

Serviceangebote einfach erstellt - mit EGL

Jeder spricht heute von SOA und Web-Services. Da darf EGL in der Diskussion natürlich nicht fehlen. EGL erlaubt es Ihnen, zwei Arten von Web-Services zu implementieren, EGL Services und EGL Web-Services.

Im EGL-Service ist die gesamte Anwendung in EGL geschrieben. Im EGL Web-Service ist der Service selbst in EGL geschrieben, den Zugriff auf diesen Service können Sie aber sowohl durch EGL als auch Nicht-EGL vornehmen. In diesem Artikel zeigen wir Ihnen an einem Beispiel, wie Sie einen neuen Web-Service mit EGL konstruieren können.

Services unterscheiden sich von den traditionellen monolithischen Programmen; sie sind modular und haben eine wohl definierte Schnittstelle für die Daten Ein- und Ausgabe. Dadurch werden sie zu Bausteinen, aus denen Sie Ihre Anwendung einfach zusammensetzen können. Die Modularität hat außerdem den angenehmen Nebeneffekt, dass die Pflege Ihrer Anwendungen damit erleichtert wird und Sie die einzelnen Elemente auch für andere Anwendungen wieder verwenden können.

Es gibt eine Vielzahl von Technologien und Sprachen, mit denen Sie diese Services erstellen können. Ein gutes Beispiel dafür sind IBM® Rational® Application Developer™ (RAD) und EGL, die Ihnen Tools zum Erstellen, Testen und Einsatz von Web-Services bieten. In diesem Beispiel zeigen wir Ihnen einen Web-Service, der mit EGL erstellt wird, der aber auch von anderen Nicht-EGL Clients benutzt (konsumiert) werden kann.

Hier ist erst einmal ein Überblick über die Schritte, die erforderlich sind, um einen EGL Web-Service zu erstellen:

1. Sie konfigurieren ein EGL Web-Projekt für Web-Services.
2. Sie kreieren eine EGL Web-Service-Datei und schreiben die Web-Service-Logik.
3. Sie generieren den Web-Service und seine Schnittstellenbeschreibung.
4. Sie benutzen die Web-Services-Tools, um Ihren Web-Service interaktiv zu testen.

Nun folgen die Einzelheiten darüber, wie Sie einen Web-Service mit EGL erstellen und testen.

Als erstes kreieren Sie ein neues EGL-Webprojekt, indem Sie **Datei > Neu > Andere** auswählen, **EGL** erweitern und **EGL-Webprojekt** anklicken.

