing of customers with open tickets and the last date they were updated. Limiting returned fields also gives users the exact information they need, leaving out anything unnecessary.

Slide 32



- Homework assignment is to debug a web service
- Homework exercise is found in the guidebook
- Follow-on Q&A session

PRESENTER 1 SAY:

For the homework practice, we're going to ask you to debug a REST web service using Fiddler and a web-based application on your own. You'll also be able to review debugging and become more familiar with how debugging tools work. You'll do this on your own after this course is over. It's a self-paced exercise set up for you in a virtual environment, and we've provided you with a guidebook that will step you through everything. In the guidebook, you'll find sets of step-by-step directions for you to access VMware, where we have set up this homework assignment, and complete the exercise. We've provided all of the software tools in VMware that you need to complete the assignment.

For those of you who haven't done this before, it may sound a little scary, but the guidebook will walk you through it.

If you have any issues or questions about the assignment, we'll have a separate Q&A session that you can dial into and ask questions.

Slide 33



PRESENTER 1 SAY:

We've covered a lot today. We first talked about common problems in REST and ways to overcome by using debugging tools like Fiddler. We explained HTTP codes and talked about how they're useful to diagnose problems and talked about CRUD. Next, we talked about securing REST APIs and stack tracing and the different protocols that keep REST APIs secure. Along the way, we discussed best practices for REST APIs that will help you in your own creation.

PRESENTER 2 SAY:

Now that we've given a summary, let's open up the conversation to you and answer any questions you might have.

Slide 34



PRESENTER 1 SAY:

Now it's time to go over your questions.

Q1: What is the number one security issue that REST APIs are facing, and how can we as programmers learn to secure our REST APIs against this issue?

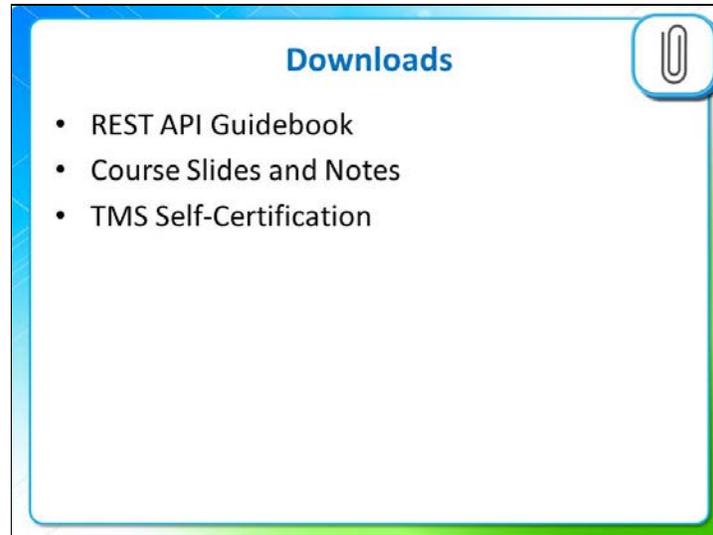
A1: The top security issue that REST APIs are facing is something called SQL Injection. This attack uses a code injection technique. It attacks by using blatant SQL statements, which are inserted into an entry field. This in turn dumps database contents to the attacker. Imperva, a security company, did a study in 2012 and found that RESTful APIs receive four attacks per month. One way of reducing this risk is to use prepared statements on your database. A prepared statement is used to execute the same database statements repeatedly with high proficiency. Another way to reduce this risk is to perform input validation on client requests. We've also included a link in the handout that talks about how to secure REST APIs.

(https://www.owasp.org/index.php/REST_Security_Cheat_Sheet)

Q2: Is there any web technology that we should avoid using in our REST APIs?

A2: It's important not to try and create your own security framework. Use an existing, open-source library that has been peer-reviewed and tested. We also suggest implementing the security techniques that we've discussed today.

Slide 35



PRESENTER 1 SAY:

We've got three handouts for you to download today. We've got a guidebook for your self-paced homework assignment, the full course transcript, and the directions for self-certifying in the TMS that you took this course.

To access the downloads, select the paperclip icon in the Lync toolbar above the list of participants. Next, select the right arrow next to the file in the Attachments pop-up menu. From there you can select Open or select Save As to navigate to a location where you can save the download.

Slide 36



PRESENTER 1 SAY:

Thank you for joining us in today's session. As we mentioned earlier, this is the last in our series of beginner courses on REST.

Be sure to check the training calendar on the ITWD Portal for more great IT Campus courses!

Slide 37

TMS Self-Certification

- Log in to the TMS <https://www.tms.va.gov>
- Enter **3878054** in the Search Catalog field on your TMS home page and select the **Go** button
- Select the **Start Course** button
- Select the **Yes** button
- Select the **OK** button
- Select the **Close Window** button
- **NEW!** Complete the course evaluation survey that is on your TMS To-Do List

For assistance, contact the **TMS Help Desk:**
vatmshelp@va.gov or **1-866-496-0463**

PRESENTER 1 SAY:

Take a minute to self-certify to get credit for your participation in this training. TMS self-certification instructions are provided on the slide.

Once you're self-certified, you will receive a link to complete an evaluation for this training.

We will leave the instructions up for a couple minutes to allow you time to self-certify.

This concludes *Introduction to Representational State Transfer (REST) with JAVA, Beginner Part 3* and our series on REST. We'll be adding more training, so keep your eyes open for those announcements. Thank you for your participation.