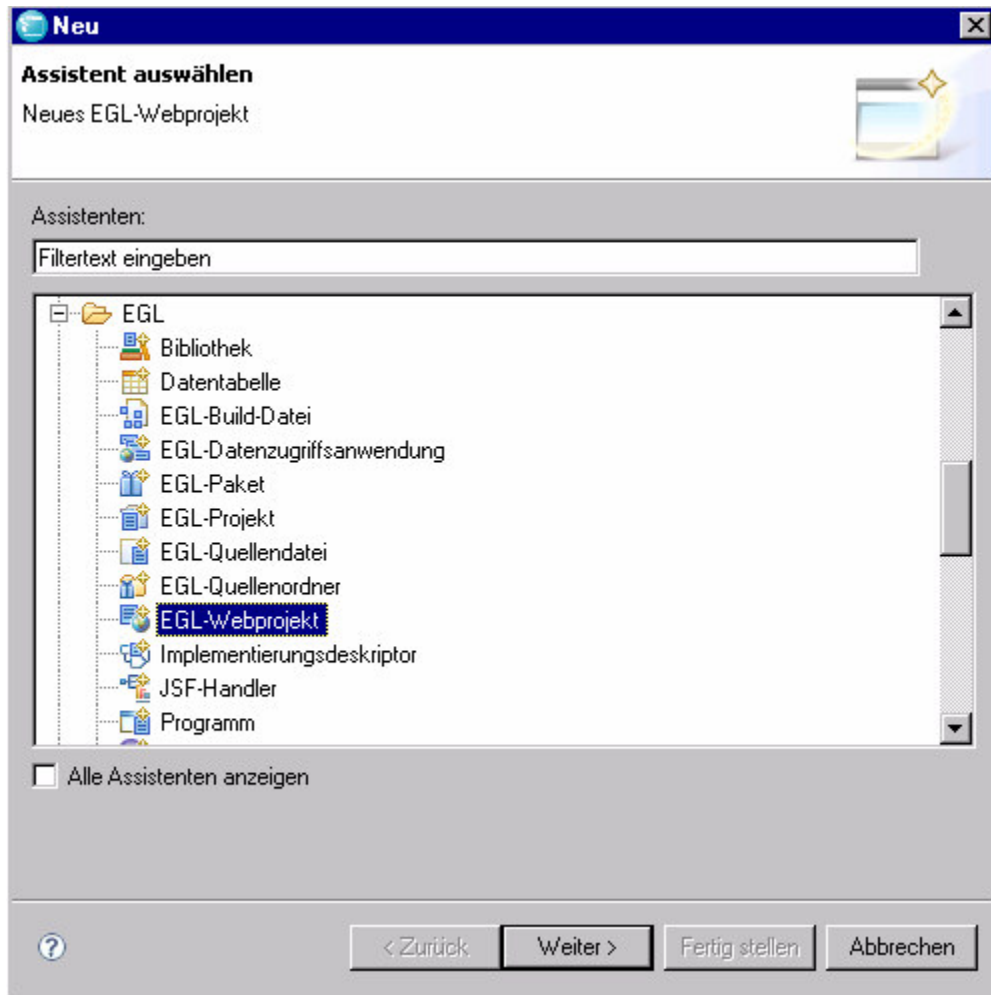


Abbildung 1 – Auswahl des EGL-Webprojekts

Weiter zeigt Ihnen den Dialog für ein neues EGL-Webprojekt an. Geben Sie Ihrem Projekt einen Namen, wir benutzen hier **Beispiel_Service**, und stellen Sie sicher, dass **Neue Deskriptoren für Projekt-Build automatisch erstellen** angeklickt ist. Als letztes müssen Sie auf dieser Seite noch die EAR-Zugehörigkeit spezifizieren. Dafür klicken Sie **Projekt zu einem EAR-Projekt hinzufügen** und geben den EAR-Projektnamen an; üblicherweise wird dafür EAR an den Projektnamen drangehängt, bei uns also **Beispiel_ServiceEAR**. Klicken Sie **Fertig stellen**.

Abbildung 2 – Assistent für ein neues EGL-Webprojekt

Falls Sie eine Meldung erhalten, ob Sie die J2EE-Perspektive öffnen wollen, klicken Sie **Nein**, Sie wollen ja in der Web-Perspektive bleiben.

Als nächstes müssen Sie die Logik für den Service erstellen. Dafür kreieren Sie eine Datei und stecken sie in ein Paket. Der Assistent **Neuer EGL-Serviceabschnitt** hilft Ihnen dabei. Erweitern Sie Ihr Projekt, klicken Sie **EGLSource** mit der rechten Maustaste an und wählen Sie **Neu > Service** aus. Geben Sie nun **services** als Paket an und einen Namen für die EGL Quellendatei; wir nennen sie in unserem Beispiel PLZService. Klicken Sie **Fertig stellen**.

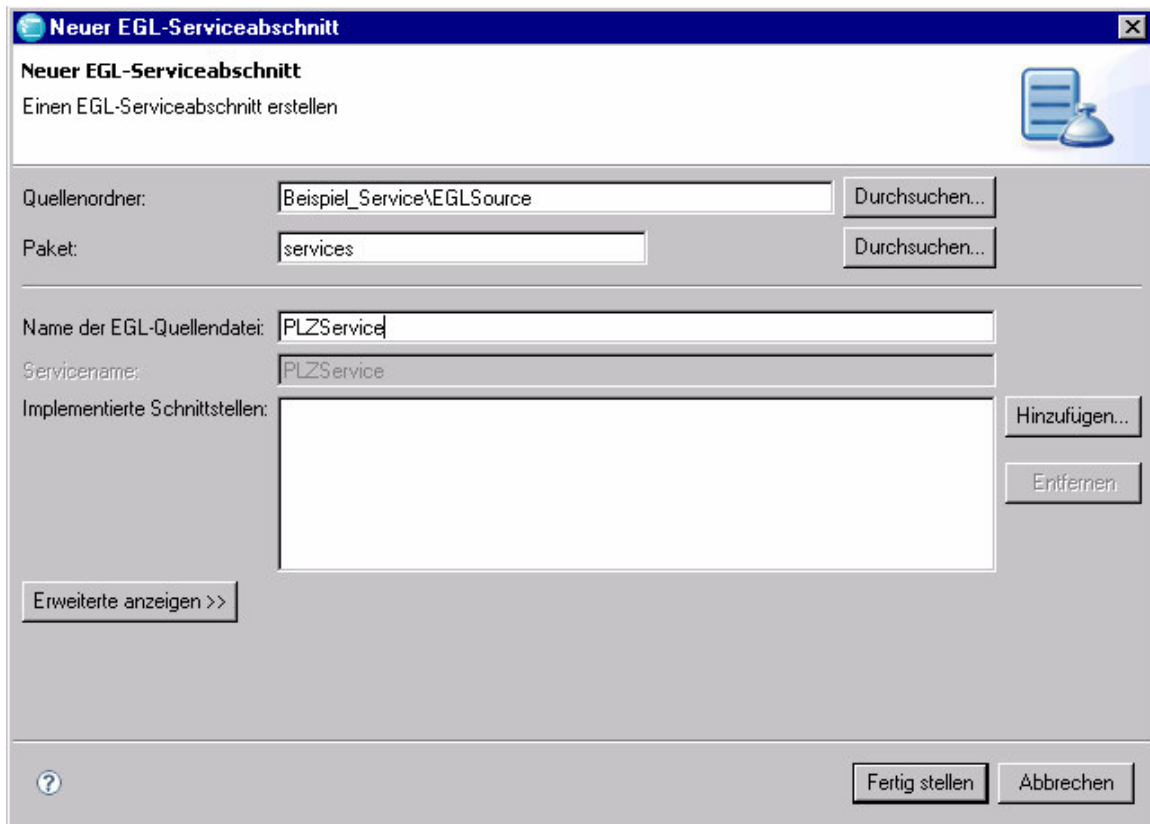


Abbildung 3 – Assistent für einen neuen EGL-Serviceabschnitt

Damit haben Sie jetzt zwar die Quelle erstellt, sie hat aber nur den Rahmen als Inhalt.

```

package services;

// service
service PLZService

    // Variable Declarations
    variableName string;

    // Function Declarations
    function functionName ()
    end

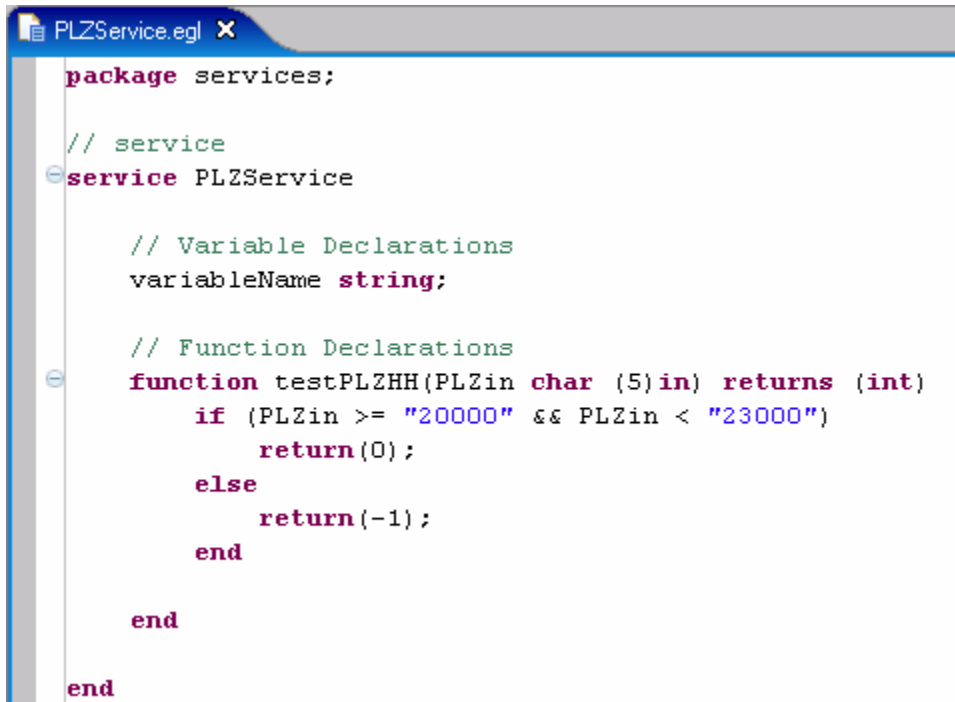
end

```

Abbildung 4 – Der generierte Rahmen

In diesen Rahmen fügen Sie nun Ihre Logik ein. Wir prüfen in unserem Beispielservice, ob eine angegebene Postleitzahl im Hamburger Bereich liegt. Da nur Zahlen zwischen 20000 bis 23000 in Frage kommen, prüft unser Code, ob

der eingegebene Wert in diesen Zahlenbereich fällt. Der fertige Service sieht dann so aus:



```
package services;

// service
service PLZService

    // Variable Declarations
    variableName string;

    // Function Declarations
    function testPLZHH(PLZin char (5)in) returns (int)
        if (PLZin >= "20000" && PLZin < "23000")
            return(0);
        else
            return(-1);
        end
    end

end
```

Abbildung 5 – Die Beispielfunktion

Benutzen Sie **Strg+S**, um den Quellencode zu sichern und anschließend **Strg+G**, um den EGL-Serviceabschnitt (den Java™ Code) zu generieren.

Bevor Sie Ihren neuen Service benutzen können, müssen Sie ihn der EGL-Umgebung beschreiben. Die Beschreibung des Services erfolgt durch den Implementierungsdeskriptor, den wir mit Hilfe eines Assistenten erstellen. Den Assistenten starten Sie, indem Sie **EGLSource** mit der rechten Maustaste anklicken und **Neu > Implementierungsdeskriptor** auswählen. Geben Sie Ihrer Implementierungsdeskriptordatei einen Namen. In unserem Beispiel benutzen wir ServiceProjektDeskriptor.

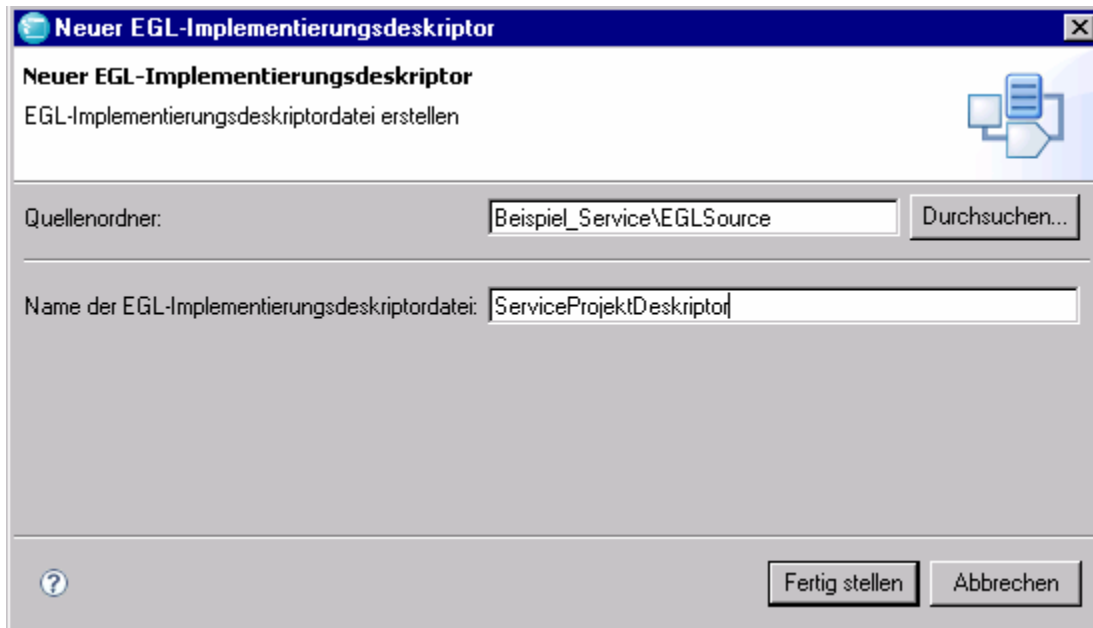


Abbildung 6 – Dialog für den EGL-Implementierungsdeskriptor

Klicken Sie **Fertig stellen**. Damit wird Ihnen die Übersicht über den EGL Implementierungsdeskriptor angezeigt. Klicken Sie auf die Registerkarte **Web-Service-Implementierung** am unteren Rand des Anzeigebereichs. Diese Registerkarte benutzen Sie immer dann, wenn Sie einen Web-Service zur Verfügung stellen wollen. Die Registerkarte **Service-Client-Bindings** benutzen Sie nur, wenn Sie auf einen bestehenden Web-Service zugreifen wollen.



Auf der Seite **Web-Services-Implementierung** klicken Sie die Taste **Hinzufügen**. Das zeigt Ihnen den Dialog **Web-Service hinzufügen** an. Wählen Sie aus der Liste **EGL-Serviceabschnitte gefunden** Ihren Service aus und klicken Sie **Hinzufügen -->**. Damit wird der Service in die Liste **EGL-Serviceabschnitte, die als Web-Services erstellt werden sollen** übernommen.

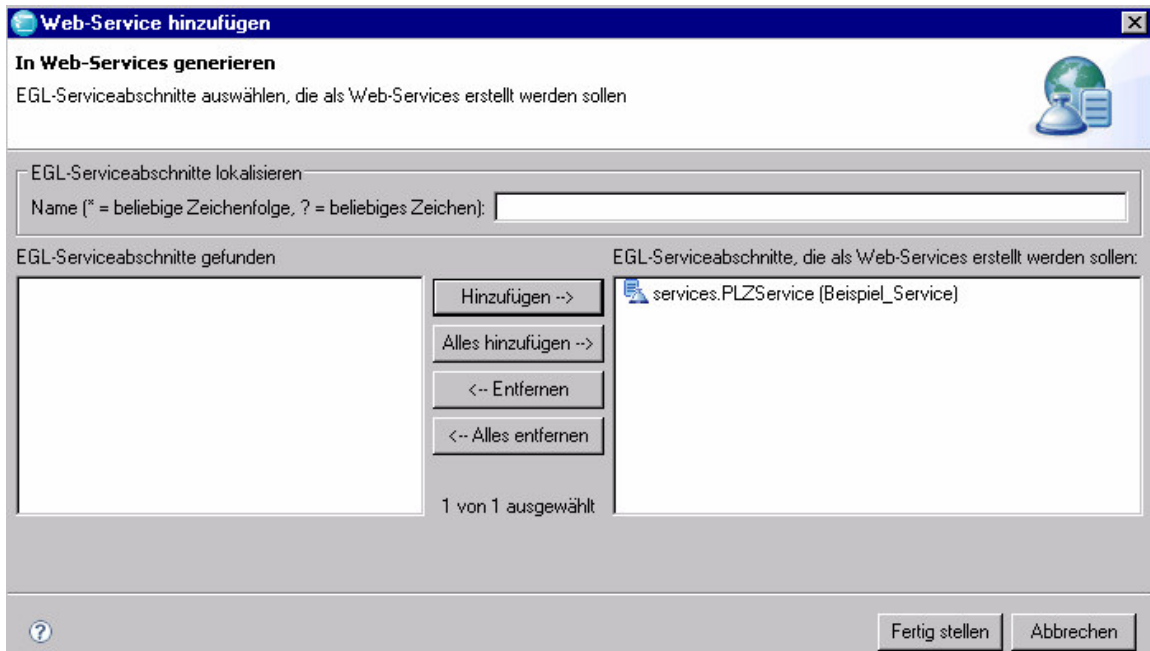


Abbildung 7 – Auswahl und Hinzufügen des Serviceabschnitts

Klicken Sie **Fertig stellen**, speichern Sie Ihre Deskriptor-Datei, in unserem Beispiel `ServiceProjektDeskriptor.egldd` und schließen Sie den Editor.

Nun müssen Sie den Deployment-Deskriptor zur Build-Datei hinzufügen, bei uns ist das `Beispiel_Service.eglbl`, damit die EGL Build-Umgebung weiß, welchen Deployment-Deskriptor Sie benutzen wollen. Erweitern Sie dafür **EGL Source** und öffnen Sie die **.eglbl** Datei.

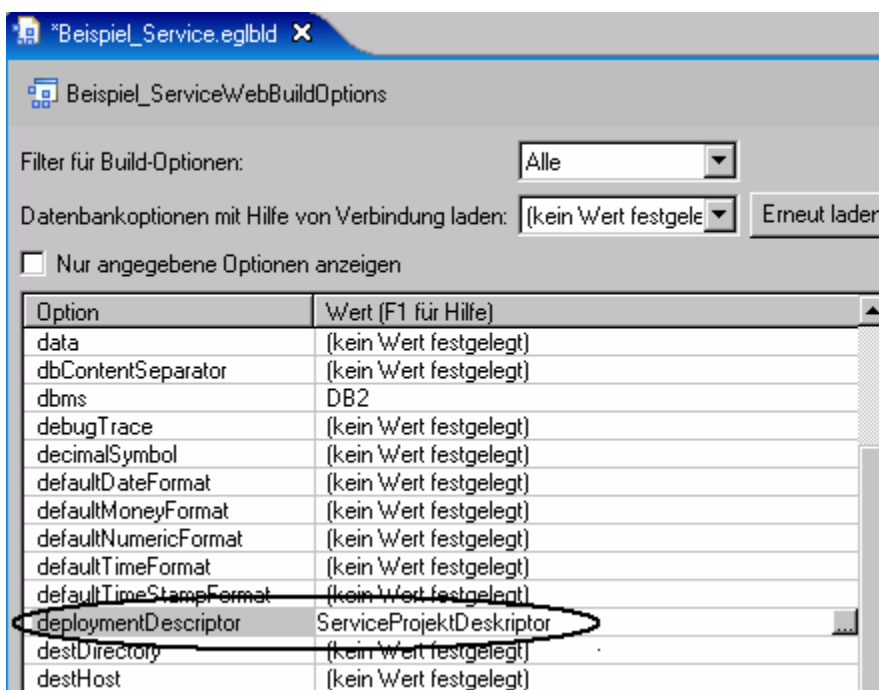


Abbildung 8 – Auswahl der Build-Optionen

Hier müssen Sie jetzt erst einmal **Nur angegebene Optionen anzeigen** abwählen. Suchen Sie anschließend in der Liste der Optionen den Eintrag **deploymentDescriptor** heraus und klicken Sie auf **(kein Wert festgelegt)**. In diesem Feld geben Sie nun den Namen Ihres Deskriptors an. Speichern und schließen Sie die Datei.

Da Sie gerade die Build-Informationen geändert haben, müssen Sie jetzt das EGL-Projekt neu generieren, damit die Änderungen in das Projekt aufgenommen werden können. Klicken Sie Ihr Projekt mit der rechten Maustaste an und wählen Sie **Generieren** aus.

Als letztes müssen Sie nun noch die WSDL-Datei generieren, die den Web-Service und die dazugehörige Schnittstelle beschreibt. Erweitern Sie dafür **EGL source\services** und klicken Sie **Quelle.egl > EGL Services > WSDL-Datei generieren**. In unserem Beispiel hatten wir die Quelle PLZService genannt. Im Dialog **WSDL-Datei erstellen** brauchen Sie nur auf **Fertig stellen** zu drücken.

Damit haben Sie Ihren Web-Service und seine Interface Beschreibung vollständig erstellt. Es bleibt nun noch das Testen. Starten Sie dafür den IBM® WebSphere® Application Server (WAS) V6.1. Fügen Sie Ihr Projekt zum Server hinzu, indem Sie die Registerkarte **Server** klicken, WebSphere Application Server V6.1 mit der rechten Maustaste anklicken und **Projekte hinzufügen und entfernen** auswählen. Im Dialog **Projekte hinzufügen und entfernen** wählen Sie Ihr Projekt aus und drücken die Taste **Hinzufügen >**. Damit erscheint Ihr Projekt in der Liste **Konfigurierte Projekte**. Drücken Sie **Fertig Stellen**.

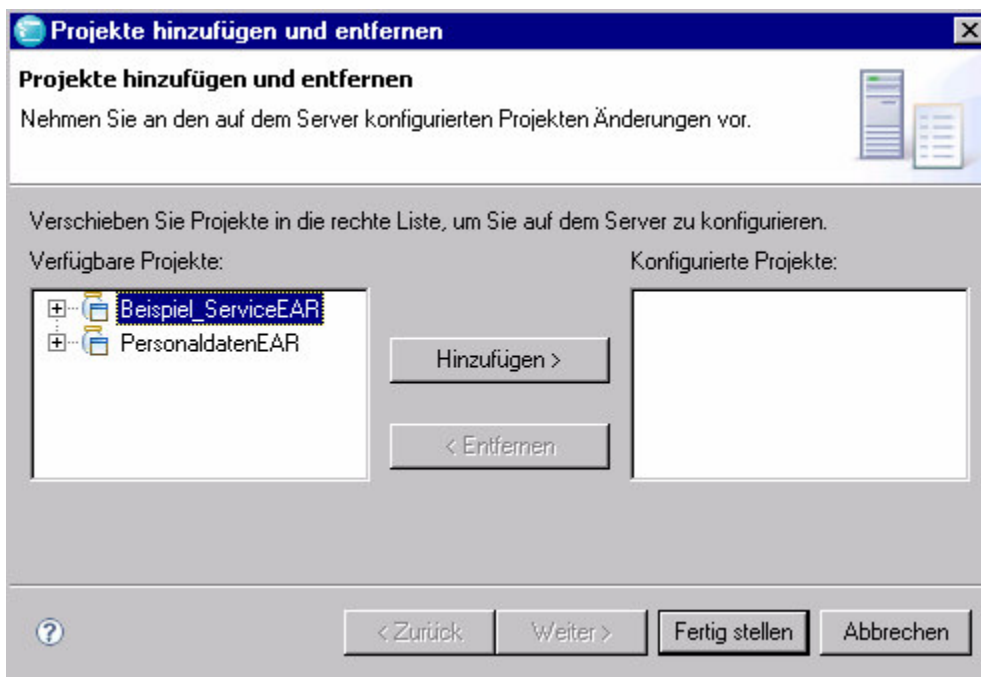


Abbildung 9 – Dialog zum Hinzufügen und Entfernen von Projekten

Klicken Sie Ihre WSDL-Datei mit der rechten Maustaste an und wählen Sie **Web-Services > Mit Web-Services-Explorer testen** aus.

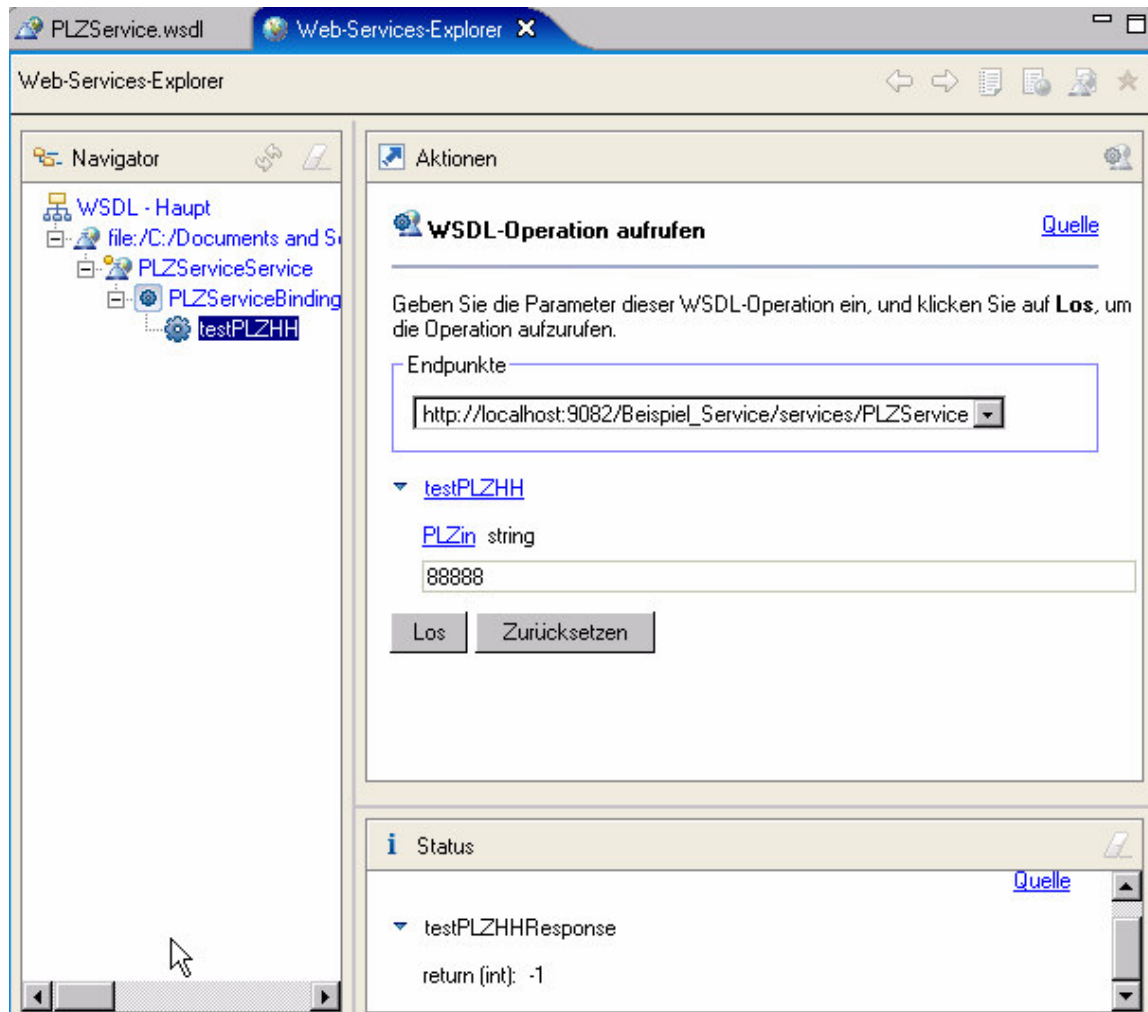


Abbildung 10 – Web-Services-Explorer zum Testen von Web-Services

Erweitern Sie im Navigator im **Web-Services-Explorer** den Binding-Knoten und klicken Sie auf die Service-Funktion. Im rechten Teil des Fensters sehen Sie nun das Eingabefeld für Ihren Webservice-Parameter. Für die unterschiedlichen Werte, die Sie hier eingeben, erhalten Sie, wenn Sie auf **Los** drücken, unter **Status** den Rückgabewert.

Zum Schluss noch ein Hinweis: Falls im Status-Bereich anstelle des Rückgabewerts eine Fehlermeldung erscheint, prüfen Sie den Port des WSDL. Als Standardwert wird hier 9080 benutzt. Falls Sie schon eine andere Versionen von WAS installiert haben, wird die Portnummer automatisch vom WAS-Installationsprogramm erhöht und kann dann beispielsweise 9082 sein. Ändern

Sie den Port im URL des WDSL, so dass er mit der Portnummer des WAS übereinstimmt.

Um herauszufinden, welchen Port WAS benutzt, klicken Sie die Registerkarte **Server** mit der rechten Maustaste an und wählen **Administrationskonsole** aus. In der Administrationskonsole erweitern Sie den Knoten **Server**, klicken auf **Anwendungsserver** und rechts in der Tabelle auf den Server. Rechts unter Übertragungen finden Sie den Knoten **Ports**. Erweitern Sie diesen Knoten und prüfen Sie in der Tabelle der Ports, welche Nummer WC_defaulthost zugeordnet ist.

Weitere Informationen zu Rational Business Developer Extension finden Sie unter folgendem Link:

<http://www.ibm.com/software/awdtools/developer/business/index.html>

Weitere Informationen zu WebSphere Development Studio für iSeries™ finden Sie unter folgendem Link:

<http://www.ibm.com/software/awdtools/iseris>

© Copyright International Business Machines Corporation, 2007. All rights reserved.

Trademarks

IBM, iSeries, Rational, Rational Application Developer, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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